CS106A
Midterm Review

Miles Seiver
1p - 4p
9 February 2014

The plan

- General info
- Karel
- Expression evaluation
- Program tracing
- Simple Java and randomness
- Graphics and MouseEvents
- String manipulation

The plan

- General info
- Karel
- Expression evaluation
- Program tracing
- Simple Java and randomness
- Graphics and MouseEvents
- String manipulation

Midterm logistics

- Midterm is this **Tuesday** from **7pm - 10pm**
- Last name **Aa - Ep:** Braun Auditorium
- Last name **Eq - Na:** Hewlett 200
- Last name **Nb - Zz:** Cemex Auditorium

Exam is open book, open notes, closed computer

The examination is open-book (specifically the course textbook *The Art and Science of Java* and the Karel the Robot coursereader) and you may make use of any handouts, course notes/slides, printouts of your programs or other notes you've taken in the class. You may not, however, use a computer of any kind (i.e., you cannot use laptops on the exam).

Exam is open book, open notes, closed computer

The examination is open-book (specifically the course textbook *The Art and Science of Java* and the Karel the Robot coursereader) and you may make use of any handouts, course notes/slides, printouts of your programs or other notes you've taken in the class. You may not, however, use a computer of any kind (i.e., you cannot use laptops on the exam).

Exam is open book, open notes, closed computer

The examination is open-book (specifically the course textbook *The Art and Science of Java* and the Karel the Robot coursereader) and you may make use of any handouts, course notes/slides, printouts of your programs or other notes you've taken in the class. You may not, however, use a computer of any kind (i.e., you cannot use laptops on the exam).

Coverage

The midterm exam covers the material presented in class up to and including string processing, which we should finish by Friday, February 7, which means that you are responsible for the Karel material plus Chapters 1-6, 8, 9, and the use of mouse listeners from Chapter 10 (sections 10.1-10.4) from *The Art and Science of Java*.

Key topics

Key topics

- Karel

Key topics

- Karel
  - without non-Karel features!

Key topics

- Karel
  - without non-Karel features!
- Primitives

Key topics

- Karel
  - without non-Karel features!
- Primitives
  - `int`, `boolean`, `double`, `char`

Key topics

- Karel
  - without non-Karel features!
- Primitives
  - `int`, `boolean`, `double`, `char`
  - type conversion (typecasting)

Key topics

- Karel
  - without non-Karel features!
- Primitives
  - `int`, `boolean`, `double`, `char`
  - type conversion (typecasting)
- Control structures

Key topics

- Karel
  - without non-Karel features!
- Primitives
  - `int`, `boolean`, `double`, `char`
  - type conversion (typecasting)
- Control structures
  - `if` (`else`), `switch`

Key topics

- Karel
  - without non-Karel features!
- Primitives
  - `int`, `boolean`, `double`, `char`
  - type conversion (typecasting)
- Control structures
  - `if` (`else`), `switch`
  - loops: `for`, `while`, `while` (`true`), `break`, `continue`

Key topics

- Karel
  - without non-Karel features!
- Primitives
  - `int`, `boolean`, `double`, `char`
  - type conversion (typecasting)
- Control structures
  - `if` (`else`), `switch`
  - loops: `for`, `while`, `while` (`true`), `break`, `continue`
- Operators

Key topics

- Karel
  - without non-Karel features!
- Primitives
  - `int`, `boolean`, `double`, `char`
  - type conversion (typecasting)
- Control structures
  - `if` (`else`), `switch`
  - loops: `for`, `while`, `while` (`true`), `break`, `continue`
- Operators
  - (`+`, `%`, `=`, `<=`, `>`, `==`, `&&`, `||`, `!`, etc.)

Key topics

- Karel
  - without non-Karel features!
- Primitives
  - `int`, `boolean`, `double`, `char`
  - type conversion (typecasting)
- Control structures
  - `if` (`else`), `switch`
  - loops: `for`, `while`, `while` (`true`), `break`, `continue`
- Operators
  - (`+`, `%`, `=`, `<=`, `>`, `==`, `&&`, `||`, `!`, etc.)
- Variables

Key topics

- Karel
  - without non-Karel features!
- Primitives
  - `int`, `boolean`, `double`, `char`
  - type conversion (typecasting)
- Control structures
  - `if` (`else`), `switch`
  - loops: `for`, `while`, `while` (`true`), `break`, `continue`
- Operators
  - (`+`, `%`, `=`, `<=`, `>`, `==`, `&&`, `||`, `!`, etc.)
- Variables
  - constants

Key topics

- Karel
  - without non-Karel features!
- Primitives
  - `int`, `boolean`, `double`, `char`
  - type conversion (typecasting)
- Control structures
  - `if` (`else`), `switch`
  - loops: `for`, `while`, `while` (`true`), `break`, `continue`
- Operators
  - (`+`, `%`, `=`, `<=`, `>`, `==`, `&&`, `||`, `!`, etc.)
- Variables
  - constants
  - pass by reference vs. by value

Key topics

- Karel
  - without non-Karel features!
- Primitives
  - `int`, `boolean`, `double`, `char`
  - type conversion (typecasting)
- Control structures
  - `if` (`else`), `switch`
  - loops: `for`, `while`, `while` (`true`), `break`, `continue`
- Operators
  - (`+`, `%`, `=`, `<=`, `>`, `==`, `&&`, `||`, `!`, etc.)
- Variables
  - constants
  - pass by reference vs. by value
  - local vs. instance

Key topics

- Karel
  - without non-Karel features!
- Primitives
  - `int`, `boolean`, `double`, `char`
  - type conversion (typecasting)
- Control structures
  - `if` (`else`), `switch`
  - loops: `for`, `while`, `while` (`true`), `break`, `continue`
- Operators
  - (`+`, `%`, `=`, `<=`, `>`, `==`, `&&`, `||`, `!`, etc.)
- Variables
  - constants
  - pass by reference vs. by value
  - local vs. instance
  - scope

Key topics

- Karel
  - without non-Karel features!
- Primitives
  - `int`, `boolean`, `double`, `char`
  - type conversion (typecasting)
- Control structures
  - `if` (`else`), `switch`
  - loops: `for`, `while`, `while` (`true`), `break`, `continue`
- Operators
  - (`+`, `%`, `=`, `<=`, `>`, `==`, `&&`, `||`, `!`, etc.)
- Variables
  - constants
  - pass by reference vs. by value
  - local vs. instance
  - scope
- Methods

Key topics

- Karel
  - without non-Karel features!
- Primitives
  - `int`, `boolean`, `double`, `char`
  - type conversion (typecasting)
- Control structures
  - `if` (`else`), `switch`
  - loops: `for`, `while`, `while` (`true`), `break`, `continue`
- Operators
  - (`+`, `%`, `=`, `<=`, `>`, `==`, `&&`, `||`, `!`, etc.)
- Variables
  - constants
  - pass by reference vs. by value
  - local vs. instance
  - scope
- Methods
  - `return` statements

Key topics

- Karel
  - without non-Karel features!
- Primitives
  - `int`, `boolean`, `double`, `char`
  - type conversion (typecasting)
- Control structures
  - `if` (`else`), `switch`
  - loops: `for`, `while`, `while` (`true`), `break`, `continue`
- Operators
  - (`+`, `%`, `=`, `<=`, `>`, `==`, `&&`, `||`, `!`, etc.)
- Variables
  - constants
  - pass by reference vs. by value
  - local vs. instance
  - scope
- Methods
  - `return` statements
  - parameters

Key topics

- Karel
  - without non-Karel features!
- Primitives
  - `int`, `boolean`, `double`, `char`
  - type conversion (typecasting)
- Control structures
  - `if` (`else`), `switch`
  - loops: `for`, `while`, `while` (`true`), `break`, `continue`
- Operators
  - (`+`, `%`, `=`, `<=`, `>`, `==`, `&&`, `||`, `!`, etc.)
- Variables
  - constants
  - pass by reference vs. by value
  - local vs. instance
  - scope
- Methods
  - `return` statements
  - parameters
- `String`

Key topics

- Karel
  - without non-Karel features!
- Primitives
  - `int`, `boolean`, `double`, `char`
  - type conversion (typecasting)
- Control structures
  - `if` (`else`), `switch`
  - loops: `for`, `while`, `while` (`true`), `break`, `continue`
- Operators
  - (`+`, `%`, `=`, `<=`, `>`, `==`, `&&`, `||`, `!`, etc.)
- Variables
  - constants
  - pass by reference vs. by value
  - local vs. instance
  - scope
- Methods
  - `return` statements
  - parameters
- `String`
- Graphics and animation

Key topics

- Karel
  - without non-Karel features!
- Primitives
  - `int`, `boolean`, `double`, `char`
  - type conversion (typecasting)
- Control structures
  - `if` (`else`), `switch`
  - loops: `for`, `while`, `while` (`true`), `break`, `continue`
- Operators
  - (`+`, `%`, `=`, `<=`, `>`, `==`, `&&`, `||`, `!`, etc.)
- Variables
  - constants
  - pass by reference vs. by value
  - local vs. instance
  - scope
- Methods
  - `return` statements
  - parameters
- `String`
- Graphics and animation
- Mouse interaction

Key topics

- Karel
  - without non-Karel features!
- Primitives
  - `int`, `boolean`, `double`, `char`
  - type conversion (typecasting)
- Control structures
  - `if` (`else`), `switch`
  - loops: `for`, `while`, `while` (`true`), `break`, `continue`
- Operators
  - (`+`, `%`, `=`, `<=`, `>`, `==`, `&&`, `||`, `!`, etc.)
- Variables
  - constants
  - pass by reference vs. by value
  - local vs. instance
  - scope
- Methods
  - `return` statements
  - parameters
- `String`
- Graphics and animation
- Mouse interaction
- `RandomGenerator`

Other tips

Other tips

- Style doesn’t matter. You don’t need to comment.

Other tips

- Style doesn’t matter. You don’t need to comment.
  - (but…)

Other tips

- Style doesn’t matter. You don’t need to comment.
  - (but…)
- Don’t worry about imports.

Other tips

- Style doesn’t matter. You don’t need to comment.
  - (but…)
- Don’t worry about imports.
- Pseudocode credit is capped at 50%.

Other tips

- Style doesn’t matter. You don’t need to comment.
  - (but…)
- Don’t worry about imports.
- Pseudocode credit is capped at 50%.
- Edge-cases are really important.

The plan

- General info
- Karel
- Expression evaluation
- Program tracing
- Simple Java and randomness
- Graphics and MouseEvents
- String manipulation

[We will] wield technology's wonders to raise health care's quality and lower its cost.

[We will] wield technology's wonders to raise health care's quality and lower its cost.

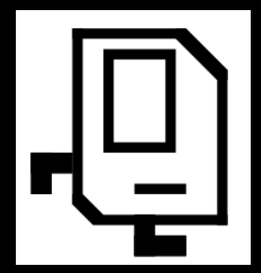
President Barack Obama

[We will] wield technology's wonders to raise health care's quality and lower its cost.

President Barack Obama January 20, 2009

[We will] wield technology's wonders to raise health care's quality and lower its cost.

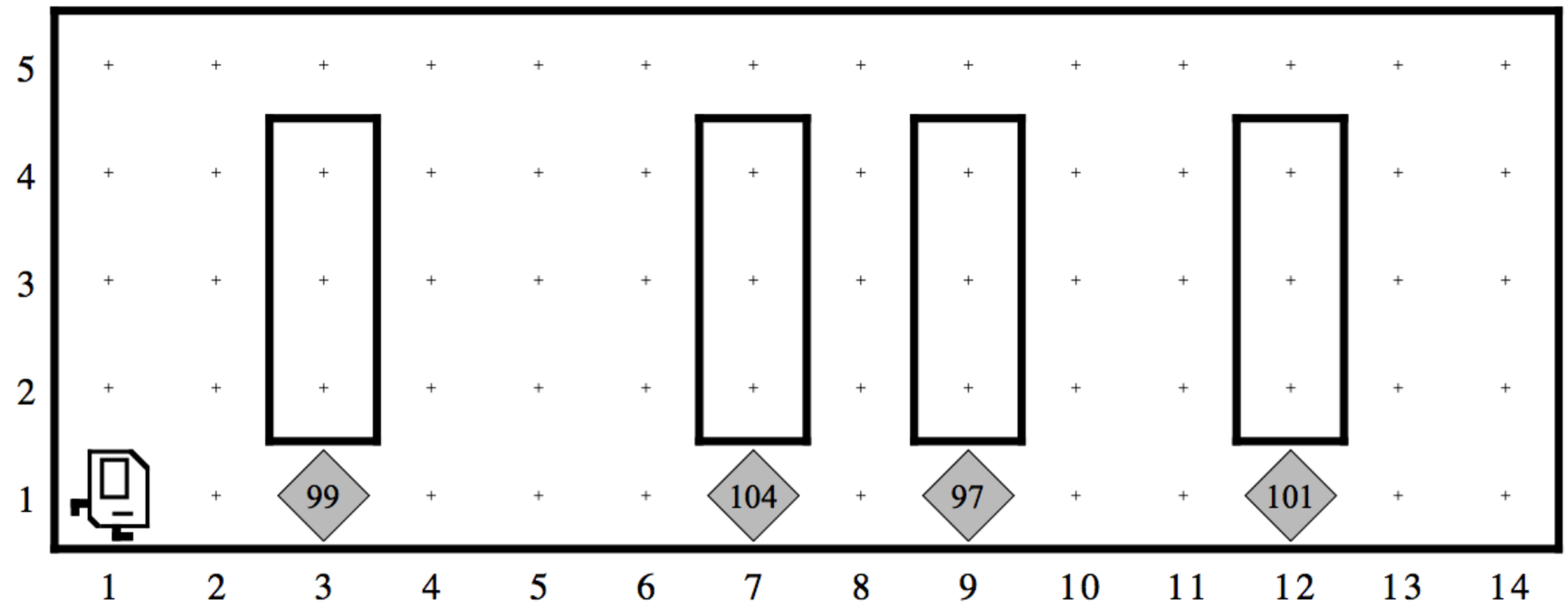
President Barack Obama January 20, 2009



[We will] wield technology's wonders to raise health care's quality and lower its cost.

President Barack Obama January 20, 2009

Given that nothing could possibly be more representative of "technology's wonders" than Karel the Robot, it is clear that a central component of the President's health care plan must involve putting our tireless robot to work in support of patient care. As part of meeting that challenge, the Obama administration is seeking to use Karel to monitor the temperatures of patients on a hospital ward that looks something like this:

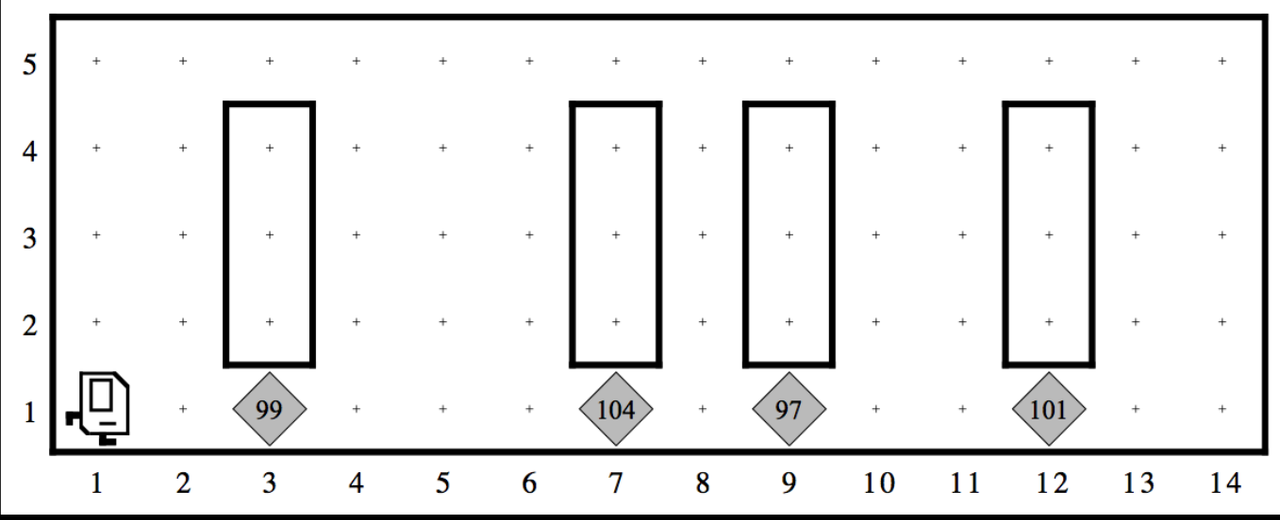


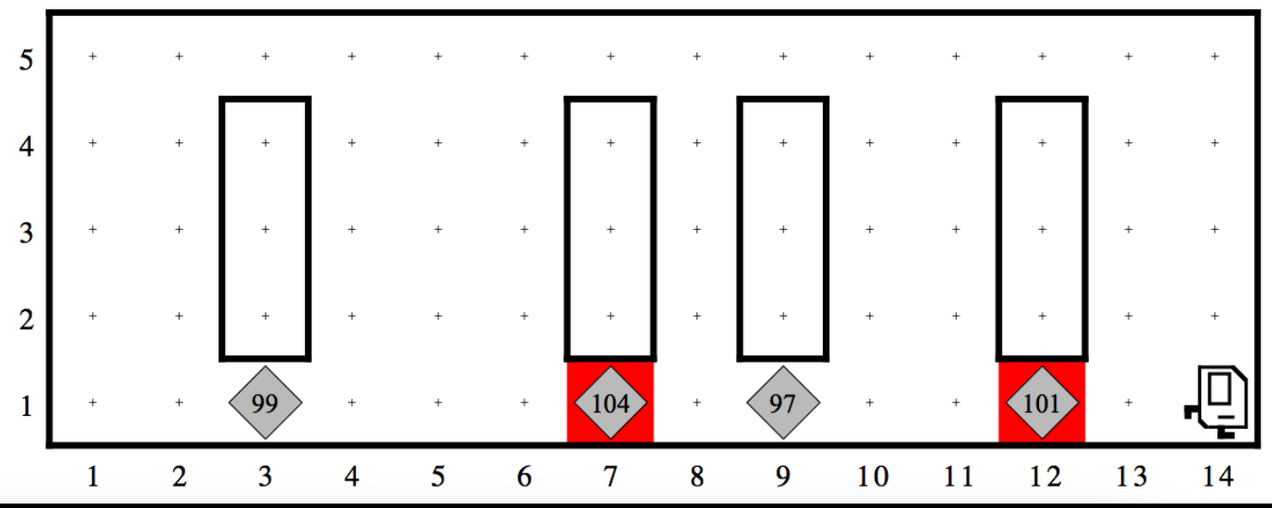
The stacks of beepers under each bed record the current temperature of the patient. Thus, the patient occupying the bed on 3rd Avenue has a temperature of 99°, which is indicated by a stack containing 99 beepers. Although 99° is slightly higher than the normal body temperature of 98.6°F, it is nothing to worry about. The patient in the 7th Avenue bed, on the other hand, has a temperature of 104°, which is dangerously high. The 97° for the patient in the 9th Avenue bed is below normal, but the 101° temperature of the patient in the bed on 12th Avenue is again high enough to cause concern.

Your job in this problem is to implement a program called **KarelCare** in which Karel checks each of the patients in turn and flags any temperature that is greater than 100. One way to flag an out-of-range temperature is to paint the corner red, which **SuperKarel** can do by calling

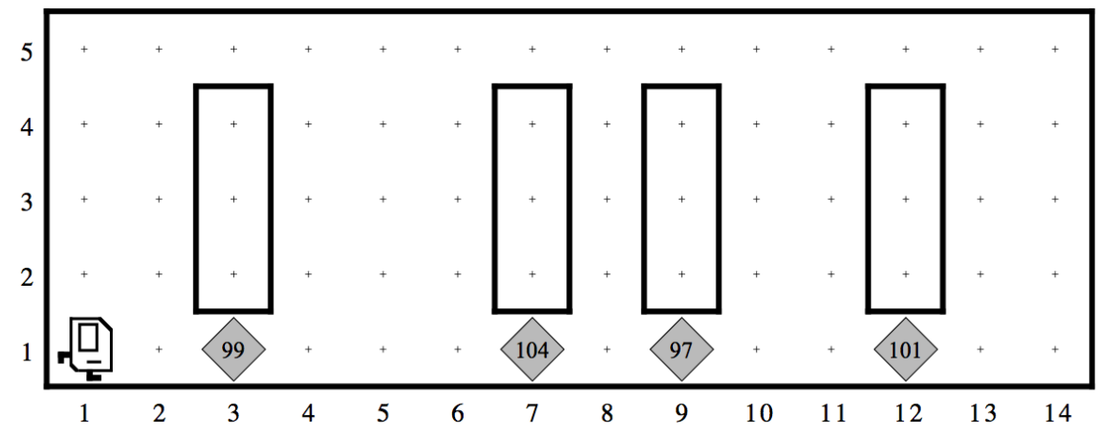
```
paintCorner(RED);
```

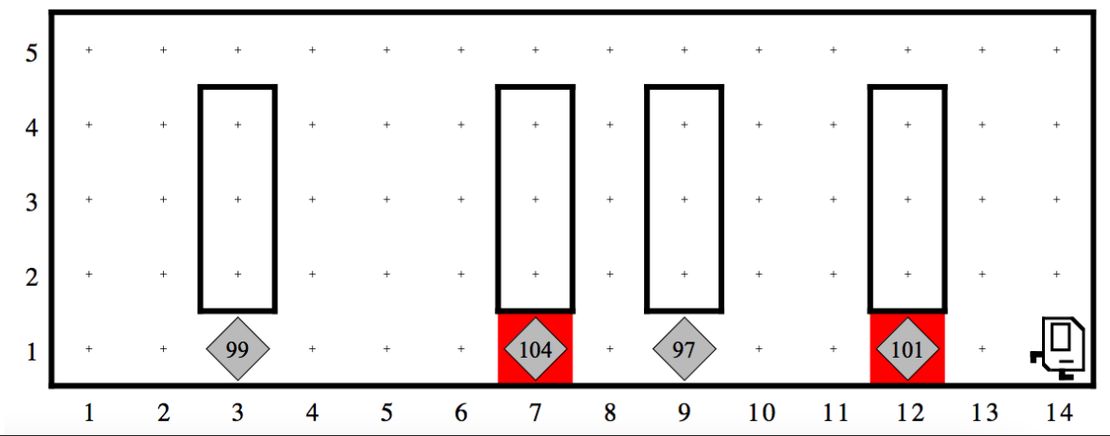
1. The only part of Karel's world you need to consider is 1st Street, which is the row at the bottom of the window (the walls marking the beds are merely decorative). Karel can find where the beds are simply by looking for the stacks of beepers.
2. Karel always begins at the corner of 1st Street and 1st Avenue, facing east, with an empty beeper bag. Karel should finish on the easternmost corner of 1st Street.
3. The world can be of any length, and there can be any number of beds. The beds, moreover, can be separated by any number of open spaces, and can even be adjacent.
4. Once Karel figures out that a particular temperature needs to be flagged, it must put all the beepers back to ensure that the beeper pile continues to show the correct temperature. Given that Karel has no way of telling how many beepers are on a corner without taking them away, the number of beepers in each pile will change as the program runs. (It may help to recall that Karel starts out with an empty beeper bag.)
5. In order to meet the hospital's threshold of concern, a temperature must be strictly greater than 100. Thus, Karel should not paint the square if the patient's temperature is exactly 100, but should do so if the temperature is 101.



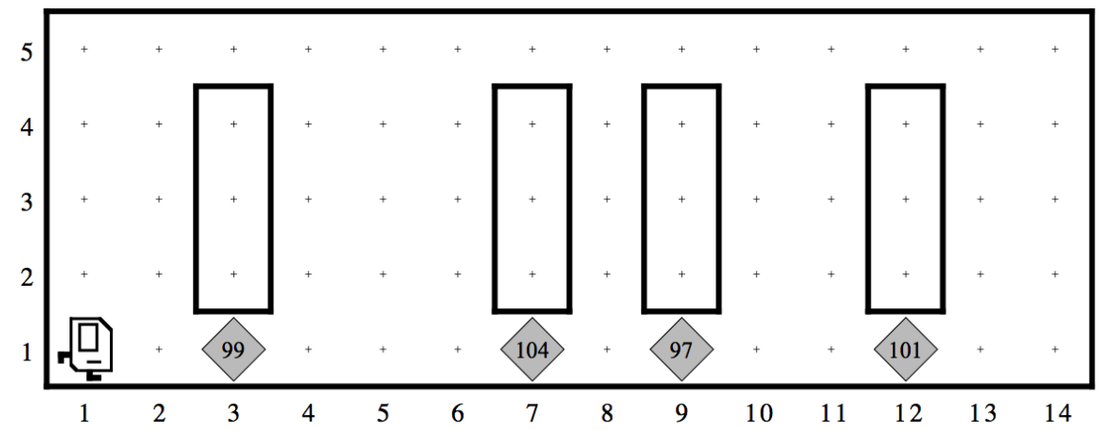


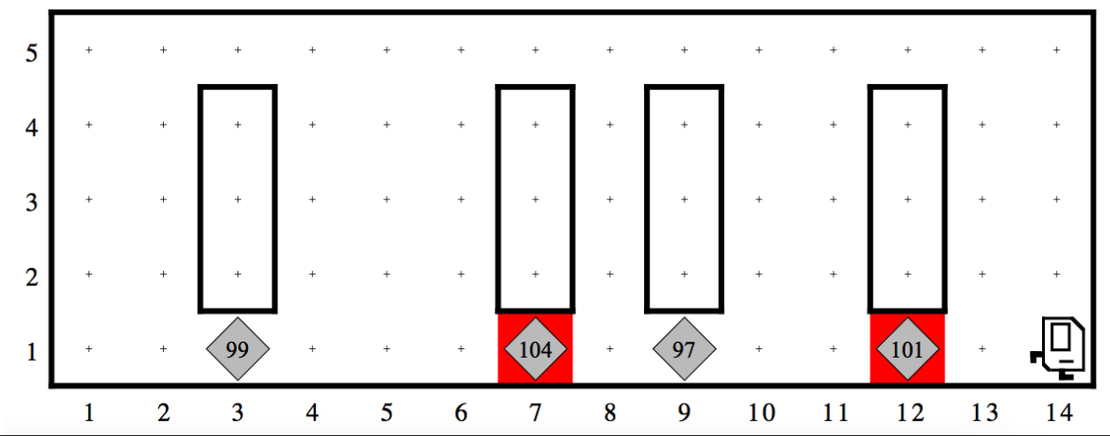
```
public class KarelCare extends SuperKarel {

   public void run() {




   }

}
```



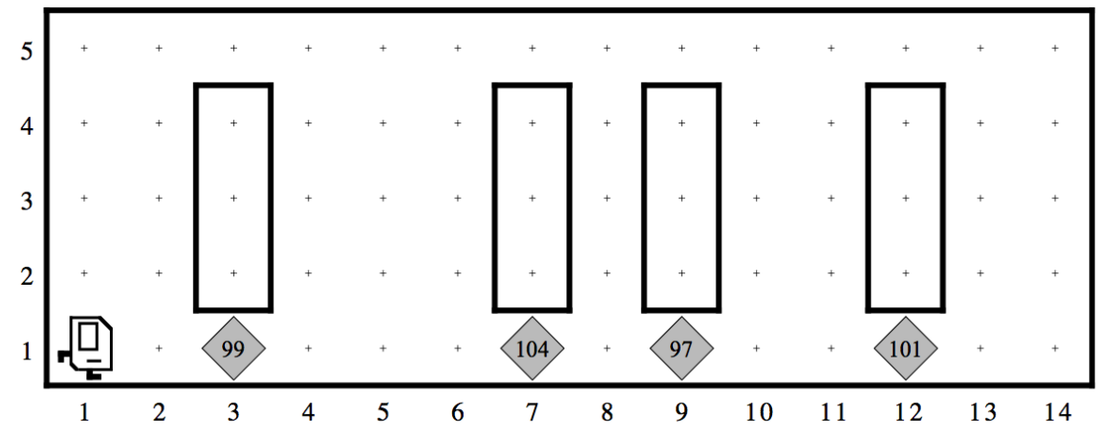


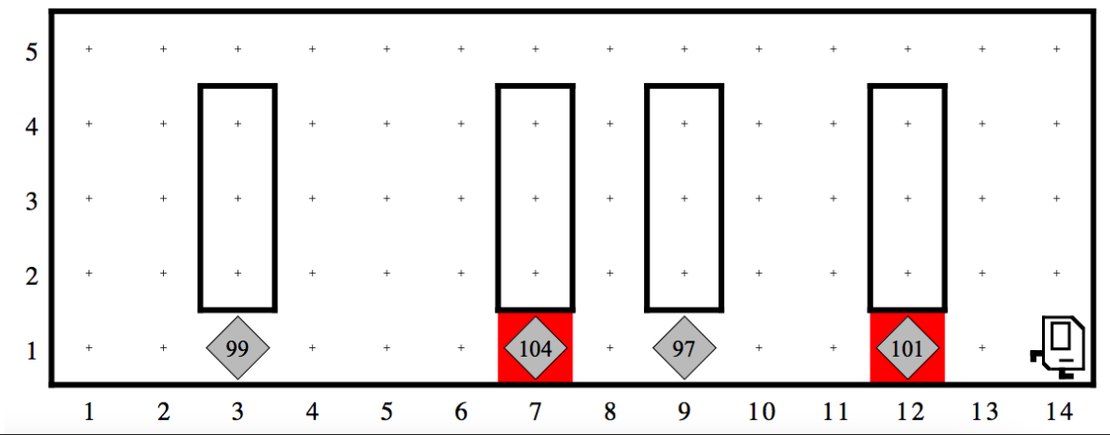
```
public class KarelCare extends SuperKarel {

   public void run() {
      while (frontIsClear()) {



         move();
      }


   }

}
```





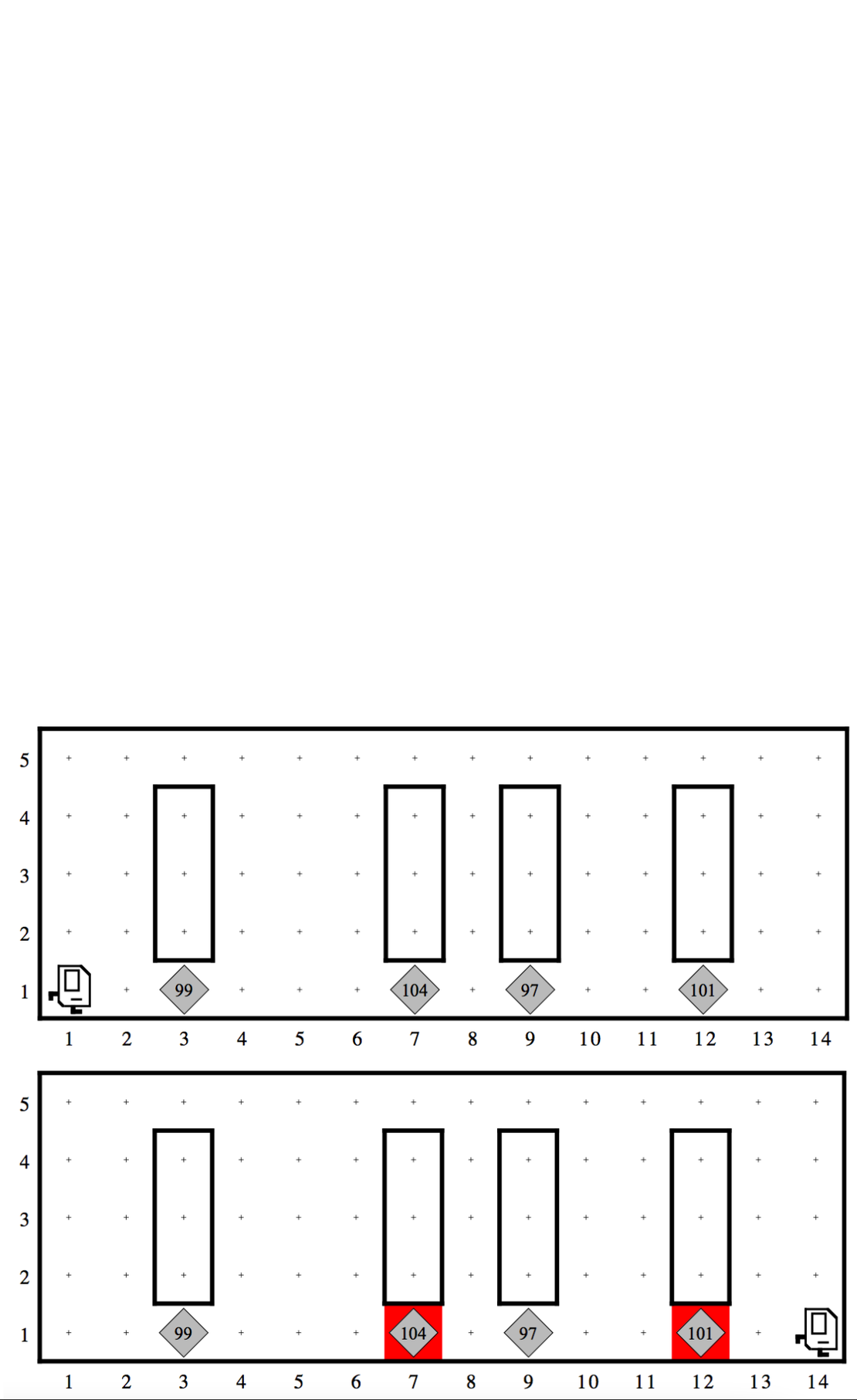
```
public class KarelCare extends SuperKarel {

   public void run() {
      while (frontIsClear()) {
         if (beepersPresent()) {

         }
         move();
      }



   }
```



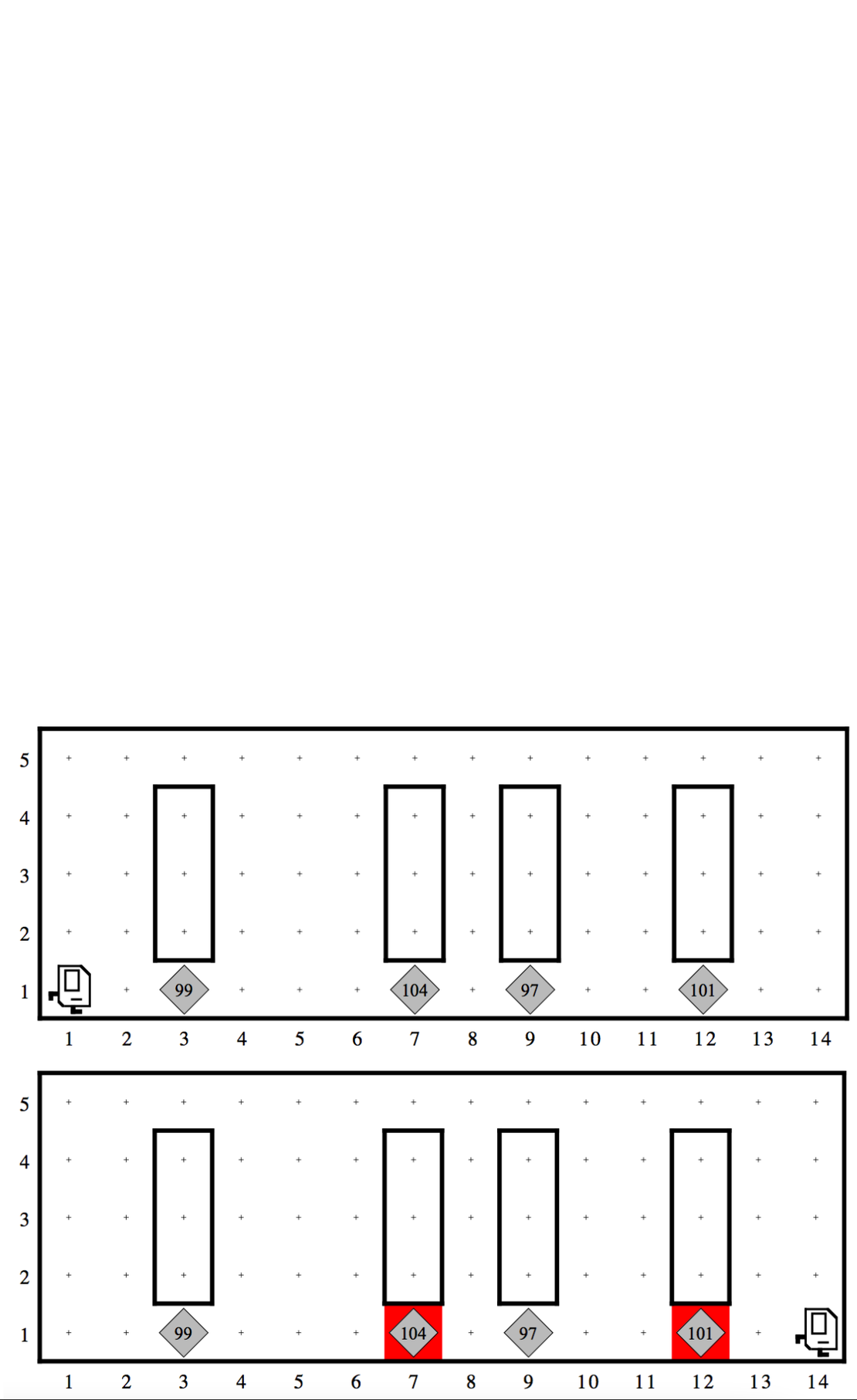



```
}
```

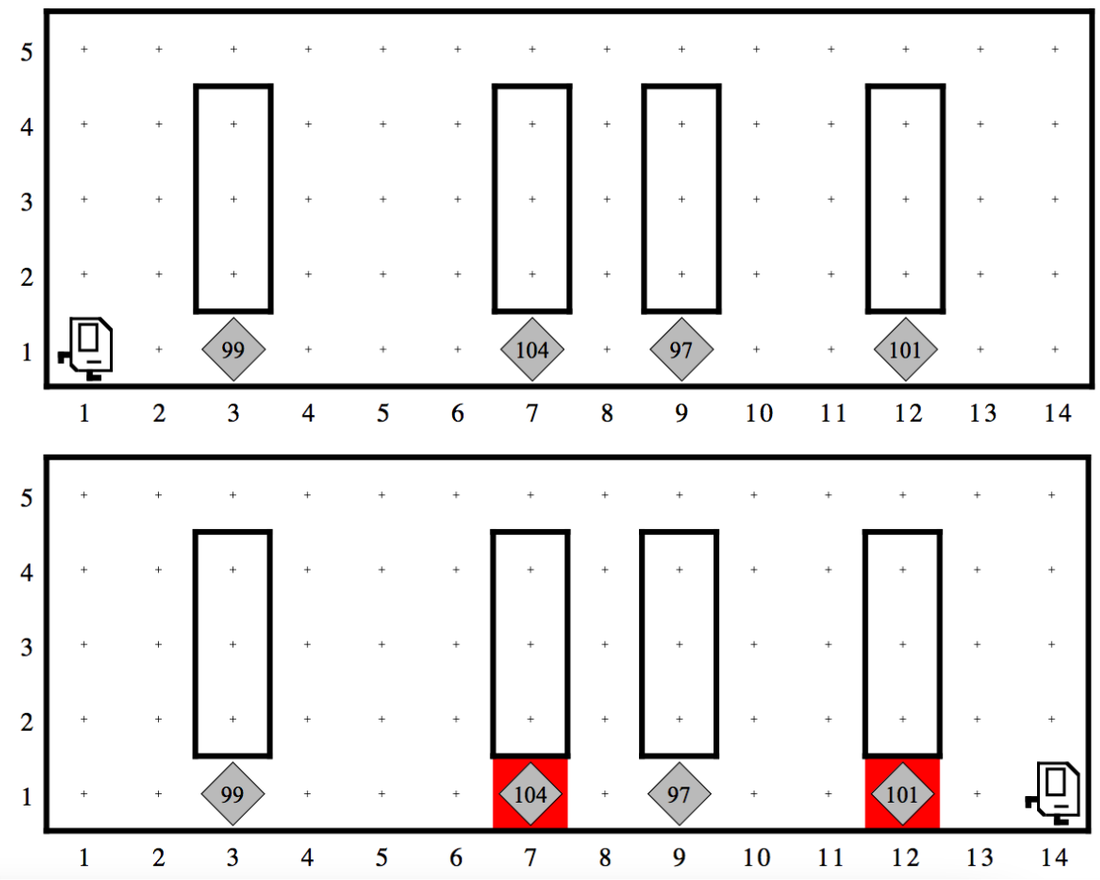
```
public class KarelCare extends SuperKarel {

   public void run() {
      while (frontIsClear()) {
         if (beepersPresent()) {
            checkTemperature();
         }
         move();
      }



   }
   private void checkTemperature() {



   }
}
```



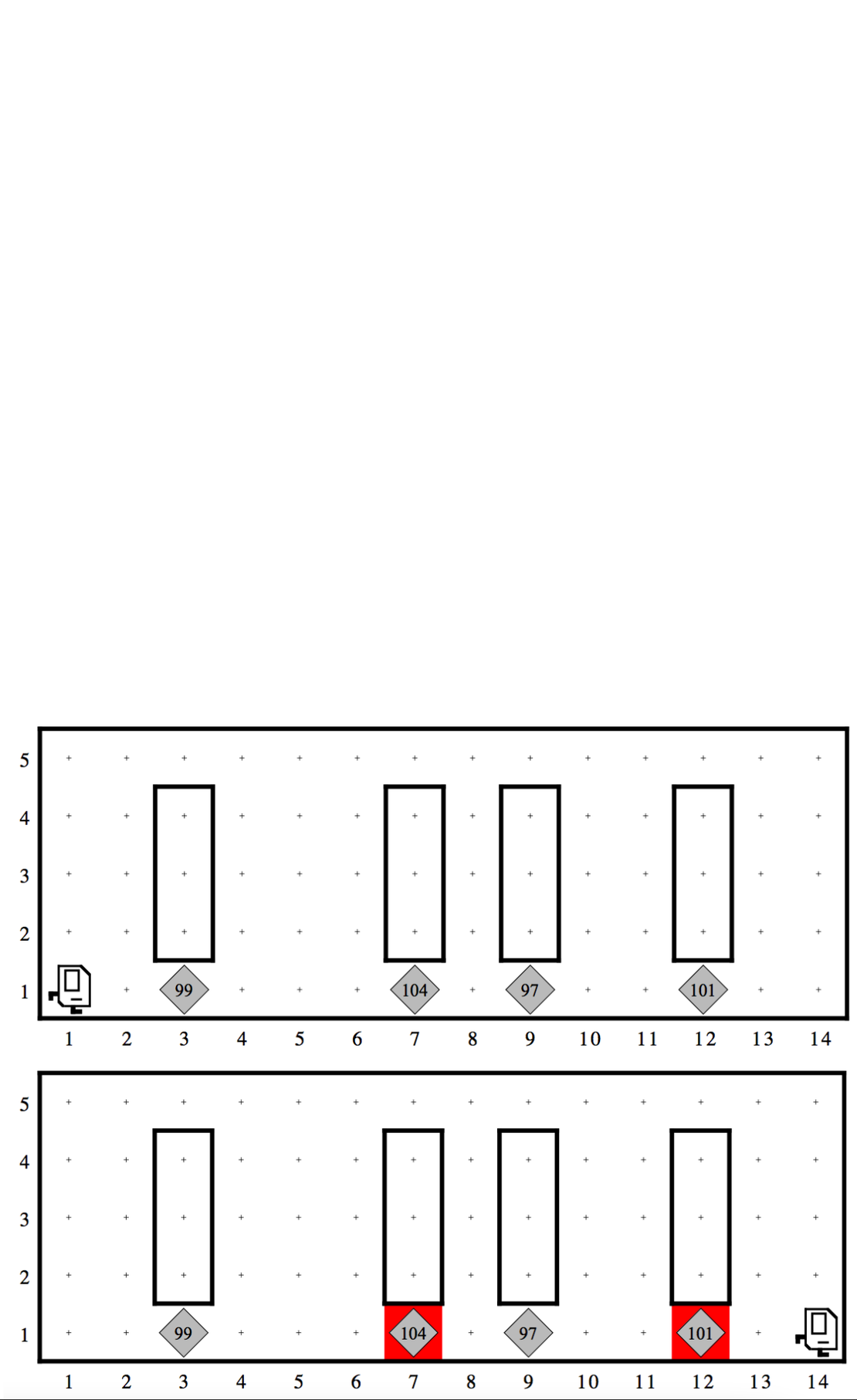
```
public class KarelCare extends SuperKarel {

   public void run() {
      while (frontIsClear()) {
         if (beepersPresent()) {
            checkTemperature();
         }
         move();
      }
      if (beepersPresent()) {
         checkTemperature();
      }
   }
   private void checkTemperature() {

   }
}
```



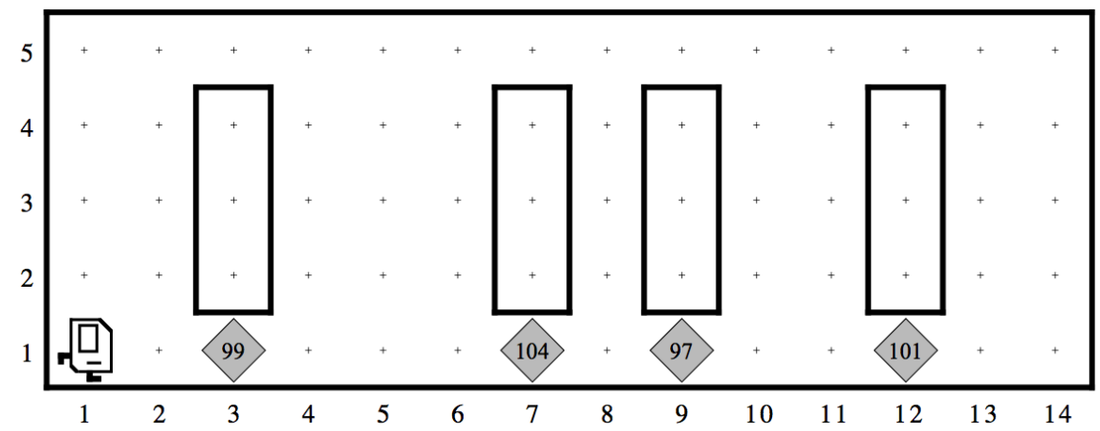
```
public class KarelCare extends SuperKarel {

   public void run() {
      while (frontIsClear()) {
         if (beepersPresent()) {
            checkTemperature();
         }
         move();
      }
      if (beepersPresent()) {
         checkTemperature();
      }
   }
   private void checkTemperature() {

   }
}
```

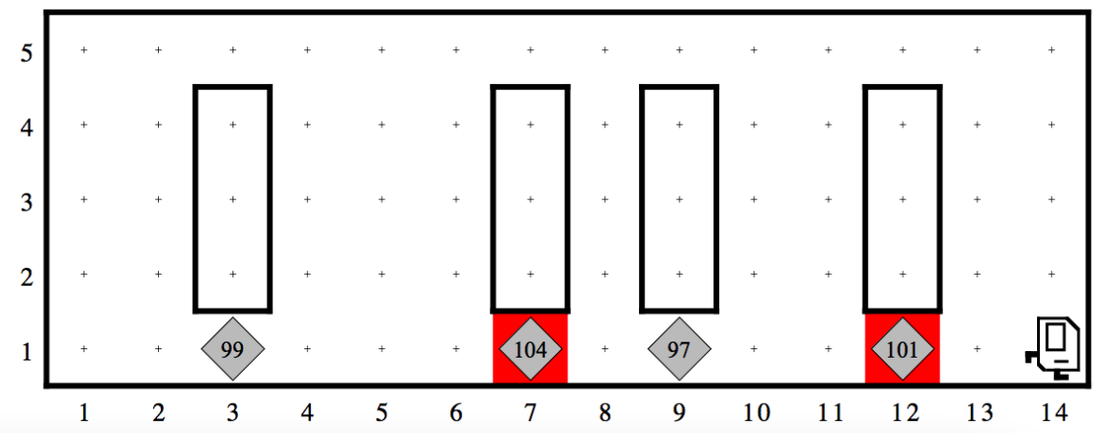


```
public class KarelCare extends SuperKarel {

   public void run() {
      while (frontIsClear()) {
         if (beepersPresent()) {
            checkTemperature();
         }
         move();
      }
      if (beepersPresent()) {
         checkTemperature();
      }
   }
   private void checkTemperature() {

   }
}
```

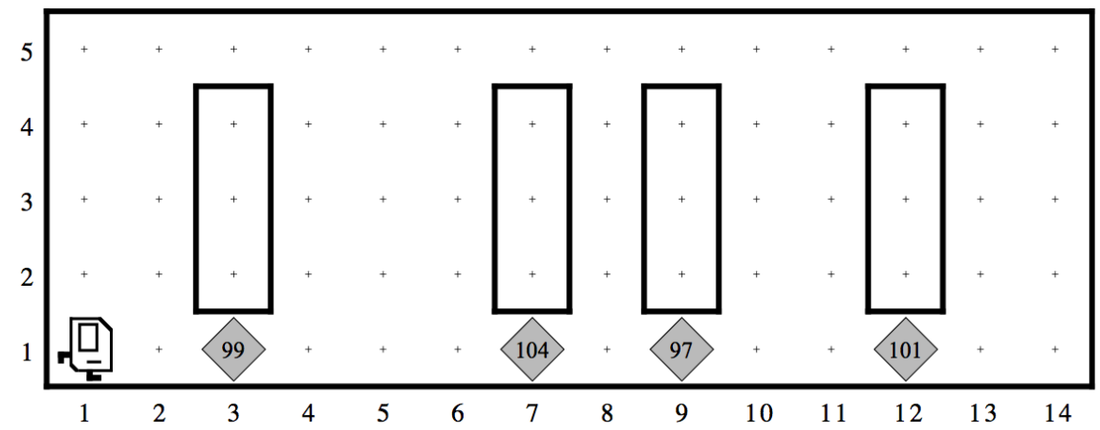


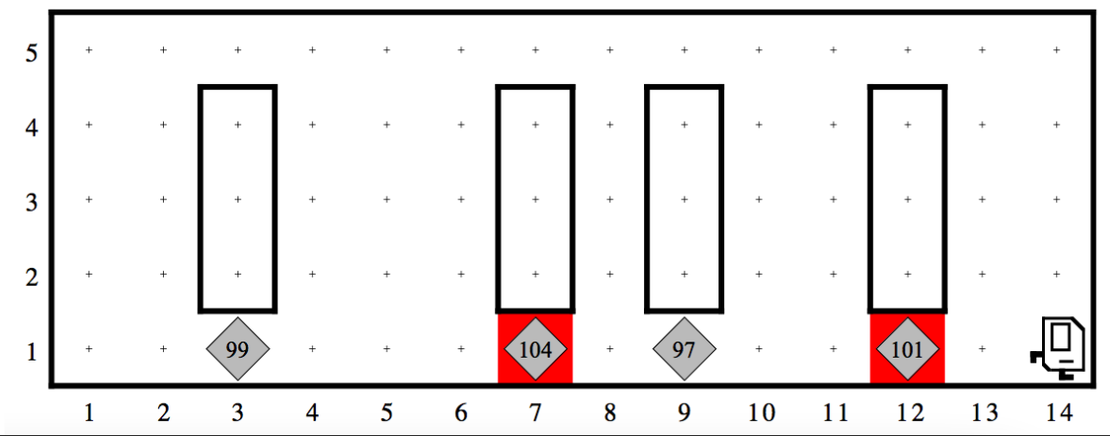
```
public class KarelCare extends SuperKarel {

   public void run() {
      while (frontIsClear()) {
         if (beepersPresent()) {
            checkTemperature();
         }
         move();
      }
      if (beepersPresent()) {
         checkTemperature();
      }
   }
   private void checkTemperature() {
      for (                                ) {




      }




   }

}
```



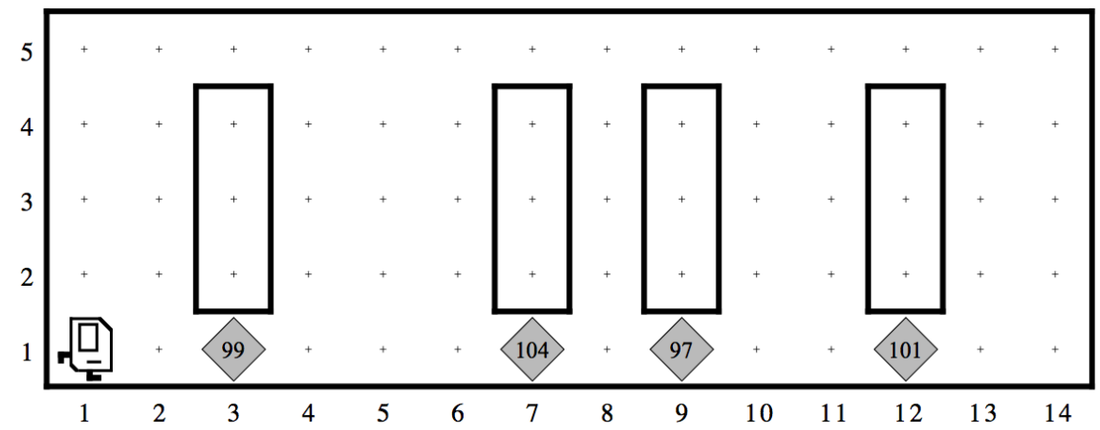


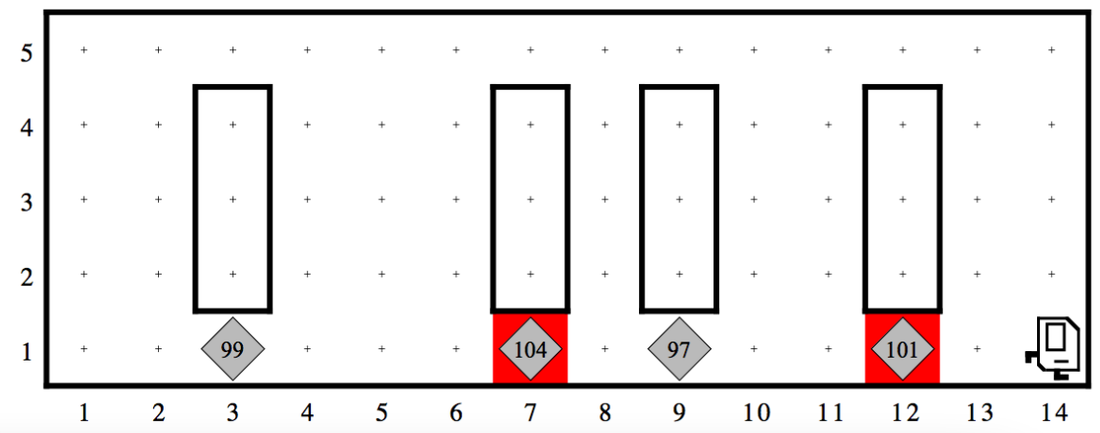
```
public class KarelCare extends SuperKarel {

   public void run() {
      while (frontIsClear()) {
         if (beepersPresent()) {
            checkTemperature();
         }
         move();
      }
      if (beepersPresent()) {
         checkTemperature();
      }
   }
   private void checkTemperature() {
      for (int i = 0; i < 100; i++) {



      }



   }

}
```



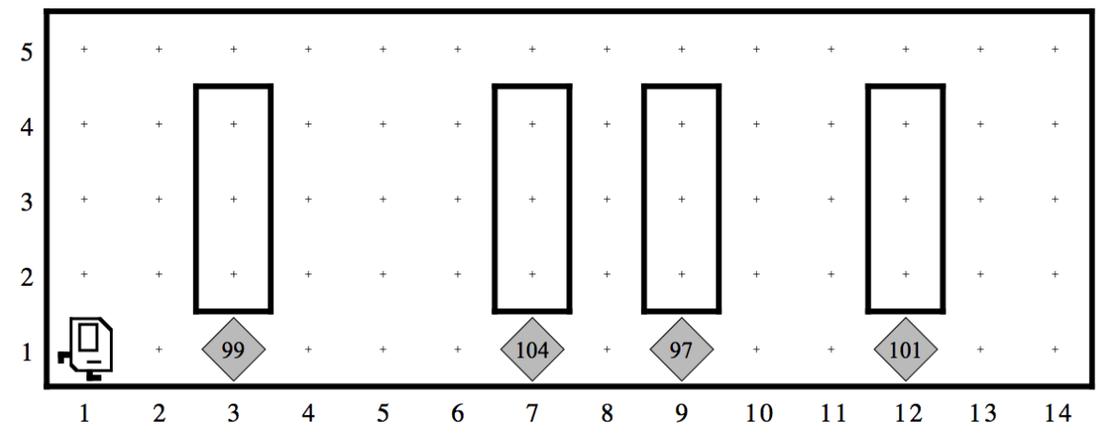


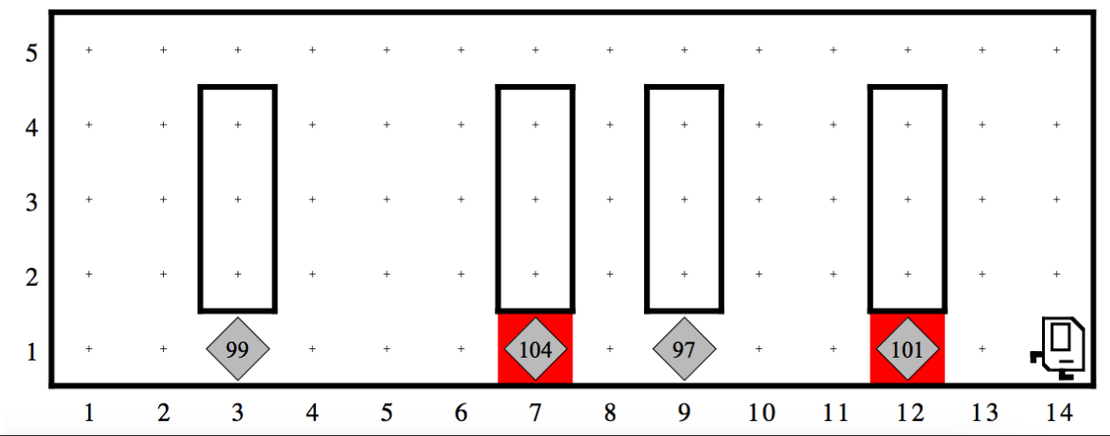
```
public class KarelCare extends SuperKarel {

   public void run() {
      while (frontIsClear()) {
         if (beepersPresent()) {
            checkTemperature();
         }
         move();
      }
      if (beepersPresent()) {
         checkTemperature();
      }
   }
   private void checkTemperature() {
      for (int i = 0; i < 100; i++) {
         if (beepersPresent()) {
            pickBeeper();
         }
      }

   }

}
```



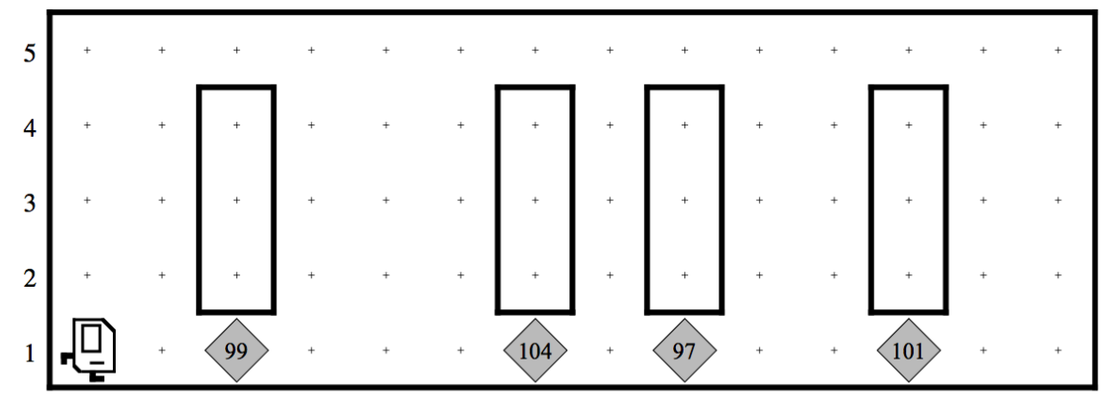


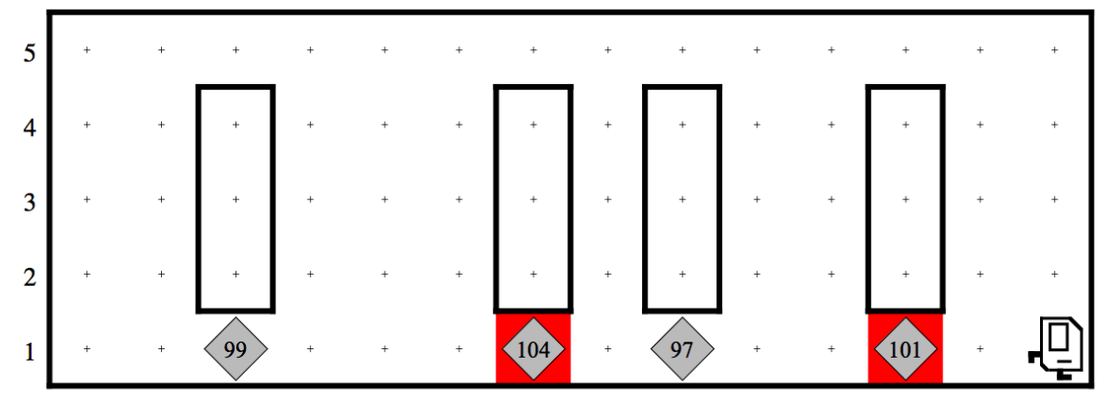
```
public class KarelCare extends SuperKarel {

   public void run() {
      while (frontIsClear()) {
         if (beepersPresent()) {
            checkTemperature();
         }
         move();
      }
      if (beepersPresent()) {
         checkTemperature();
      }
   }
   private void checkTemperature() {
      for (int i = 0; i < 100; i++) {
         if (beepersPresent()) {
            pickBeeper();
         }
      }
      if (beepersPresent()) {

      }



   }

}
```





```
public class KarelCare extends SuperKarel {

   public void run() {
      while (frontIsClear()) {
         if (beepersPresent()) {
            checkTemperature();
         }
         move();
      }
      if (beepersPresent()) {
         checkTemperature();
      }
   }
   private void checkTemperature() {
      for (int i = 0; i < 100; i++) {
         if (beepersPresent()) {
            pickBeeper();
         }
      }
      if (beepersPresent()) {
         paintCorner(RED);
      }



   }

}
```





```
public class KarelCare extends SuperKarel {

   public void run() {
      while (frontIsClear()) {
         if (beepersPresent()) {
            checkTemperature();
         }
         move();
      }
      if (beepersPresent()) {
         checkTemperature();
      }
   }
   private void checkTemperature() {
      for (int i = 0; i < 100; i++) {
         if (beepersPresent()) {
            pickBeeper();
         }
      }
      if (beepersPresent()) {
         paintCorner(RED);
      }
      while (beepersInBag()) {
         putBeeper();
      }
   }

}
```

The plan

- General info
- Karel
- Expression evaluation
- Program tracing
- Simple Java and randomness
- Graphics and MouseEvents
- String manipulation

Problem 2: Simple Java expressions, statements, and methods (20 points)

(2a) Compute the value of each of the following Java expressions. If an error occurs during any of these evaluations, write "Error" on that line and explain briefly why the error occurs.

```
5.0 / 4 - 4 / 5
```

_____

```
7 < 9 - 5 && 3 % 0 == 3
```

_____

```
"B" + 8 + 4
```

_____

Expressions

- Order of operations
- Short circuit evaluation
- Undefined operations
  - divide by zero
  - subtract from a String
  - etc.

Operator Precedence

Operators	Precedence
postfix	*expr*++ *expr*--
unary	++*expr* --*expr* +*expr* -*expr* ~ !
multiplicative	* / %
additive	+ -
shift	<< >> >>>
relational	< > <= >= instanceof
equality	== !=
bitwise AND	&
bitwise exclusive OR	^
bitwise inclusive OR	|
logical AND	&&
logical OR	||
ternary	? :
assignment	= += -= *= /= %= &= ^= |= <<= >>= >>>=

Operators of equal precedence are evaluated left to right

Short circuit evaluation

Short circuit evaluation

- **`x || y`** means **`x OR y`**
  - what if **`x`** is **`true`**?
  - **`y`** doesn't matter!

Short circuit evaluation

- **`x || y`** means **`x OR y`**
  - what if **`x`** is **`true`**?
  - **`y`** doesn't matter!
- **`x && y`** means **`x AND y`**
  - what if **`x`** is **`false`**?
  - **`x`** doesn't matter!

Short circuit evaluation

- **`x || y`** means **`x OR y`**
  - what if **`x`** is **`true`**?
  - **`y`** doesn't matter!
- **`x && y`** means **`x AND y`**
  - what if **`x`** is **`false`**?
  - **`x`** doesn't matter!
- This is important because it can mean error code won't run
  - **`3 == 3 || 4 / 0`**

Expression evaluation

Expression evaluation

- `“F” - “F”`

Expression evaluation

- `“F” - “F”` `Error: Strings don’t subtract`

Expression evaluation

- `"F" - "F"` `Error: Strings don't subtract`
- `1 – 2 – 3 - 4.0`

Expression evaluation

- `"F" - "F"` `Error: Strings don't subtract`
- `1 – 2 – 3 - 4.0` `-8.0`

Expression evaluation

- `"F" - "F"` Error: Strings don't subtract
- `1 — 2 — 3 - 4.0` -8.0
- `1 / (100 / 101)`

Expression evaluation

- `"F" - "F"` Error: Strings don't subtract
- `1 — 2 — 3 - 4.0` -8.0
- `1 / (100 / 101)` Error: Divide by zero

Expression evaluation

- “F” - “F” Error: Strings don’t subtract
- 1 – 2 – 3 - 4.0 -8.0
- 1 / (100 / 101) Error: Divide by zero
- (“1" + 1) + (1 + 1 / 2)

Expression evaluation

- “F” - “F” Error: Strings don’t subtract
- 1 – 2 – 3 - 4.0 -8.0
- 1 / (100 / 101) Error: Divide by zero
- (“1" + 1) + (1 + 1 / 2) "111"

Expression evaluation

- `"F" - "F"` Error: Strings don't subtract
- `1 — 2 — 3 - 4.0` -8.0
- `1 / (100 / 101)` Error: Divide by zero
- `("1" + 1) + (1 + 1 / 2)` "111"
- `("1" + 1) + (1 + 1 / 2.0)`

Expression evaluation

- “F” - “F” Error: Strings don’t subtract
- 1 – 2 – 3 - 4.0 -8.0
- 1 / (100 / 101) Error: Divide by zero
- (“1" + 1) + (1 + 1 / 2) "111"
- (“1" + 1) + (1 + 1 / 2.0) “111.5"

More expressions

More expressions

- `(char)('Z'- 10 / 2)`

More expressions

- `(char)('Z'- 10 / 2)`

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

More expressions

- `(char)('Z'- 10 / 2)` `'U'`

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

More expressions

- `(char)('Z'- 10 / 2)` `'U'`
- `7 < 9 - 5 && 3 % 0 == 3`

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

More expressions

- `(char)('Z'- 10 / 2)` `'U'`
- `7 < 9 - 5 && 3 % 0 == 3` `false`

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

More expressions

- `(char)('Z'- 10 / 2)` `'U'`
- `7 < 9 - 5 && 3 % 0 == 3` `false`
- `1 + 2 + "1 + 2" + 1 + 2`

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

More expressions

- `(char)('Z'- 10 / 2)` `'U'`
- `7 < 9 - 5 && 3 % 0 == 3` `false`
- `1 + 2 + "1 + 2" + 1 + 2`

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

More expressions

- `(char)('Z'- 10 / 2)` `'U'`
- `7 < 9 - 5 && 3 % 0 == 3` `false`
- `1 + 2 + "1 + 2" + 1 + 2`

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

More expressions

- `(char)('Z'- 10 / 2)` `'U'`
- `7 < 9 - 5 && 3 % 0 == 3` `false`
- `1 + 2 + "1 + 2" + 1 + 2`

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

More expressions

- `(char)('Z'- 10 / 2)` `'U'`
- `7 < 9 - 5 && 3 % 0 == 3` `false`
- `1 + 2 + "1 + 2" + 1 + 2`

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

More expressions

- `(char)('Z'- 10 / 2)` `'U'`
- `7 < 9 - 5 && 3 % 0 == 3` `false`
- `1 + 2 + "1 + 2" + 1 + 2` `"31 + 212"`

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

More expressions

- `(char)('Z'- 10 / 2)` `'U'`
- `7 < 9 - 5 && 3 % 0 == 3` `false`
- `1 + 2 + "1 + 2" + 1 + 2` `"31 + 212"`
- `5.0 / 4 - 4 / 5`

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

More expressions

- `(char)('Z'- 10 / 2)` `'U'`
- `7 < 9 - 5 && 3 % 0 == 3` `false`
- `1 + 2 + "1 + 2" + 1 + 2` `"31 + 212"`
- `5.0 / 4 - 4 / 5` `1.25`

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

The plan

- General info
- Karel
- Expression evaluation
- Program tracing
- Simple Java and randomness
- Graphics and MouseEvents
- String manipulation

(2b) What output is printed by the following program:

```
/*
 * File: Problem2b.java
 * --------------------
 * This program doesn't do anything useful and exists only to test
 * your understanding of method calls and parameter passing.
 */

import acm.program.*;

public class Problem2b extends ConsoleProgram {

  public void run() {
     int num1 = 2;
     int num2 = 13;
     println("The 1st number is: " + Mystery(num1, 6));
     println("The 2nd number is: " + Mystery(num2 % 5, 1 + num1 * 2));
  }


  private int Mystery(int num1, int num2) {
     num1 = Unknown(num1, num2);
     num2 = Unknown(num2, num1);
     return(num2);
  }


  private int Unknown(int num1, int num2) {
     int num3 = num1 + num2;
     num2 += num3 * 2;
     return(num2);
  }

}
```

Answer:

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
	public void run() {
		int tinker  = 36;
		int tailor  = 54;
		int soldier = smiley(tailor, tinker);
		println("soldier = " + soldier);
	}

	private int smiley(int tinker, int tailor) {
		int poorMan = guillam(tinker, tailor);
		println("poorMan = " + poorMan);

		int beggarMan = guillam(tailor + 9, tinker / 9);
		println("beggarMan = " + beggarMan);

		return poorMan + beggarMan;
	}

	private int guillam(int karla, int mundt) {
		karla %= 10;
		mundt /= 10;
		return 100 * karla + mundt;
	}
}
```

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

smiley

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

smiley

guillam

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

smiley

guillam

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker:
tailor:
soldier:

smiley

guillam

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker:
tailor:
soldier:

smiley

guillam

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor:
soldier:

smiley

guillam

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor:
soldier:

smiley

guillam

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

guillam

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

guillam

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

guillam

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker:
tailor:
poorMan:
beggarMan:

guillam

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor:
poorMan:
beggarMan:

guillam

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan:
beggarMan:

guillam

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan:
beggarMan:

guillam

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan:
beggarMan:

guillam

karla:
mundt:

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan:
beggarMan:

guillam

karla: **54**
mundt:

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan:
beggarMan:

guillam

karla: **54**
mundt: **36**

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan:
beggarMan:

guillam

karla: **4**
mundt: **36**

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan:
beggarMan:

guillam

karla: **4**
mundt: **36**

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan:
beggarMan:

guillam

karla: **4**
mundt: **3**

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan:
beggarMan:

guillam

karla: **4**
mundt: **3**

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan: **403**
beggarMan:

guillam

karla: **4**
mundt: **3**

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan: **403**
beggarMan:

guillam

karla: **4**
mundt: **3**

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan: **403**
beggarMan:

guillam

karla:
mundt:

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan: **403**
beggarMan:

guillam

karla:
mundt:

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan: **403**
beggarMan:

guillam

karla:
mundt:

poorMan = 403

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan: **403**
beggarMan:

guillam

karla:
mundt:

poorMan = 403

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan: **403**
beggarMan:

guillam

karla:
mundt:

poorMan = 403

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan: **403**
beggarMan:

guillam

karla: **45**
mundt:

poorMan = 403

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan: **403**
beggarMan:

guillam

karla: **45**
mundt: **6**

poorMan = 403

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan: **403**
beggarMan:

guillam

karla: **5**
mundt: **6**

poorMan = 403

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan: **403**
beggarMan:

guillam

karla: **5**
mundt: **6**

poorMan = 403

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan: **403**
beggarMan:

guillam

karla: **5**
mundt: **0**

poorMan = 403

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan: **403**
beggarMan:

guillam

karla: **5**
mundt: **0**

poorMan = 403

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan: **403**
beggarMan: **500**

guillam

karla: **5**
mundt: **0**

poorMan = 403

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan: **403**
beggarMan: **500**

guillam

karla: **5**
mundt: **0**

poorMan = 403

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan: **403**
beggarMan: **500**

guillam

karla:
mundt:

poorMan = 403

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan: **403**
beggarMan: **500**

guillam

karla:
mundt:

poorMan = 403
beggarMan = 500

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan: **403**
beggarMan: **500**

guillam

karla:
mundt:

poorMan = 403
beggarMan = 500

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier: **903**

smiley

tinker: **54**
tailor: **36**
poorMan: **403**
beggarMan: **500**

guillam

karla:
mundt:

poorMan = 403
beggarMan = 500

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier: **903**

smiley

tinker: **54**
tailor: **36**
poorMan: **403**
beggarMan: **500**

guillam

karla:
mundt:

poorMan = 403
beggarMan = 500

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier: **903**

smiley

tinker:
tailor:
poorMan:
beggarMan:

guillam

karla:
mundt:

poorMan = 403
beggarMan = 500

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier: **903**

smiley

tinker:
tailor:
poorMan:
beggarMan:

guillam

karla:
mundt:

poorMan = 403
beggarMan = 500

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier: **903**

smiley

tinker:
tailor:
poorMan:
beggarMan:

guillam

karla:
mundt:

poorMan = 403
beggarMan = 500
soldier = 903

```
/*
 * File: Trace.java
 * ________________
 * This program doesn't do anything useful
 * and exists only to test understanding
 * of parameters and String methods.
 */
import acm.program.*;

public class Problem2c extends ConsoleProgram {

   public void run() {
      String s1 = "To err";
      String s2 = "is human!";
      s1 = forgive(s1, s2);
      println(s1 + " " + s2);
   }

   private String forgive(String me, String you) {
      String heart = me.substring(0, you.length() - me.length());
      you = "" + you.charAt(me.length());
      int amount = heart.length();
      me = me.substring(amount + 2) + me.charAt(amount);
      heart += understanding(you, 2) + you + me;
      return heart;
   }

   private char understanding(String you, int num) {
      return (char)(you.charAt(0) + num);
   }

}
```

```
/*
 * File: Trace.java
 * ________________
 * This program doesn't do anything useful
 * and exists only to test understanding
 * of parameters and String methods.
 */
import acm.program.*;

public class Problem2c extends ConsoleProgram {

   public void run() {
      String s1 = "To err";
      String s2 = "is human!";
      s1 = forgive(s1, s2);
      println(s1 + " " + s2);
   }

   private String forgive(String me, String you) {
      String heart = me.substring(0, you.length() - me.length());
      you = "" + you.charAt(me.length());
      int amount = heart.length();
      me = me.substring(amount + 2) + me.charAt(amount);
      heart += understanding(you, 2) + you + me;
      return heart;
   }

   private char understanding(String you, int num) {
      return (char)(you.charAt(0) + num);
   }

}
```

RUN

s1

s2

```
/*
 * File: Trace.jav
 * ---------------
 * This program do
 * and exists only
 * of parameters a
 */
import acm.program.*;

public class Problem2c extends ConsoleProgram {

   public void run() {
      String s1 = "To err";
      String s2 = "is human!";
      s1 = forgive(s1, s2);
      println(s1 + " " + s2);
   }

   private String forgive(String me, String you) {
      String heart = me.substring(0, you.length() - me.length());
      you = "" + you.charAt(me.length());
      int amount = heart.length();
      me = me.substring(amount + 2) + me.charAt(amount);
      heart += understanding(you, 2) + you + me;
      return heart;
   }

   private char understanding(String you, int num) {
      return (char)(you.charAt(0) + num);
   }

}
```

RUN

s1 “To err”

s2

```
/*
 * File: Trace.jav
 * ----------------
 * This program do
 * and exists only
 * of parameters a
 */
import acm.program.*;

public class Problem2c extends ConsoleProgram {

   public void run() {
      String s1 = "To err";
      String s2 = "is human!";
      s1 = forgive(s1, s2);
      println(s1 + " " + s2);
   }

   private String forgive(String me, String you) {
      String heart = me.substring(0, you.length() - me.length());
      you = "" + you.charAt(me.length());
      int amount = heart.length();
      me = me.substring(amount + 2) + me.charAt(amount);
      heart += understanding(you, 2) + you + me;
      return heart;
   }

   private char understanding(String you, int num) {
      return (char)(you.charAt(0) + num);
   }

}
```

RUN

s1 “To err”

s2

```
/*
 * File: Trace.jav
 * ----------------
 * This program do
 * and exists only
 * of parameters a
 */
import acm.program.*;

public class Problem2c extends ConsoleProgram {

   public void run() {
      String s1 = "To err";
      String s2 = "is human!";
      s1 = forgive(s1, s2);
      println(s1 + " " + s2);
   }

   private String forgive(String me, String you) {
      String heart = me.substring(0, you.length() - me.length());
      you = "" + you.charAt(me.length());
      int amount = heart.length();
      me = me.substring(amount + 2) + me.charAt(amount);
      heart += understanding(you, 2) + you + me;
      return heart;
   }

   private char understanding(String you, int num) {
      return (char)(you.charAt(0) + num);
   }

}
```

RUN

s1 “To err”

s2 “is human!”

```
/*
 * File: Trace.jav
 * ----------------
 * This program do
 * and exists only
 * of parameters a
 */
import acm.program.*;

public class Problem2c extends ConsoleProgram {

   public void run() {
      String s1 = "To err";
      String s2 = "is human!";
      s1 = forgive(s1, s2);
      println(s1 + " " + s2);
   }

   private String forgive(String me, String you) {
      String heart = me.substring(0, you.length() - me.length());
      you = "" + you.charAt(me.length());
      int amount = heart.length();
      me = me.substring(amount + 2) + me.charAt(amount);
      heart += understanding(you, 2) + you + me;
      return heart;
   }

   private char understanding(String you, int num) {
      return (char)(you.charAt(0) + num);
   }

}
```

RUN

s1 “To err”

s2 “is human!”

```
/*
 * File: Trace.jav
 * ----------------
 * This program do
 * and exists only
 * of parameters a
 */
import acm.program.*;

public class Problem2c extends ConsoleProgram {

   public void run() {
      String s1 = "To err";
      String s2 = "is human!";
      s1 = forgive(s1, s2);
      println(s1 + " " + s2);
   }

   private String forgive(String me, String you) {
      String heart = me.substring(0, you.length() - me.length());
      you = "" + you.charAt(me.length());
      int amount = heart.length();
      me = me.substring(amount + 2) + me.charAt(amount);
      heart += understanding(you, 2) + you + me;
      return heart;
   }

   private char understanding(String you, int num) {
      return (char)(you.charAt(0) + num);
   }

}
```

RUN

s1 “To err”

s2 “is human!”

```
/*
 * File: Trace.jav
 * ----------------
 * This program do
 * and exists only
 * of parameters a
 */
import acm.program.*;

public class Problem2c extends ConsoleProgram {

   public void run() {
      String s1 = "To err";
      String s2 = "is human!";
      s1 = forgive(s1, s2);
      println(s1 + " " + s2);
   }

   private String forgive(String me, String you) {
      String heart = me.substring(0, you.length() - me.length());
      you = "" + you.charAt(me.length());
      int amount = heart.length();
      me = me.substring(amount + 2) + me.charAt(amount);
      heart += understanding(you, 2) + you + me;
      return heart;
   }

   private char understanding(String you, int num) {
      return (char)(you.charAt(0) + num);
   }

}
```

```
/*
 * File: Trace.jav
 * ----------------
 * This program do
 * and exists only
 * of parameters a
 */
import acm.program.*;

public class Problem2c extends ConsoleProgram {

   public void run() {
      String s1 = "To err";
      String s2 = "is human!";
      s1 = forgive(s1, s2);
      println(s1 + " " + s2);
   }

   private String forgive(String me, String you) {
      String heart = me.substring(0, you.length() - me.length());
      you = "" + you.charAt(me.length());
      int amount = heart.length();
      me = me.substring(amount + 2) + me.charAt(amount);
      heart += understanding(you, 2) + you + me;
      return heart;
   }

   private char understanding(String you, int num) {
      return (char)(you.charAt(0) + num);
   }

}
```

RUN

s1 “To err”

s2 “is human!”

FORGIVE

me

you

heart

amount

```
/*
 * File: Trace.jav
 * ----------------
 * This program do
 * and exists only
 * of parameters a
 */
import acm.program.*;

public class Problem2c extends ConsoleProgram {

   public void run() {
      String s1 = "To err";
      String s2 = "is human!";
      s1 = forgive(s1, s2);
      println(s1 + " " + s2);
   }

   private String forgive(String me, String you) {
      String heart = me.substring(0, you.length() - me.length());
      you = "" + you.charAt(me.length());
      int amount = heart.length();
      me = me.substring(amount + 2) + me.charAt(amount);
      heart += understanding(you, 2) + you + me;
      return heart;
   }

   private char understanding(String you, int num) {
      return (char)(you.charAt(0) + num);
   }

}
```

RUN

s1 “To err”

s2 “is human!”

FORGIVE

me “To err”

you

heart

amount

```
/*
 * File: Trace.jav
 * ----------------
 * This program do
 * and exists only
 * of parameters a
 */
import acm.program.*;

public class Problem2c extends ConsoleProgram {

   public void run() {
      String s1 = "To err";
      String s2 = "is human!";
      s1 = forgive(s1, s2);
      println(s1 + " " + s2);
   }

   private String forgive(String me, String you) {
      String heart = me.substring(0, you.length() - me.length());
      you = "" + you.charAt(me.length());
      int amount = heart.length();
      me = me.substring(amount + 2) + me.charAt(amount);
      heart += understanding(you, 2) + you + me;
      return heart;
   }

   private char understanding(String you, int num) {
      return (char)(you.charAt(0) + num);
   }

}
```

RUN

s1 “To err”

s2 “is human!”

FORGIVE

me “To err”

you “is human!”

heart

amount

```
/*
 * File: Trace.jav
 * ----------------
 * This program do
 * and exists only
 * of parameters a
 */
import acm.program.*;

public class Problem2c extends ConsoleProgram {

   public void run() {
      String s1 = "To err";
      String s2 = "is human!";
      s1 = forgive(s1, s2);
      println(s1 + " " + s2);
   }

   private String forgive(String me, String you) {
      String heart = me.substring(0, you.length() - me.length());
      you = "" + you.charAt(me.length());
      int amount = heart.length();
      me = me.substring(amount + 2) + me.charAt(amount);
      heart += understanding(you, 2) + you + me;
      return heart;
   }

   private char understanding(String you, int num) {
      return (char)(you.charAt(0) + num);
   }

}
```

RUN

s1 “To err”

s2 “is human!”

FORGIVE

me	“To err”
you	“is human!”
heart	“To ”
amount	

```
/*
 * File: Trace.jav
 * ----------------
 * This program do
 * and exists only
 * of parameters a
 */
import acm.program.*;

public class Problem2c extends ConsoleProgram {

   public void run() {
      String s1 = "To err";
      String s2 = "is human!";
      s1 = forgive(s1, s2);
      println(s1 + " " + s2);
   }

   private String forgive(String me, String you) {
      String heart = me.substring(0, you.length() - me.length());
      you = "" + you.charAt(me.length());
      int amount = heart.length();
      me = me.substring(amount + 2) + me.charAt(amount);
      heart += understanding(you, 2) + you + me;
      return heart;
   }

   private char understanding(String you, int num) {
      return (char)(you.charAt(0) + num);
   }

}
```

RUN

s1	"To err"
s2	"is human!"

FORGIVE

me	"To err"
you	"is human!"
heart	"To "
amount	

```
/*
 * File: Trace.jav
 * ----------------
 * This program do
 * and exists only
 * of parameters a
 */
import acm.program.*;

public class Problem2c extends ConsoleProgram {

   public void run() {
      String s1 = "To err";
      String s2 = "is human!";
      s1 = forgive(s1, s2);
      println(s1 + " " + s2);
   }

   private String forgive(String me, String you) {
      String heart = me.substring(0, you.length() - me.length());
      you = "" + you.charAt(me.length());
      int amount = heart.length();
      me = me.substring(amount + 2) + me.charAt(amount);
      heart += understanding(you, 2) + you + me;
      return heart;
   }

   private char understanding(String you, int num) {
      return (char)(you.charAt(0) + num);
   }

}
```

RUN

s1 “To err”

s2 “is human!”

FORGIVE

me “To err”

you “a”

heart “To ”

amount

```
/*
 * File: Trace.jav
 * ----------------
 * This program do
 * and exists only
 * of parameters a
 */
import acm.program.*;

public class Problem2c extends ConsoleProgram {

   public void run() {
      String s1 = "To err";
      String s2 = "is human!";
      s1 = forgive(s1, s2);
      println(s1 + " " + s2);
   }

   private String forgive(String me, String you) {
      String heart = me.substring(0, you.length() - me.length());
      you = "" + you.charAt(me.length());
      int amount = heart.length();
      me = me.substring(amount + 2) + me.charAt(amount);
      heart += understanding(you, 2) + you + me;
      return heart;
   }

   private char understanding(String you, int num) {
      return (char)(you.charAt(0) + num);
   }

}
```

RUN

s1 “To err”

s2 “is human!”

FORGIVE

me “To err”

you “a”

heart “To ”

amount

```
/*
 * File: Trace.jav
 * ----------------
 * This program do
 * and exists only
 * of parameters a
 */
import acm.program.*;

public class Problem2c extends ConsoleProgram {

   public void run() {
      String s1 = "To err";
      String s2 = "is human!";
      s1 = forgive(s1, s2);
      println(s1 + " " + s2);
   }

   private String forgive(String me, String you) {
      String heart = me.substring(0, you.length() - me.length());
      you = "" + you.charAt(me.length());
      int amount = heart.length();
      me = me.substring(amount + 2) + me.charAt(amount);
      heart += understanding(you, 2) + you + me;
      return heart;
   }

   private char understanding(String you, int num) {
      return (char)(you.charAt(0) + num);
   }

}
```

RUN

s1	"To err"
s2	"is human!"

FORGIVE

me	"To err"
you	"a"
heart	"To "
amount	3

```
/*
 * File: Trace.jav
 * ----------------
 * This program do
 * and exists only
 * of parameters a
 */
import acm.program.*;

public class Problem2c extends ConsoleProgram {

   public void run() {
      String s1 = "To err";
      String s2 = "is human!";
      s1 = forgive(s1, s2);
      println(s1 + " " + s2);
   }

   private String forgive(String me, String you) {
      String heart = me.substring(0, you.length() - me.length());
      you = "" + you.charAt(me.length());
      int amount = heart.length();
      me = me.substring(amount + 2) + me.charAt(amount);
      heart += understanding(you, 2) + you + me;
      return heart;
   }

   private char understanding(String you, int num) {
      return (char)(you.charAt(0) + num);
   }

}
```

RUN

s1	"To err"
s2	"is human!"

FORGIVE

me	"To err"
you	"a"
heart	"To "
amount	3

```
/*
 * File: Trace.jav
 * ----------------
 * This program do
 * and exists only
 * of parameters a
 */
import acm.program.*;

public class Problem2c extends ConsoleProgram {

   public void run() {
      String s1 = "To err";
      String s2 = "is human!";
      s1 = forgive(s1, s2);
      println(s1 + " " + s2);
   }

   private String forgive(String me, String you) {
      String heart = me.substring(0, you.length() - me.length());
      you = "" + you.charAt(me.length());
      int amount = heart.length();
      me = me.substring(amount + 2) + me.charAt(amount);
      heart += understanding(you, 2) + you + me;
      return heart;
   }

   private char understanding(String you, int num) {
      return (char)(you.charAt(0) + num);
   }

}
```

RUN

s1 “To err”

s2 “is human!”

FORGIVE

me “re”

you “a”

heart “To ”

amount 3

```
/*
 * File: Trace.jav
 * ----------------
 * This program do
 * and exists only
 * of parameters a
 */
import acm.program.*;

public class Problem2c extends ConsoleProgram {

   public void run() {
      String s1 = "To err";
      String s2 = "is human!";
      s1 = forgive(s1, s2);
      println(s1 + " " + s2);
   }

   private String forgive(String me, String you) {
      String heart = me.substring(0, you.length() - me.length());
      you = "" + you.charAt(me.length());
      int amount = heart.length();
      me = me.substring(amount + 2) + me.charAt(amount);
      heart += understanding(you, 2) + you + me;
      return heart;
   }

   private char understanding(String you, int num) {
      return (char)(you.charAt(0) + num);
   }

}
```

RUN

s1 “To err”

s2 “is human!”

FORGIVE

me “re”

you “a”

heart “To ”

amount 3

```
/*
 * File: Trace.jav
 * ----------------
 * This program do
 * and exists only
 * of parameters a
 */
import acm.program.*;

public class Problem2c extends ConsoleProgram {

    public void run() {
        String s1 = "To err";
        String s2 = "is human!";
        s1 = forgive(s1, s2);
        println(s1 + " " + s2);
    }

    private String forgive(String me, String you) {
        String heart = me.substring(0, you.length() - me.length());
        you = "" + you.charAt(me.length());
        int amount = heart.length();
        me = me.substring(amount + 2) + me.charAt(amount);
        heart += understanding(you, 2) + you + me;
        return heart;
    }

    private char understanding(String you, int num) {
        return (char)(you.charAt(0) + num);
    }

}
```

RUN

s1 “To err”

s2 “is human!”

FORGIVE

me “re”

you “a”

heart “To ”

amount 3

```
/*
 * File: Trace.jav
 * ----------------
 * This program do
 * and exists only
 * of parameters a
 */
import acm.program.*;

public class Problem2c extends ConsoleProgram {

    public void run() {
        String s1 = "To err";
        String s2 = "is human!";
        s1 = forgive(s1, s2);
        println(s1 + " " + s2);
    }

    private String forgive(String me, String you) {
        String heart = me.substring(0, you.length() - me.length());
        you = "" + you.charAt(me.length());
        int amount = heart.length();
        me = me.substring(amount + 2) + me.charAt(amount);
        heart += understanding(you, 2) + you + me;
        return heart;
    }

    private char understanding(String you, int num) {
        return (char)(you.charAt(0) + num);
    }

}
```

RUN

s1 "To err"

s2 "is human!"

FORGIVE

me "re"

you "a"

heart "To "

amount 3

UNDERSTANDING

you

num

```
/*
 * File: Trace.jav
 * ----------------
 * This program do
 * and exists only
 * of parameters a
 */
import acm.program.*;

public class Problem2c extends ConsoleProgram {

    public void run() {
        String s1 = "To err";
        String s2 = "is human!";
        s1 = forgive(s1, s2);
        println(s1 + " " + s2);
    }

    private String forgive(String me, String you) {
        String heart = me.substring(0, you.length() - me.length());
        you = "" + you.charAt(me.length());
        int amount = heart.length();
        me = me.substring(amount + 2) + me.charAt(amount);
        heart += understanding(you, 2) + you + me;
        return heart;
    }

    private char understanding(String you, int num) {
        return (char)(you.charAt(0) + num);
    }

}
```

RUN

s1 “To err”

s2 “is human!”

FORGIVE

me “re”

you “a”

heart “To ”

amount 3

UNDERSTANDING

you “a”

num

```
/*
 * File: Trace.jav
 * ----------------
 * This program do
 * and exists only
 * of parameters a
 */
import acm.program.*;

public class Problem2c extends ConsoleProgram {

    public void run() {
        String s1 = "To err";
        String s2 = "is human!";
        s1 = forgive(s1, s2);
        println(s1 + " " + s2);
    }

    private String forgive(String me, String you) {
        String heart = me.substring(0, you.length() - me.length());
        you = "" + you.charAt(me.length());
        int amount = heart.length();
        me = me.substring(amount + 2) + me.charAt(amount);
        heart += understanding(you, 2) + you + me;
        return heart;
    }

    private char understanding(String you, int num) {
        return (char)(you.charAt(0) + num);
    }

}
```

RUN

s1	"To err"
s2	"is human!"

FORGIVE

me	"re"
you	"a"
heart	"To "
amount	3

UNDERSTANDING

you	"a"
num	2

```
/*
 * File: Trace.jav
 * ----------------
 * This program do
 * and exists only
 * of parameters a
 */
import acm.program.*;

public class Problem2c extends ConsoleProgram {

    public void run() {
        String s1 = "To err";
        String s2 = "is human!";
        s1 = forgive(s1, s2);
        println(s1 + " " + s2);
    }

    private String forgive(String me, String you) {
        String heart = me.substring(0, you.length() - me.length());
        you = "" + you.charAt(me.length());
        int amount = heart.length();
        me = me.substring(amount + 2) + me.charAt(amount);
        heart += understanding(you, 2) + you + me;
        return heart;
    }

    private char understanding(String you, int num) {
        return (char)(you.charAt(0) + num);
    }

}
```

RUN

s1	"To err"
s2	"is human!"

FORGIVE

me	"re"
you	"a"
heart	"To "
amount	3

UNDERSTANDING

you	"a"
num	2

```
/*
 * File: Trace.jav
 * ----------------
 * This program do
 * and exists only
 * of parameters a
 */
import acm.program.*;

public class Problem2c extends ConsoleProgram {

   public void run() {
      String s1 = "To err";
      String s2 = "is human!";
      s1 = forgive(s1, s2);
      println(s1 + " " + s2);
   }

   private String forgive(String me, String you) {
      String heart = me.substring(0, you.length() - me.length());
      you = "" + you.charAt(me.length());
      int amount = heart.length();
      me = me.substring(amount + 2) + me.charAt(amount);
      heart += understanding(you, 2) + you + me;
      return heart;
   }

   private char understanding(String you, int num) {
      return (char)(you.charAt(0) + num);
   }

}
```

RUN

s1	"To err"
s2	"is human!"

FORGIVE

me	"re"
you	"a"
heart	"To care"
amount	3

UNDERSTANDING

you	"a"
num	2

```
/*
 * File: Trace.jav
 * ----------------
 * This program do
 * and exists only
 * of parameters a
 */
import acm.program.*;

public class Problem2c extends ConsoleProgram {

    public void run() {
        String s1 = "To err";
        String s2 = "is human!";
        s1 = forgive(s1, s2);
        println(s1 + " " + s2);
    }

    private String forgive(String me, String you) {
        String heart = me.substring(0, you.length() - me.length());
        you = "" + you.charAt(me.length());
        int amount = heart.length();
        me = me.substring(amount + 2) + me.charAt(amount);
        heart += understanding(you, 2) + you + me;
        return heart;
    }

    private char understanding(String you, int num) {
        return (char)(you.charAt(0) + num);
    }

}
```

RUN

s1	"To err"
s2	"is human!"

FORGIVE

me	"re"
you	"a"
heart	"To care"
amount	3

UNDERSTANDING

you	"a"
num	2

```
/*
 * File: Trace.jav
 * ----------------
 * This program do
 * and exists only
 * of parameters a
 */
import acm.program.*;

public class Problem2c extends ConsoleProgram {

   public void run() {
      String s1 = "To err";
      String s2 = "is human!";
      s1 = forgive(s1, s2);
      println(s1 + " " + s2);
   }

   private String forgive(String me, String you) {
      String heart = me.substring(0, you.length() - me.length());
      you = "" + you.charAt(me.length());
      int amount = heart.length();
      me = me.substring(amount + 2) + me.charAt(amount);
      heart += understanding(you, 2) + you + me;
      return heart;
   }

   private char understanding(String you, int num) {
      return (char)(you.charAt(0) + num);
   }

}
```

RUN

s1 "To err"

s2 "is human!"

FORGIVE

me "re"

you "a"

heart "To care"

amount 3

UNDERSTANDING

you "a"

num 2

```
/*
 * File: Trace.jav
 * ----------------
 * This program do
 * and exists only
 * of parameters a
 */
import acm.program.*;

public class Problem2c extends ConsoleProgram {

   public void run() {
      String s1 = "To err";
      String s2 = "is human!";
      s1 = forgive(s1, s2);
      println(s1 + " " + s2);
   }

   private String forgive(String me, String you) {
      String heart = me.substring(0, you.length() - me.length());
      you = "" + you.charAt(me.length());
      int amount = heart.length();
      me = me.substring(amount + 2) + me.charAt(amount);
      heart += understanding(you, 2) + you + me;
      return heart;
   }

   private char understanding(String you, int num) {
      return (char)(you.charAt(0) + num);
   }

}
```

RUN

s1 "To care"

s2 "is human!"

FORGIVE

me "re"

you "a"

heart "To care"

amount 3

UNDERSTANDING

you "a"

num 2

```
/*
 * File: Trace.jav
 * ----------------
 * This program do
 * and exists only
 * of parameters a
 */
import acm.program.*;

public class Problem2c extends ConsoleProgram {

   public void run() {
      String s1 = "To err";
      String s2 = "is human!";
      s1 = forgive(s1, s2);
      println(s1 + " " + s2);
   }

   private String forgive(String me, String you) {
      String heart = me.substring(0, you.length() - me.length());
      you = "" + you.charAt(me.length());
      int amount = heart.length();
      me = me.substring(amount + 2) + me.charAt(amount);
      heart += understanding(you, 2) + you + me;
      return heart;
   }

   private char understanding(String you, int num) {
      return (char)(you.charAt(0) + num);
   }

}
```

RUN

s1 “To care”

s2 “is human!”

FORGIVE

me “re”

you “a”

heart “To care”

amount 3

UNDERSTANDING

you “a”

num 2

```
/*
 * File: Trace.jav
 * ----------------
 * This program do
 * and exists only
 * of parameters a
 */
import acm.program.*;

public class Problem2c extends ConsoleProgram {

   public void run() {
      String s1 = "To err";
      String s2 = "is human!";
      s1 = forgive(s1, s2);
      println(s1 + " " + s2);
   }

   private String forgive(String me, String you) {
      String heart = me.substring(0, you.length() - me.length());
      you = "" + you.charAt(me.length());
      int amount = heart.length();
      me = me.substring(amount + 2) + me.charAt(amount);
      heart += understanding(you, 2) + you + me;
      return heart;
   }

   private char understanding(String you, int num) {
      return (char)(you.charAt(0) + num);
   }

}
```

RUN

s1 “To care”

s2 “is human!”

“To care is human!”

FORGIVE

me “re”

you “a”

heart “To care”

amount 3

UNDERSTANDING

you “a”

num 2

```
/*
 * File: Trace.jav
 * ----------------
 * This program do
 * and exists only
 * of parameters a
 */
import acm.program.*;

public class Problem2c extends ConsoleProgram {

   public void run() {
      String s1 = "To err";
      String s2 = "is human!";
      s1 = forgive(s1, s2);
      println(s1 + " " + s2);
   }

   private String forgive(String me, String you) {
      String heart = me.substring(0, you.length() - me.length());
      you = "" + you.charAt(me.length());
      int amount = heart.length();
      me = me.substring(amount + 2) + me.charAt(amount);
      heart += understanding(you, 2) + you + me;
      return heart;
   }

   private char understanding(String you, int num) {
      return (char)(you.charAt(0) + num);
   }

}
```

RUN

s1 “To care”

s2 “is human!”

“To care is human!”

FORGIVE

me “re”

you “a”

heart “To care”

amount 3

UNDERSTANDING

you “a”

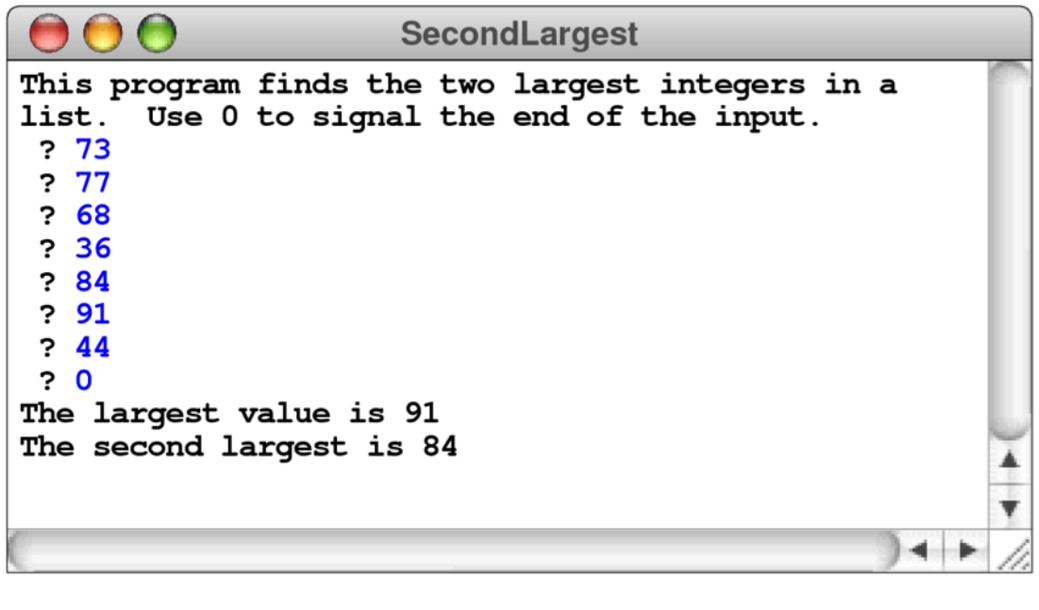
num 2

The plan

- General info
- Karel
- Expression evaluation
- Program tracing
- Simple Java and randomness
- Graphics and MouseEvents
- String manipulation

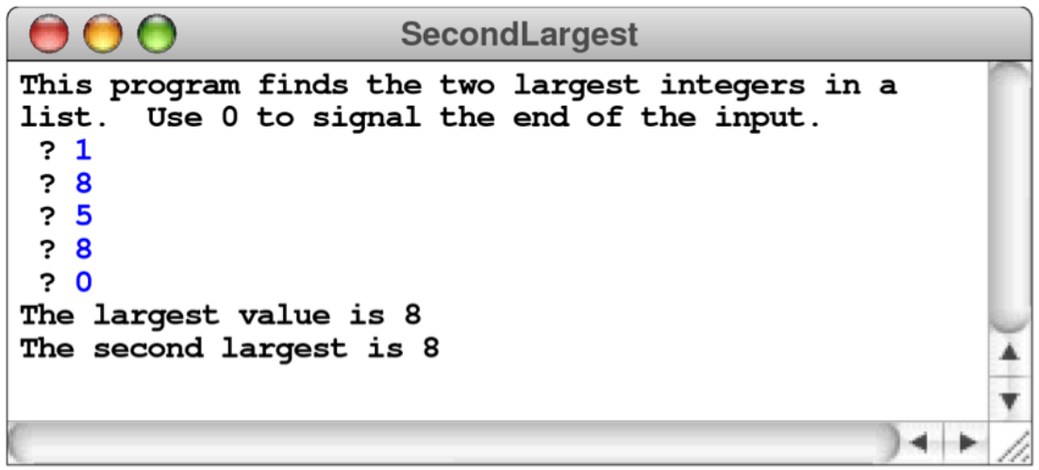
Problem 3: Simple Java programs (15 points)

In Assignment #2, you wrote a program to find the largest and smallest integers in a list entered by the user. For this problem, write a similar program that instead finds the largest and the second-largest integer. As in the homework problem, you should use 0 as a sentinel to indicate the end of the input list. Thus, a sample run of the program might look like this:



To reduce the number of special cases, you may make the following assumptions in writing your code:

- The user must enter at least two values before the sentinel.
- All input values are positive integers.
- If the largest value appears more than once, that value should be listed as both the largest and second-largest value, as shown in the following sample run:



```
/*
 * File: SecondLargest.java
 * ------------------------
 * This program finds the largest and second largest values in a list.
 */

import acm.program.*;

public class SecondLargest extends ConsoleProgram {

   public void run() {

   }

/* Sentinel value to signal end of input */
   private static final int SENTINEL = 0;

}
```

```
/*
 * File: SecondLargest.java
 * ------------------------
 * This program finds the largest and second largest values in a list.
 */

import acm.program.*;

public class SecondLargest extends ConsoleProgram {

   public void run() {
      println("This program finds the two largest integers in a");
      println("list.  Use " + SENTINEL + " to signal the end of the input.");
```

```
   }

/* Sentinel value to signal end of input */
   private static final int SENTINEL = 0;

}
```

```
/*
 * File: SecondLargest.java
 * ------------------------
 * This program finds the largest and second largest values in a list.
 */

import acm.program.*;

public class SecondLargest extends ConsoleProgram {

   public void run() {
      println("This program finds the two largest integers in a");
      println("list.  Use " + SENTINEL + " to signal the end of the input.");

      int largest = 0;
      int secondLargest = 0;
```

```
   }

/* Sentinel value to signal end of input */
   private static final int SENTINEL = 0;

}
```

```
/*
 * File: SecondLargest.java
 * ------------------------
 * This program finds the largest and second largest values in a list.
 */

import acm.program.*;

public class SecondLargest extends ConsoleProgram {

   public void run() {
      println("This program finds the two largest integers in a");
      println("list.  Use " + SENTINEL + " to signal the end of the input.");

      int largest = 0;
      int secondLargest = 0;
      while (true) {

      }

   }

/* Sentinel value to signal end of input */
   private static final int SENTINEL = 0;

}
```

```
/*
 * File: SecondLargest.java
 * ------------------------
 * This program finds the largest and second largest values in a list.
 */

import acm.program.*;

public class SecondLargest extends ConsoleProgram {

   public void run() {
      println("This program finds the two largest integers in a");
      println("list.  Use " + SENTINEL + " to signal the end of the input.");

      int largest = 0;
      int secondLargest = 0;
      while (true) {
         int number = readInt(" ? ");
```

```
      }

   }

/* Sentinel value to signal end of input */
   private static final int SENTINEL = 0;

}
```

```
/*
 * File: SecondLargest.java
 * ------------------------
 * This program finds the largest and second largest values in a list.
 */

import acm.program.*;

public class SecondLargest extends ConsoleProgram {

   public void run() {
      println("This program finds the two largest integers in a");
      println("list.  Use " + SENTINEL + " to signal the end of the input.");

      int largest = 0;
      int secondLargest = 0;
      while (true) {
         int number = readInt(" ? ");
         if (number == SENTINEL) break;
```

```
      }

   }

/* Sentinel value to signal end of input */
   private static final int SENTINEL = 0;

}
```

```
/*
 * File: SecondLargest.java
 * ------------------------
 * This program finds the largest and second largest values in a list.
 */

import acm.program.*;

public class SecondLargest extends ConsoleProgram {

   public void run() {
      println("This program finds the two largest integers in a");
      println("list.  Use " + SENTINEL + " to signal the end of the input.");
      int count = 0;
      int largest = 0;
      int secondLargest = 0;
      while (true) {
         int number = readInt(" ? ");
         if (number == SENTINEL) break;
```

```
      }

   }

/* Sentinel value to signal end of input */
   private static final int SENTINEL = 0;

}
```

```
/*
 * File: SecondLargest.java
 * ------------------------
 * This program finds the largest and second largest values in a list.
 */

import acm.program.*;

public class SecondLargest extends ConsoleProgram {

   public void run() {
      println("This program finds the two largest integers in a");
      println("list.  Use " + SENTINEL + " to signal the end of the input.");
      int count = 0;
      int largest = 0;
      int secondLargest = 0;
      while (true) {
         int number = readInt(" ? ");
         if (number == SENTINEL) break;
         count++;
```

```
      }
```

```
   }

/* Sentinel value to signal end of input */
   private static final int SENTINEL = 0;

}
```

```
/*
 * File: SecondLargest.java
 * ------------------------
 * This program finds the largest and second largest values in a list.
 */

import acm.program.*;

public class SecondLargest extends ConsoleProgram {

   public void run() {
      println("This program finds the two largest integers in a");
      println("list.  Use " + SENTINEL + " to signal the end of the input.");
      int count = 0;
      int largest = 0;
      int secondLargest = 0;
      while (true) {
         int number = readInt(" ? ");
         if (number == SENTINEL) break;
         count++;
         if (count == 1) {

         } else {




         }
      }
   }

/* Sentinel value to signal end of input */
   private static final int SENTINEL = 0;

}
```

```
/*
 * File: SecondLargest.java
 * ------------------------
 * This program finds the largest and second largest values in a list.
 */

import acm.program.*;

public class SecondLargest extends ConsoleProgram {

   public void run() {
      println("This program finds the two largest integers in a");
      println("list.  Use " + SENTINEL + " to signal the end of the input.");
      int count = 0;
      int largest = 0;
      int secondLargest = 0;
      while (true) {
         int number = readInt(" ? ");
         if (number == SENTINEL) break;
         count++;
         if (count == 1) {
            largest = number;
         } else {
```

```
         }
      }
   }

/* Sentinel value to signal end of input */
   private static final int SENTINEL = 0;

}
```

```
/*
 * File: SecondLargest.java
 * ------------------------
 * This program finds the largest and second largest values in a list.
 */

import acm.program.*;

public class SecondLargest extends ConsoleProgram {

   public void run() {
      println("This program finds the two largest integers in a");
      println("list.  Use " + SENTINEL + " to signal the end of the input.");
      int count = 0;
      int largest = 0;
      int secondLargest = 0;
      while (true) {
         int number = readInt(" ? ");
         if (number == SENTINEL) break;
         count++;
         if (count == 1) {
            largest = number;
         } else {
            if (number > largest) {









         }
      }

   }

/* Sentinel value to signal end of input */
   private static final int SENTINEL = 0;

}
```

```
/*
 * File: SecondLargest.java
 * ------------------------
 * This program finds the largest and second largest values in a list.
 */

import acm.program.*;

public class SecondLargest extends ConsoleProgram {

   public void run() {
      println("This program finds the two largest integers in a");
      println("list.  Use " + SENTINEL + " to signal the end of the input.");
      int count = 0;
      int largest = 0;
      int secondLargest = 0;
      while (true) {
         int number = readInt(" ? ");
         if (number == SENTINEL) break;
         count++;
         if (count == 1) {
            largest = number;
         } else {
            if (number > largest) {
               secondLargest = largest;









         }
      }

   }

/* Sentinel value to signal end of input */
   private static final int SENTINEL = 0;

}
```

```
/*
 * File: SecondLargest.java
 * ------------------------
 * This program finds the largest and second largest values in a list.
 */

import acm.program.*;

public class SecondLargest extends ConsoleProgram {

   public void run() {
      println("This program finds the two largest integers in a");
      println("list.  Use " + SENTINEL + " to signal the end of the input.");
      int count = 0;
      int largest = 0;
      int secondLargest = 0;
      while (true) {
         int number = readInt(" ? ");
         if (number == SENTINEL) break;
         count++;
         if (count == 1) {
            largest = number;
         } else {
            if (number > largest) {
               secondLargest = largest;
               largest = number;


         }
      }

   }

/* Sentinel value to signal end of input */
   private static final int SENTINEL = 0;

}
```

```
/*
 * File: SecondLargest.java
 * ------------------------
 * This program finds the largest and second largest values in a list.
 */

import acm.program.*;

public class SecondLargest extends ConsoleProgram {

   public void run() {
      println("This program finds the two largest integers in a");
      println("list.  Use " + SENTINEL + " to signal the end of the input.");
      int count = 0;
      int largest = 0;
      int secondLargest = 0;
      while (true) {
         int number = readInt(" ? ");
         if (number == SENTINEL) break;
         count++;
         if (count == 1) {
            largest = number;
         } else {
            if (number > largest) {
               secondLargest = largest;
               largest = number;
            } else if (count == 2 || number > secondLargest) {

            }
         }
      }

   }

/* Sentinel value to signal end of input */
   private static final int SENTINEL = 0;

}
```

```
/*
 * File: SecondLargest.java
 * ------------------------
 * This program finds the largest and second largest values in a list.
 */

import acm.program.*;

public class SecondLargest extends ConsoleProgram {

   public void run() {
      println("This program finds the two largest integers in a");
      println("list.  Use " + SENTINEL + " to signal the end of the input.");
      int count = 0;
      int largest = 0;
      int secondLargest = 0;
      while (true) {
         int number = readInt(" ? ");
         if (number == SENTINEL) break;
         count++;
         if (count == 1) {
            largest = number;
         } else {
            if (number > largest) {
               secondLargest = largest;
               largest = number;
            } else if (count == 2 || number > secondLargest) {
               secondLargest = number;
            }
         }
      }
   }

/* Sentinel value to signal end of input */
   private static final int SENTINEL = 0;

}
```

```
/*
 * File: SecondLargest.java
 * ------------------------
 * This program finds the largest and second largest values in a list.
 */

import acm.program.*;

public class SecondLargest extends ConsoleProgram {

   public void run() {
      println("This program finds the two largest integers in a");
      println("list.  Use " + SENTINEL + " to signal the end of the input.");
      int count = 0;
      int largest = 0;
      int secondLargest = 0;
      while (true) {
         int number = readInt(" ? ");
         if (number == SENTINEL) break;
         count++;
         if (count == 1) {
            largest = number;
         } else {
            if (number > largest) {
               secondLargest = largest;
               largest = number;
            } else if (count == 2 || number > secondLargest) {
               secondLargest = number;
            }
         }
      }
      if (count == 0) {
         println("No values were entered");
      } else {




      }
   }

/* Sentinel value to signal end of input */
   private static final int SENTINEL = 0;

}
```

```
/*
 * File: SecondLargest.java
 * ------------------------
 * This program finds the largest and second largest values in a list.
 */

import acm.program.*;

public class SecondLargest extends ConsoleProgram {

   public void run() {
      println("This program finds the two largest integers in a");
      println("list.  Use " + SENTINEL + " to signal the end of the input.");
      int count = 0;
      int largest = 0;
      int secondLargest = 0;
      while (true) {
         int number = readInt(" ? ");
         if (number == SENTINEL) break;
         count++;
         if (count == 1) {
            largest = number;
         } else {
            if (number > largest) {
               secondLargest = largest;
               largest = number;
            } else if (count == 2 || number > secondLargest) {
               secondLargest = number;
            }
         }
      }
      if (count == 0) {
         println("No values were entered");
      } else {
         println("The largest value is " + largest);



      }
   }

/* Sentinel value to signal end of input */
   private static final int SENTINEL = 0;

}
```

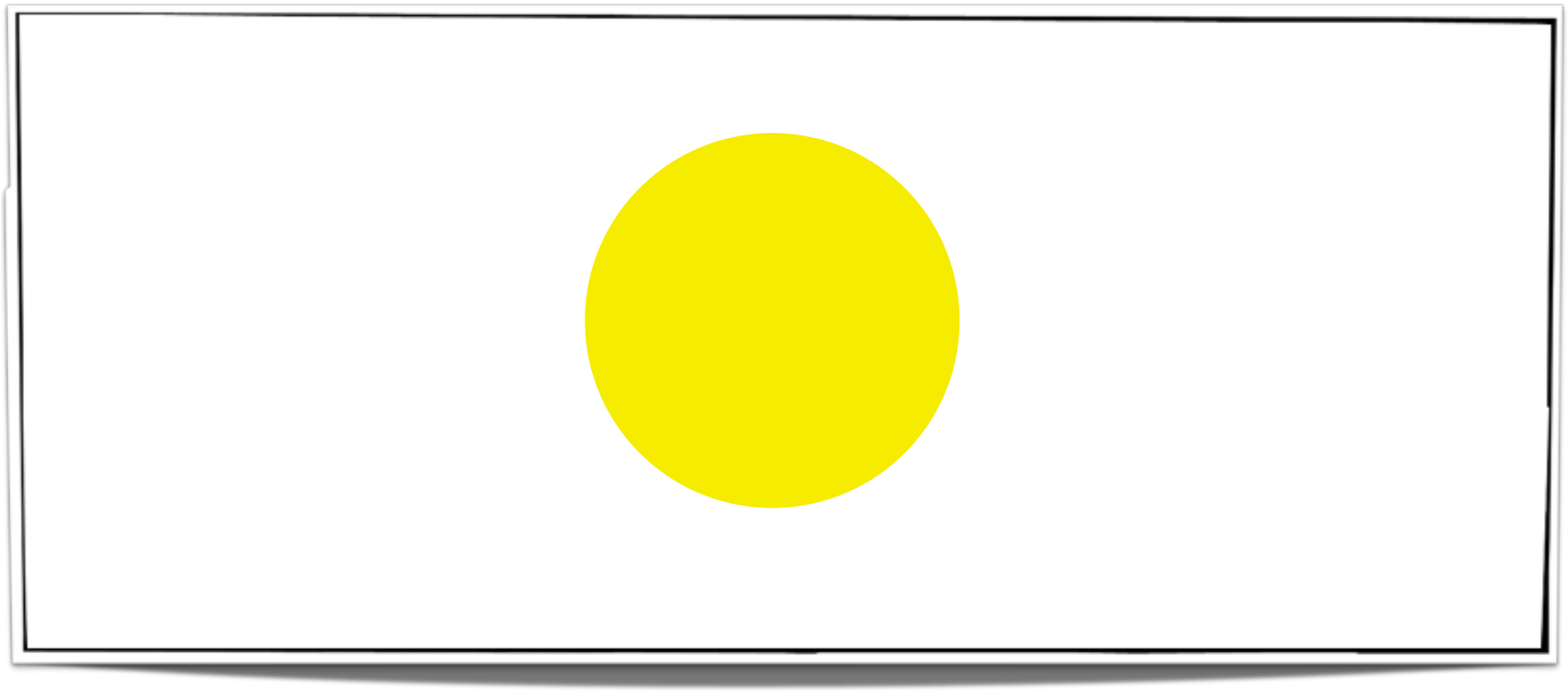
```
/*
 * File: SecondLargest.java
 * ------------------------
 * This program finds the largest and second largest values in a list.
 */

import acm.program.*;

public class SecondLargest extends ConsoleProgram {

   public void run() {
      println("This program finds the two largest integers in a");
      println("list.  Use " + SENTINEL + " to signal the end of the input.");
      int count = 0;
      int largest = 0;
      int secondLargest = 0;
      while (true) {
         int number = readInt(" ? ");
         if (number == SENTINEL) break;
         count++;
         if (count == 1) {
            largest = number;
         } else {
            if (number > largest) {
               secondLargest = largest;
               largest = number;
            } else if (count == 2 || number > secondLargest) {
               secondLargest = number;
            }
         }
      }
      if (count == 0) {
         println("No values were entered");
      } else {
         println("The largest value is " + largest);
         if (count > 1) {

         }
      }
   }

/* Sentinel value to signal end of input */
   private static final int SENTINEL = 0;

}
```

```
/*
 * File: SecondLargest.java
 * ------------------------
 * This program finds the largest and second largest values in a list.
 */

import acm.program.*;

public class SecondLargest extends ConsoleProgram {

   public void run() {
      println("This program finds the two largest integers in a");
      println("list.  Use " + SENTINEL + " to signal the end of the input.");
      int count = 0;
      int largest = 0;
      int secondLargest = 0;
      while (true) {
         int number = readInt(" ? ");
         if (number == SENTINEL) break;
         count++;
         if (count == 1) {
            largest = number;
         } else {
            if (number > largest) {
               secondLargest = largest;
               largest = number;
            } else if (count == 2 || number > secondLargest) {
               secondLargest = number;
            }
         }
      }
      if (count == 0) {
         println("No values were entered");
      } else {
         println("The largest value is " + largest);
         if (count > 1) {
            println("The second largest value is " + secondLargest);
         }
      }
   }

/* Sentinel value to signal end of input */
   private static final int SENTINEL = 0;

}
```

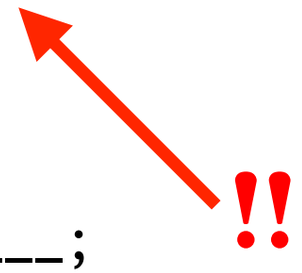
The plan

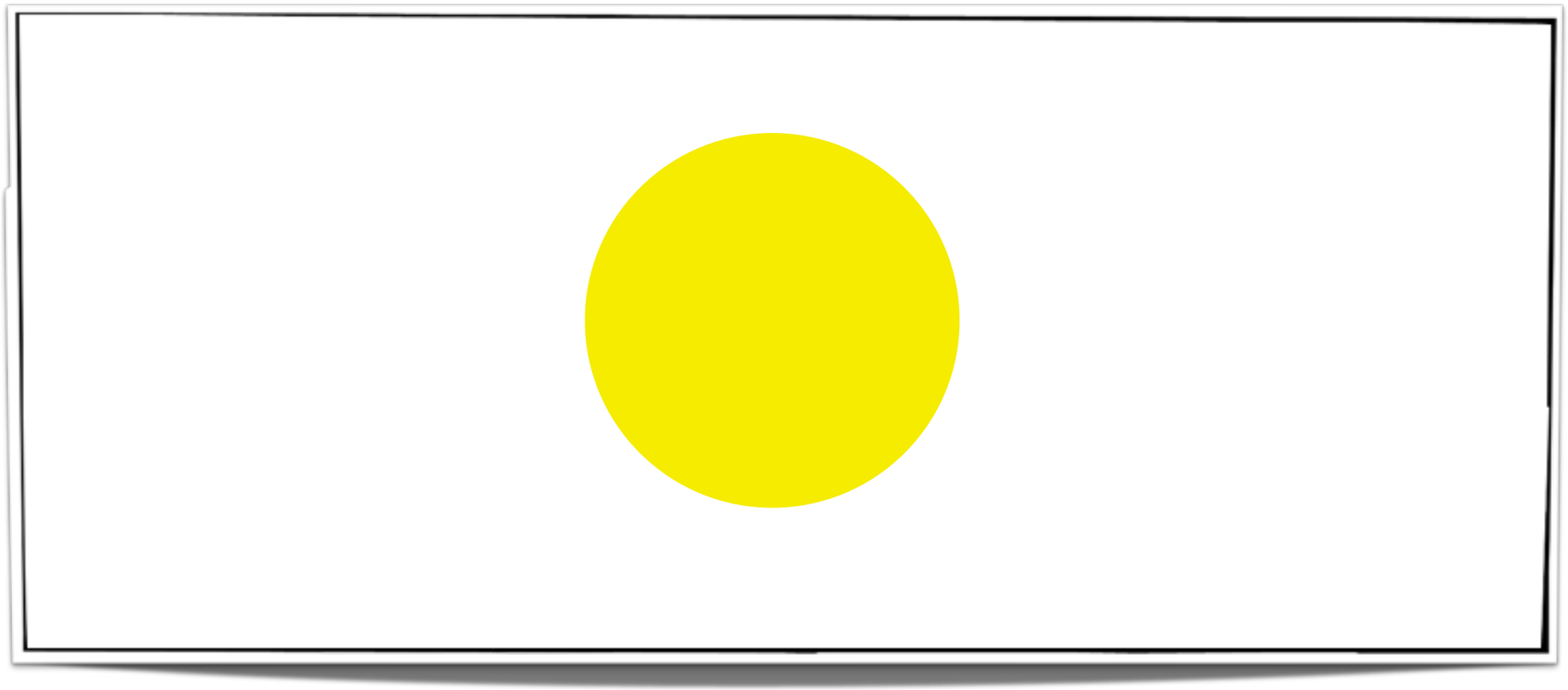
- General info
- Karel
- Expression evaluation
- Program tracing
- Simple Java and randomness
- Graphics and MouseEvents
- String manipulation

```
private void drawSun() {
  GOval sun = new GOval(SUN_DIAMETER, SUN_DIAMETER);
  sun.setColor(Color.YELLOW);
  sun.setFilled(true);
  sun.setFillColor(Color.YELLOW);
  double sunX = ____________________________________;
  double sunY = ____________________________________;
  add(sun, sunX, sunY);
}
```

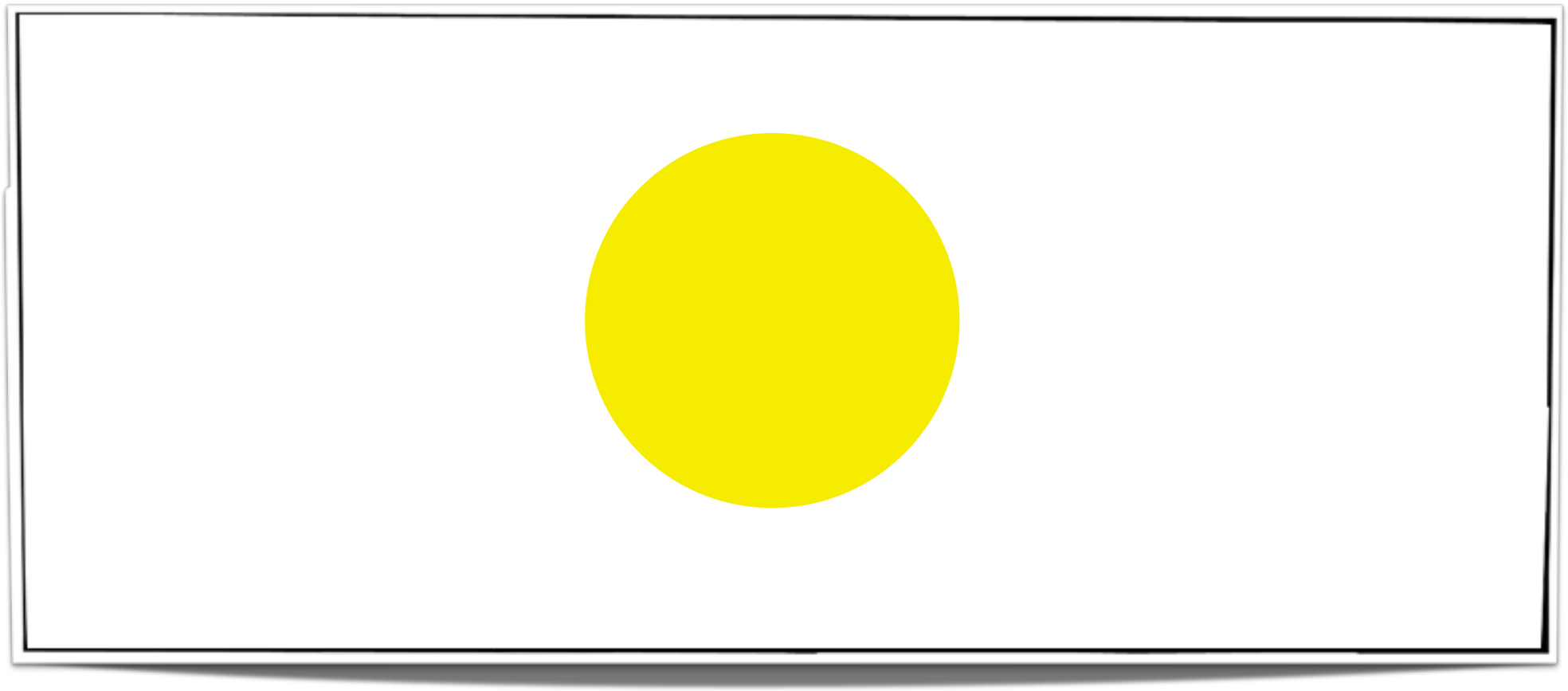


```
private void drawSun() {
  GOval sun = new GOval(SUN_DIAMETER, SUN_DIAMETER);
  sun.setColor(Color.YELLOW);
  sun.setFilled(true);
  sun.setFillColor(Color.YELLOW);
  double sunX = ____________________________________;
  double sunY = ____________________________________;
  add(sun, sunX, sunY);
}
```

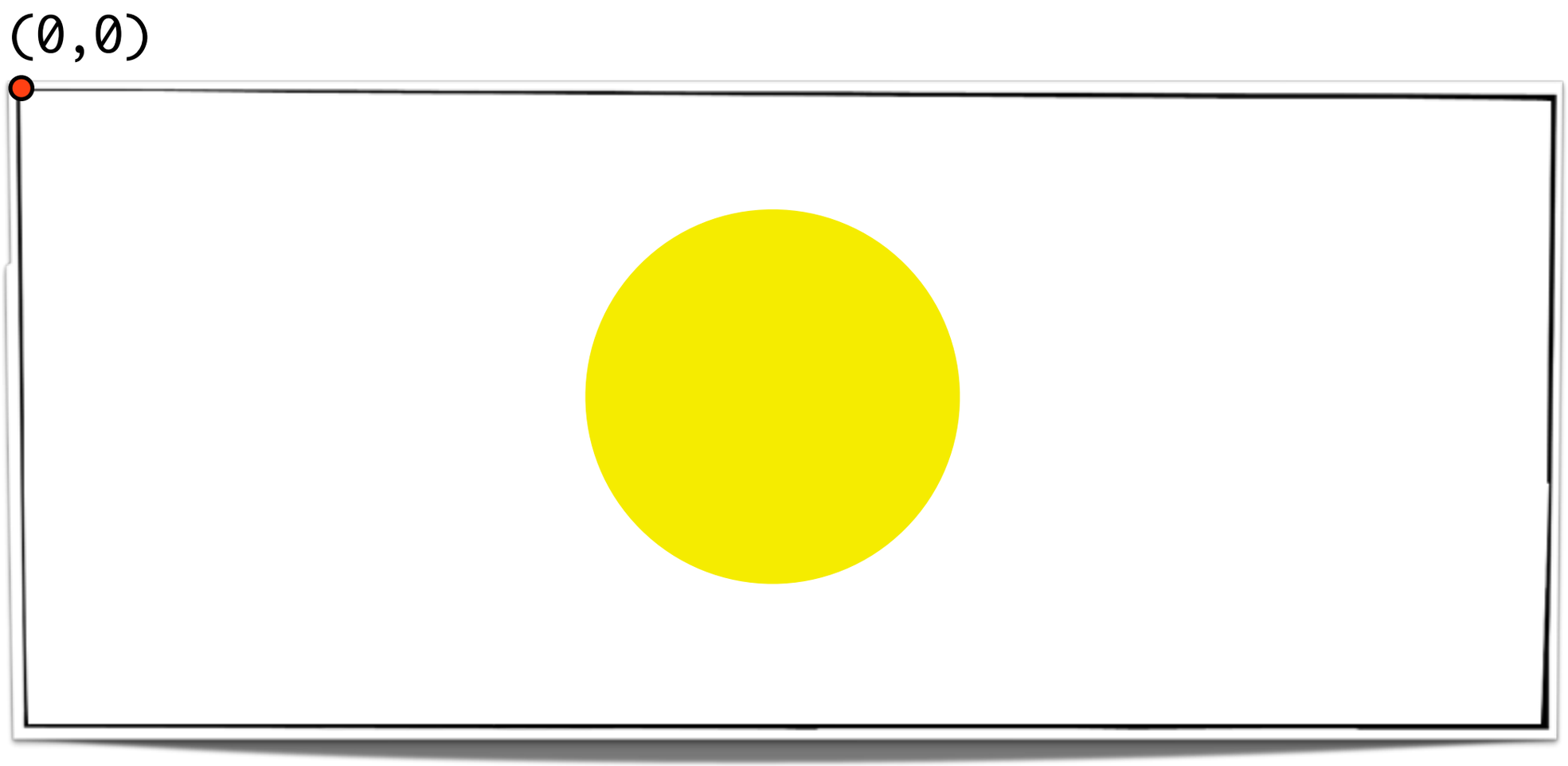




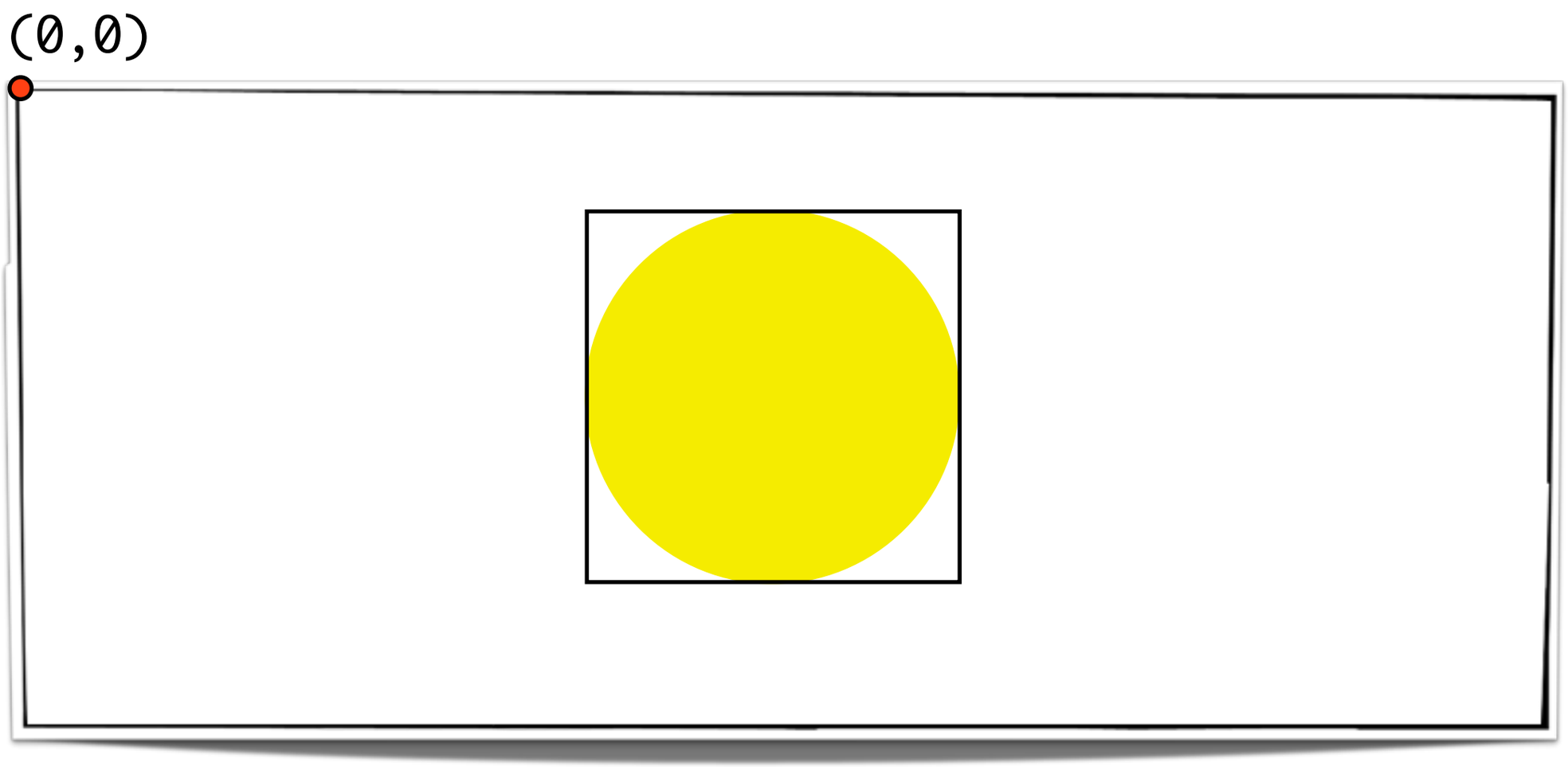
```
private void drawSun() {
  GOval sun = new GOval(SUN_DIAMETER, SUN_DIAMETER);
  sun.setColor(Color.YELLOW);
  sun.setFilled(true);
  sun.setFillColor(Color.YELLOW);
  double sunX = ____________________________________;
  double sunY = ____________________________________;
  add(sun, sunX, sunY);
}
```



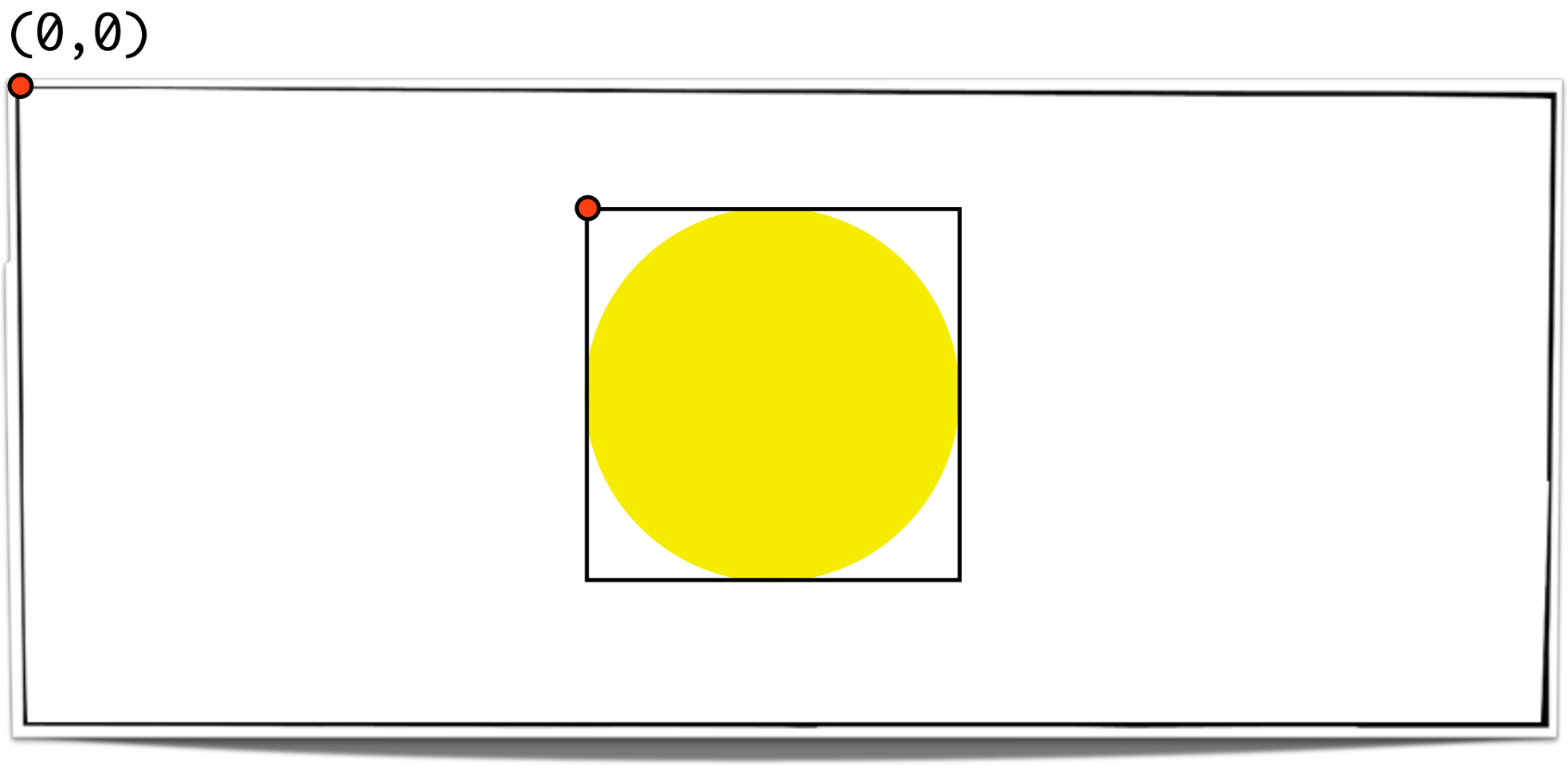
```
private void drawSun() {
  GOval sun = new GOval(SUN_DIAMETER, SUN_DIAMETER);
  sun.setColor(Color.YELLOW);
  sun.setFilled(true);
  sun.setFillColor(Color.YELLOW);
  double sunX = ____________________________________;
  double sunY = ____________________________________;
  add(sun, sunX, sunY);
}
```



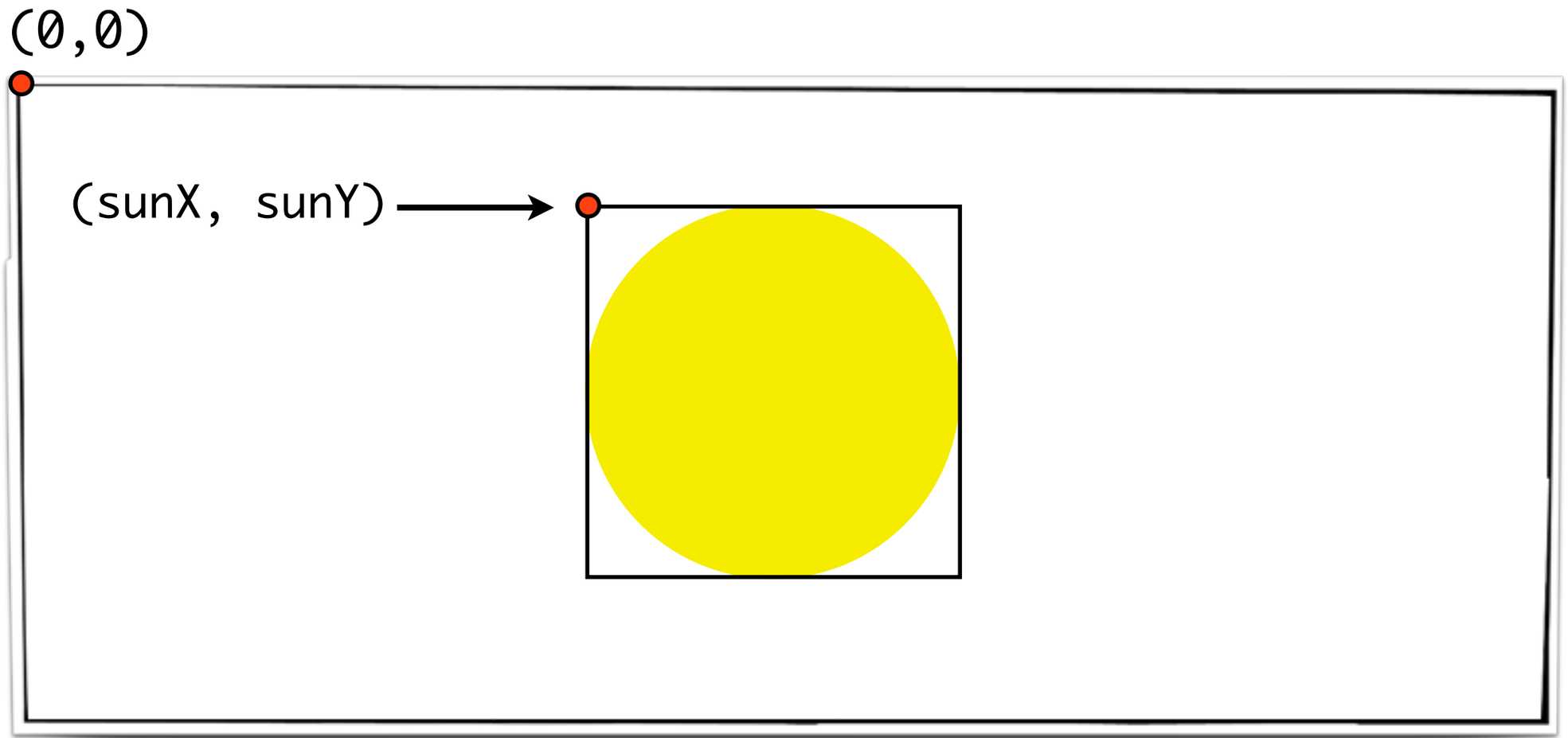
```
private void drawSun() {
  GOval sun = new GOval(SUN_DIAMETER, SUN_DIAMETER);
  sun.setColor(Color.YELLOW);
  sun.setFilled(true);
  sun.setFillColor(Color.YELLOW);
  double sunX = ____________________________________;
  double sunY = ____________________________________;
  add(sun, sunX, sunY);
}
```



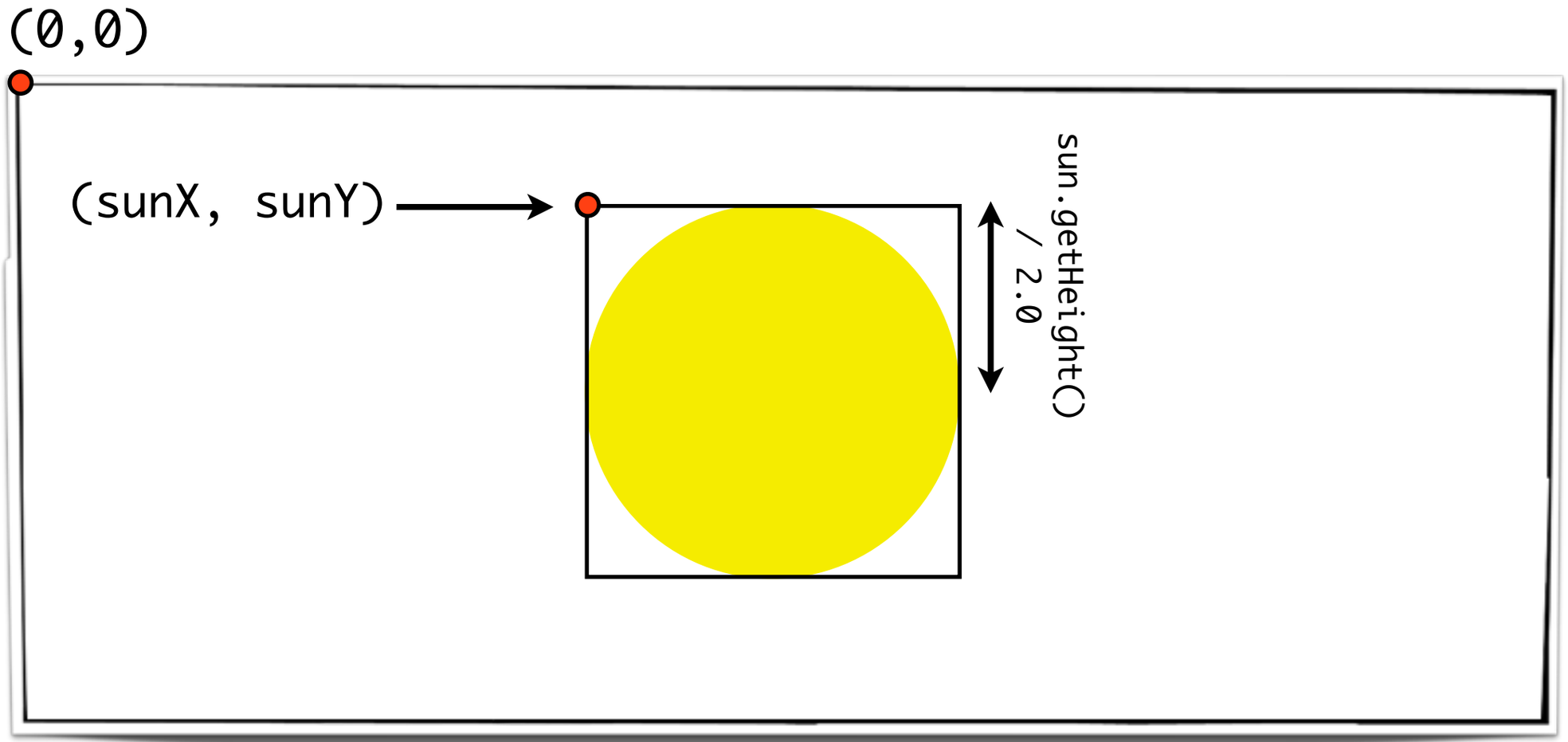
```
private void drawSun() {
   GOval sun = new GOval(SUN_DIAMETER, SUN_DIAMETER);
   sun.setColor(Color.YELLOW);
   sun.setFilled(true);
   sun.setFillColor(Color.YELLOW);
   double sunX = ____________________________________;
   double sunY = ____________________________________;
   add(sun, sunX, sunY);
}
```



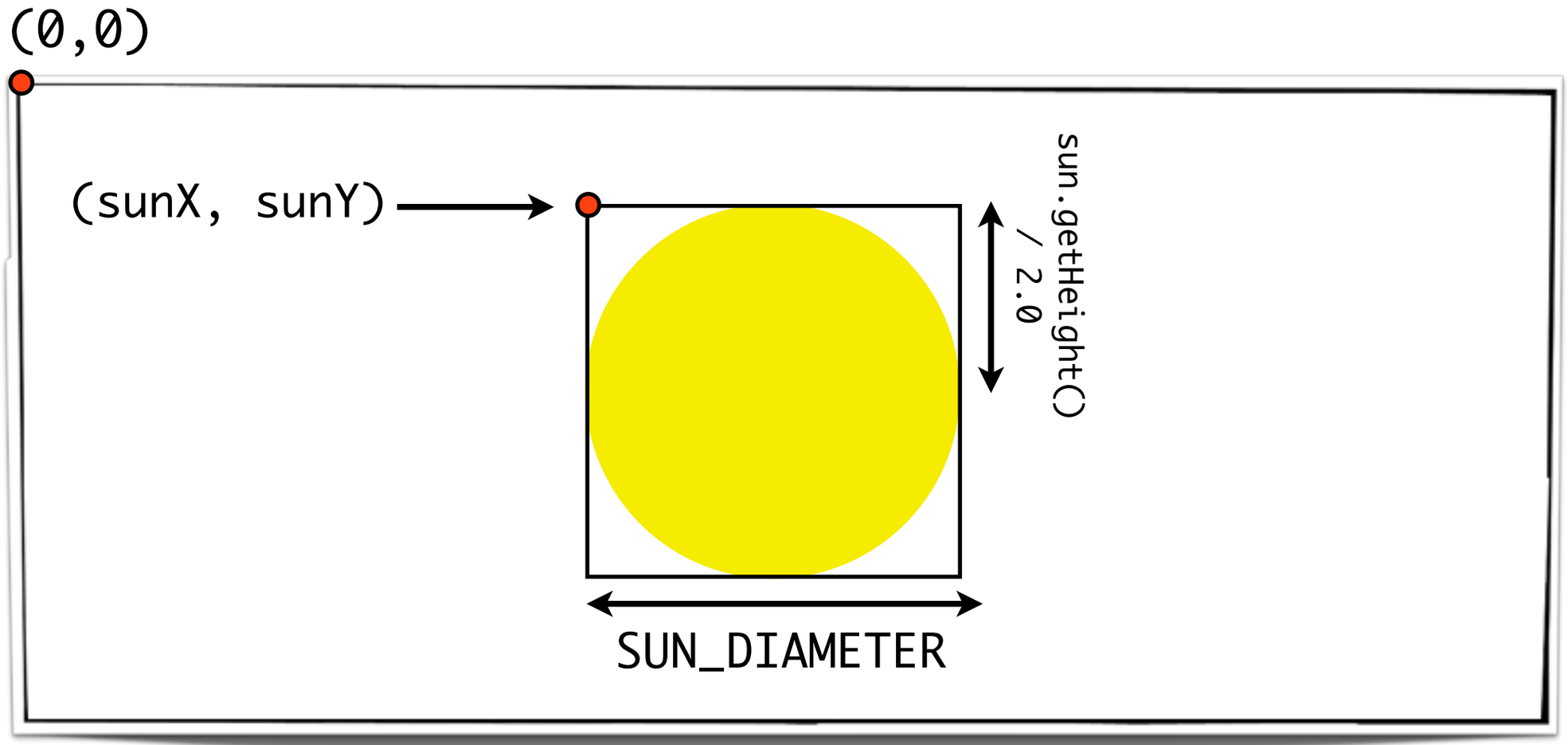
```
private void drawSun() {
  GOval sun = new GOval(SUN_DIAMETER, SUN_DIAMETER);
  sun.setColor(Color.YELLOW);
  sun.setFilled(true);
  sun.setFillColor(Color.YELLOW);
  double sunX = ____________________________________;
  double sunY = ____________________________________;
  add(sun, sunX, sunY);
}
```



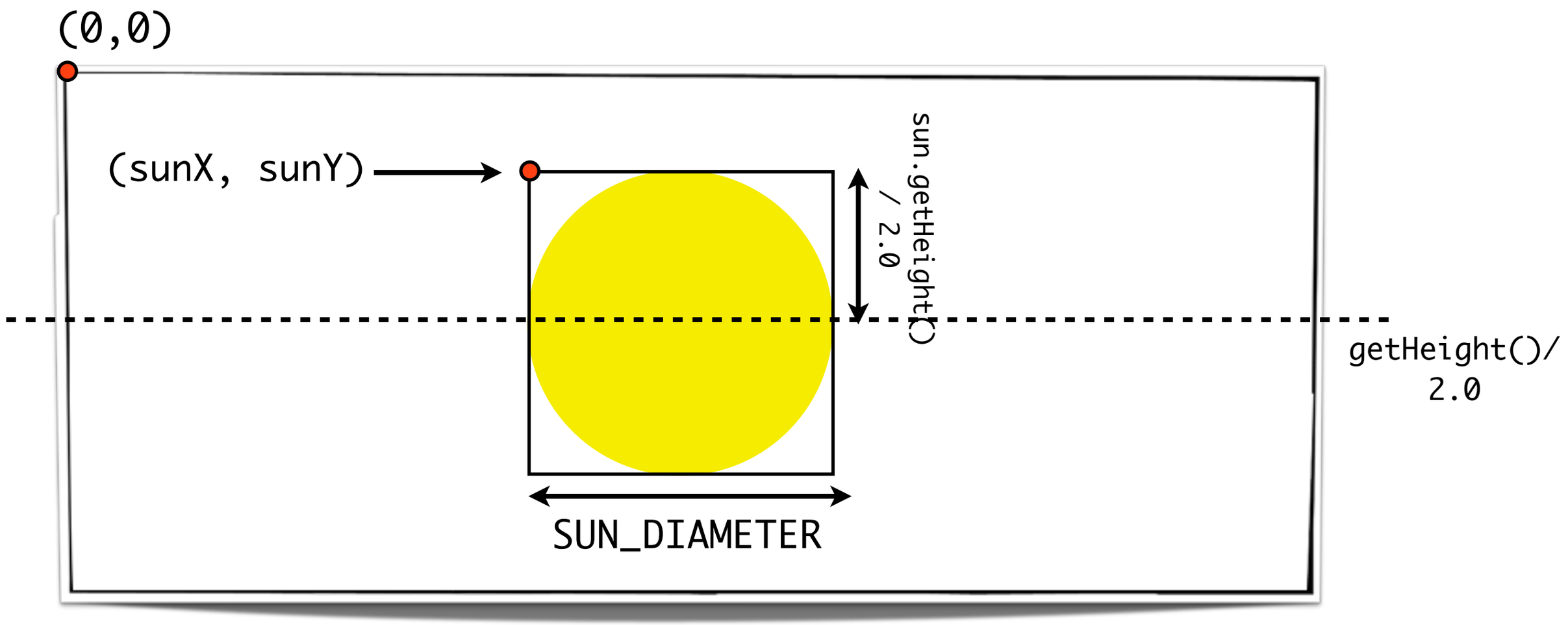
```
private void drawSun() {
   GOval sun = new GOval(SUN_DIAMETER, SUN_DIAMETER);
   sun.setColor(Color.YELLOW);
   sun.setFilled(true);
   sun.setFillColor(Color.YELLOW);
   double sunX = ____________________________________;
   double sunY = ____________________________________;
   add(sun, sunX, sunY);
}
```



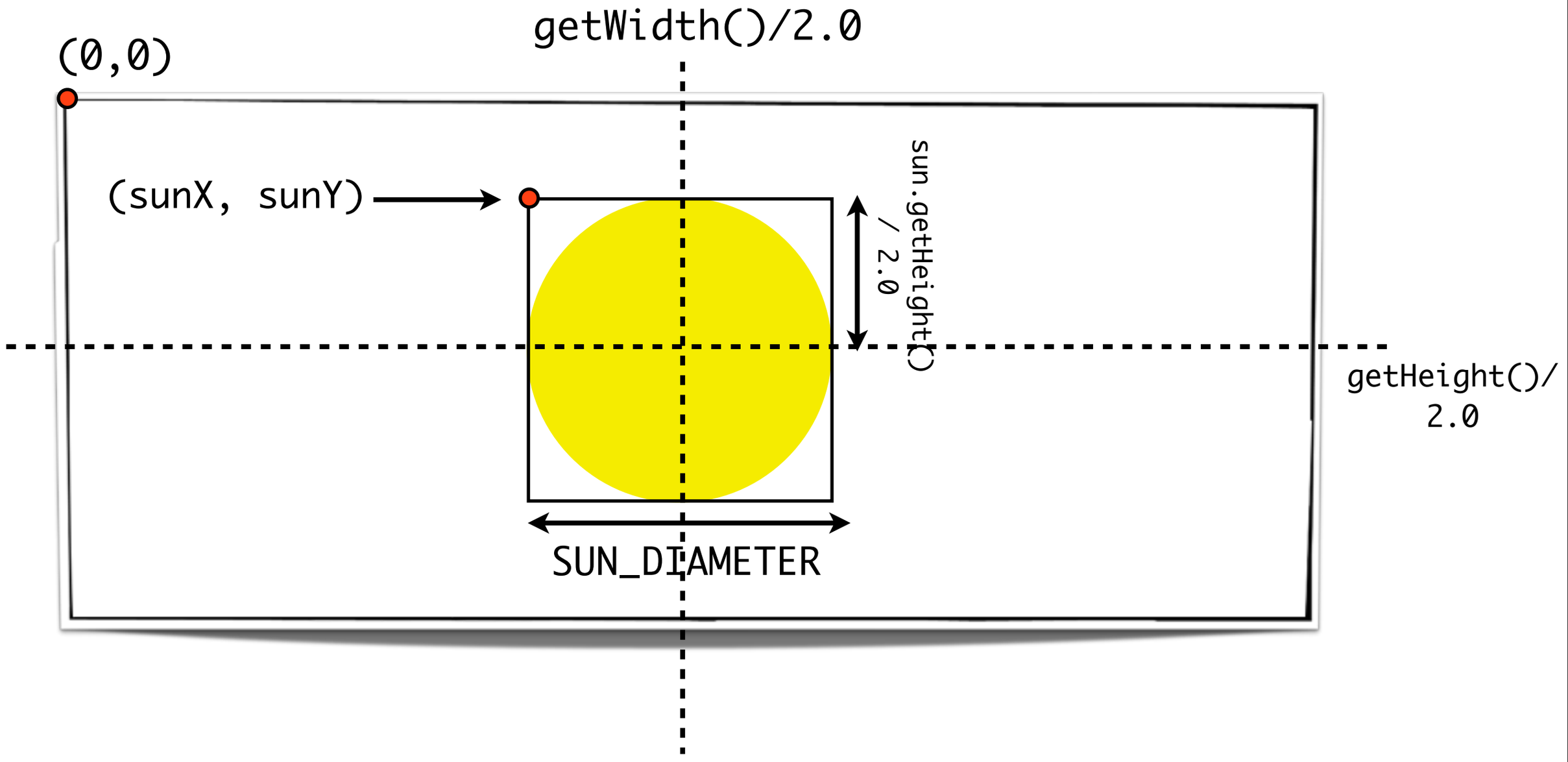
```
private void drawSun() {
   GOval sun = new GOval(SUN_DIAMETER, SUN_DIAMETER);
   sun.setColor(Color.YELLOW);
   sun.setFilled(true);
   sun.setFillColor(Color.YELLOW);
   double sunX = ____________________________________;
   double sunY = ____________________________________;
   add(sun, sunX, sunY);
}
```



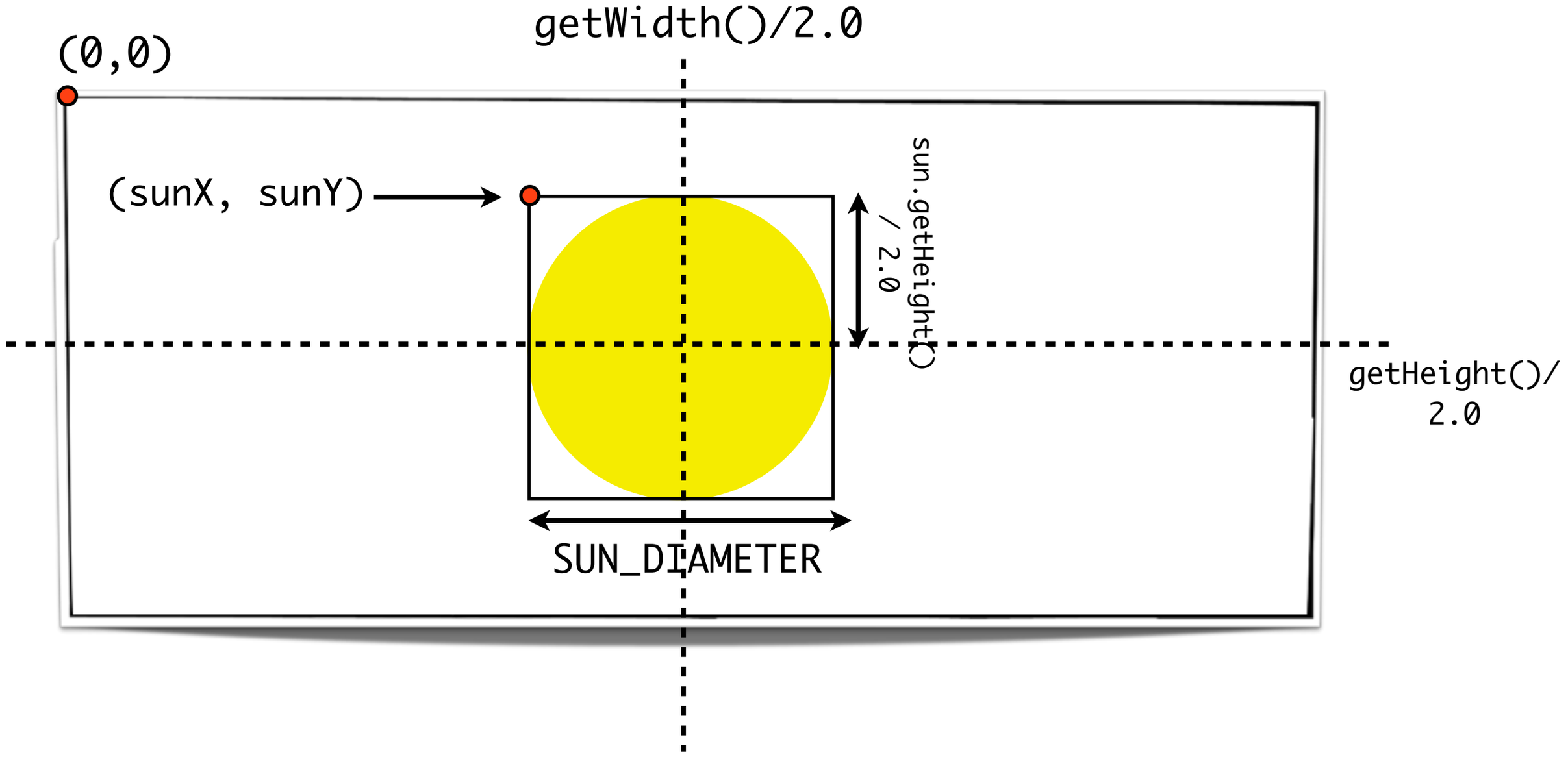
```
private void drawSun() {
  GOval sun = new GOval(SUN_DIAMETER, SUN_DIAMETER);
  sun.setColor(Color.YELLOW);
  sun.setFilled(true);
  sun.setFillColor(Color.YELLOW);
  double sunX = ____________________________________;
  double sunY = ____________________________________;
  add(sun, sunX, sunY);
}
```



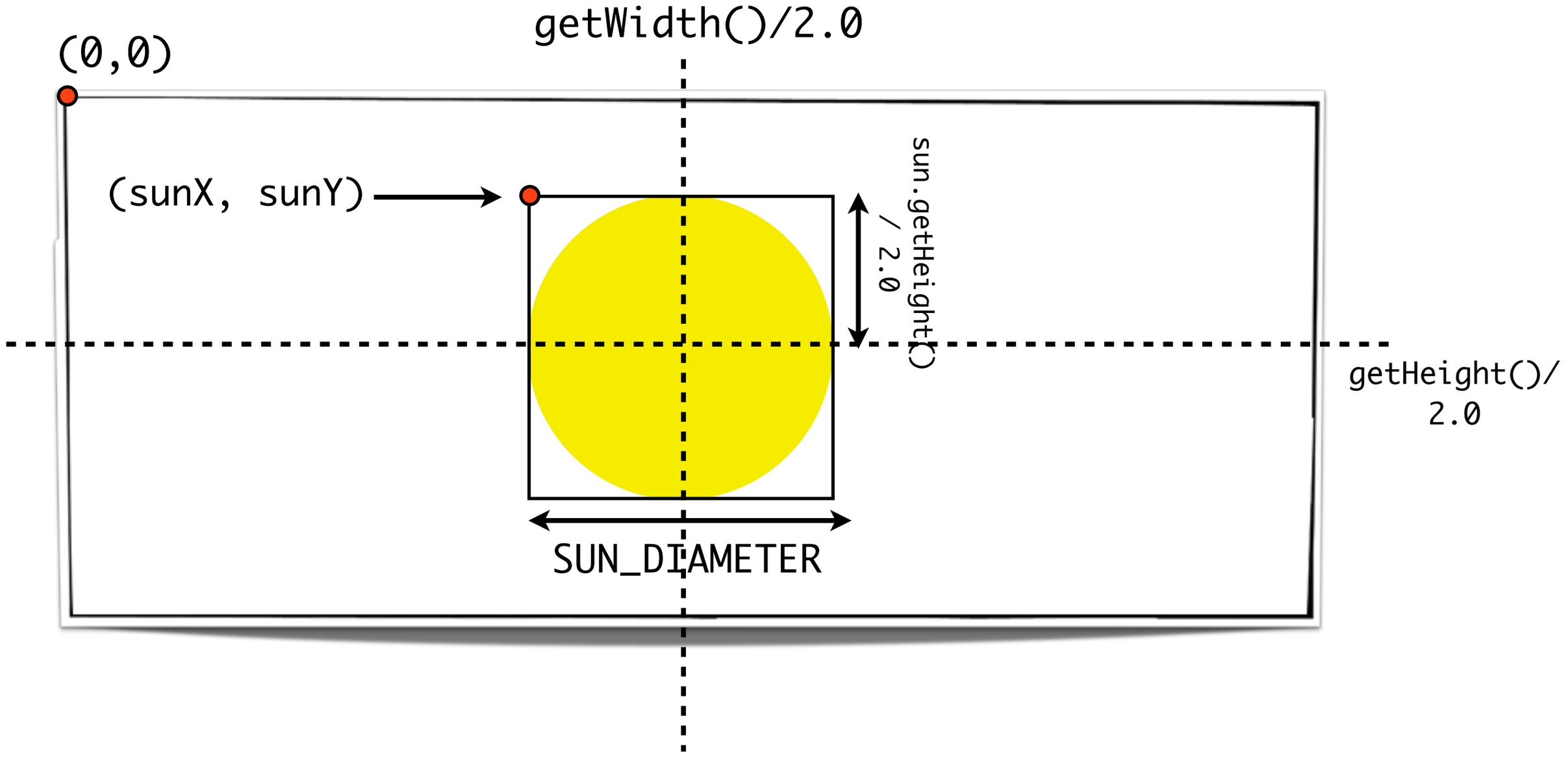
```
private void drawSun() {
  GOval sun = new GOval(SUN_DIAMETER, SUN_DIAMETER);
  sun.setColor(Color.YELLOW);
  sun.setFilled(true);
  sun.setFillColor(Color.YELLOW);
  double sunX = ____________________________________;
  double sunY = ____________________________________;
  add(sun, sunX, sunY);
}
```



```
private void drawSun() {
  GOval sun = new GOval(SUN_DIAMETER, SUN_DIAMETER);
  sun.setColor(Color.YELLOW);
  sun.setFilled(true);
  sun.setFillColor(Color.YELLOW);
  double sunX = getWidth()/2.0 - sun.getWidth()/2.0;
  double sunY = ____________________________________;
  add(sun, sunX, sunY);
}
```

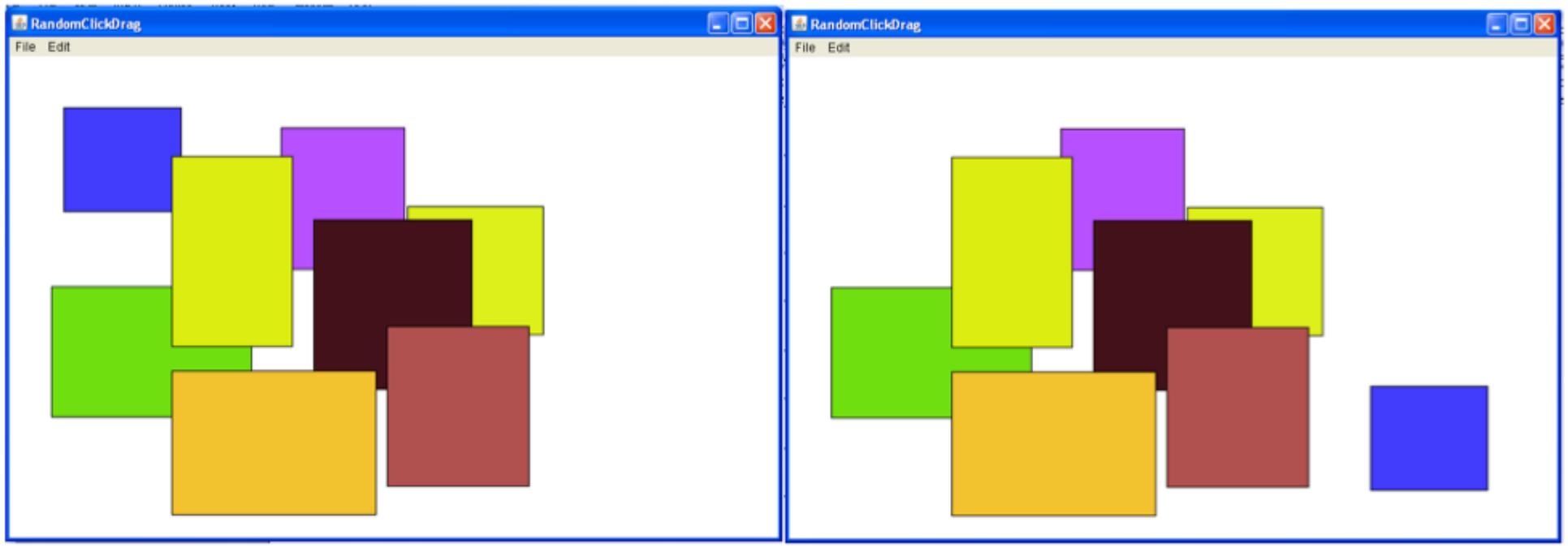


```
private void drawSun() {
  GOval sun = new GOval(SUN_DIAMETER, SUN_DIAMETER);
  sun.setColor(Color.YELLOW);
  sun.setFilled(true);
  sun.setFillColor(Color.YELLOW);
  double sunX = getWidth()/2.0 - sun.getWidth()/2.0;
  double sunY = getHeight()/2.0 - sun.getHeight()/2.0;
  add(sun, sunX, sunY);
}
```



RandomClickDrag

In this problem you will write a program that draws GRects of random sizes on the screen at the click of the mouse and allows the user to drag them around the canvas. The GRects can change colors and be dragged around the screen. For example, after a few clicks, your canvas will look as shown below at the left. However, if you press the mouse down on top of the square at the top left, hold, and move it to the bottom right corner, the square will move and the image will now look as shown below at the right.



- Length and width are randomly chosen, as well as the initial color
- Clicking on an existing piece will simply choose a new color for that GRect
- Size is bounded by MAX_LEN and MIN_LEN

mouseListeners

```
public void mouseMoved(MouseEvent e)
public void mouseDragged(MouseEvent e)
public void mousePressed(MouseEvent e)
public void mouseReleased(MouseEvent e)
public void mouseClicked(MouseEvent e)
public void mouseEntered(MouseEvent e)
public void mouseExited(MouseEvent e)
```

```
public class RandomClickDrag extends GraphicsProgram {
    private static final double MAX_LEN = 200;
    private static final double MIN_LEN = 100;




    public void run() {

    }
```

```
}
```

```
public class RandomClickDrag extends GraphicsProgram {
    private static final double MAX_LEN = 200;
    private static final double MIN_LEN = 100;




    public void run() {
        addMouseListeners();
    }
```

```
}
```

```
public class RandomClickDrag extends GraphicsProgram {
    private static final double MAX_LEN = 200;
    private static final double MIN_LEN = 100;



    public void run() {
        addMouseListeners();
    }
```

```
public void mouseMoved(MouseEvent e)
public void mouseDragged(MouseEvent e)
public void mousePressed(MouseEvent e)
public void mouseReleased(MouseEvent e)
public void mouseClicked(MouseEvent e)
public void mouseEntered(MouseEvent e)
public void mouseExited(MouseEvent e)
```

```
}
```

```
public class RandomClickDrag extends GraphicsProgram {
    private static final double MAX_LEN = 200;
    private static final double MIN_LEN = 100;




    public void run() {
        addMouseListeners();
    }

    public void mouseClicked(MouseEvent e) {

    }
.
.
}
```

```
public void mouseMoved(MouseEvent e)
public void mouseDragged(MouseEvent e)
public void mousePressed(MouseEvent e)
public void mouseReleased(MouseEvent e)
public void mouseClicked(MouseEvent e)
public void mouseEntered(MouseEvent e)
public void mouseExited(MouseEvent e)
```

```
public class RandomClickDrag extends GraphicsProgram {
    private static final double MAX_LEN = 200;
    private static final double MIN_LEN = 100;




    public void run() {
        addMouseListeners();
    }

    public void mouseClicked(MouseEvent e) {




    }

    public void mousePressed(MouseEvent e) {



    }
```

```
public void mouseMoved(MouseEvent e)
public void mouseDragged(MouseEvent e)
public void mousePressed(MouseEvent e)
public void mouseReleased(MouseEvent e)
public void mouseClicked(MouseEvent e)
public void mouseEntered(MouseEvent e)
public void mouseExited(MouseEvent e)
```

```
}
```

```
public class RandomClickDrag extends GraphicsProgram {
    private static final double MAX_LEN = 200;
    private static final double MIN_LEN = 100;




    public void run() {
        addMouseListeners();
    }

    public void mouseClicked(MouseEvent e) {


    }
    public void mousePressed(MouseEvent e) {


    }
    public void mouseDragged(MouseEvent e) {



    }
}
```

```
public void mouseMoved(MouseEvent e)
public void mouseDragged(MouseEvent e)
public void mousePressed(MouseEvent e)
public void mouseReleased(MouseEvent e)
public void mouseClicked(MouseEvent e)
public void mouseEntered(MouseEvent e)
public void mouseExited(MouseEvent e)
```

```
public class RandomClickDrag extends GraphicsProgram {
    private static final double MAX_LEN = 200;
    private static final double MIN_LEN = 100;









    public void run() {
        addMouseListeners();
    }

    public void mouseClicked(MouseEvent e) {
```

```
    }
    public void mousePressed(MouseEvent e) {


    }
    public void mouseDragged(MouseEvent e) {


    }
}
```

```
public class RandomClickDrag extends GraphicsProgram {
    private static final double MAX_LEN = 200;
    private static final double MIN_LEN = 100;









    public void run() {
        addMouseListeners();
    }

    public void mouseClicked(MouseEvent e) {
        GRect clickedRect = (GRect) getElementAt(e.getX(), e.getY());
```

```
    }
    public void mousePressed(MouseEvent e) {


    }
    public void mouseDragged(MouseEvent e) {




    }
}
```

```
public class RandomClickDrag extends GraphicsProgram {
    private static final double MAX_LEN = 200;
    private static final double MIN_LEN = 100;









    public void run() {
        addMouseListeners();
    }

    public void mouseClicked(MouseEvent e) {
        GRect clickedRect = (GRect) getElementAt(e.getX(), e.getY());

        if (clickedRect != null) {

        } else {



        }
    }

    public void mousePressed(MouseEvent e) {



    }

    public void mouseDragged(MouseEvent e) {









    }
}
```

```
public class RandomClickDrag extends GraphicsProgram {
    private static final double MAX_LEN = 200;
    private static final double MIN_LEN = 100;









    public void run() {
        addMouseListeners();
    }

    public void mouseClicked(MouseEvent e) {
        GRect clickedRect = (GRect) getElementAt(e.getX(), e.getY());

        if (clickedRect != null) {
            clickedRect.setFillColor(rgen.nextColor());
        } else {



        }
    }

    public void mousePressed(MouseEvent e) {



    }

    public void mouseDragged(MouseEvent e) {









    }
}
```

```
public class RandomClickDrag extends GraphicsProgram {
    private static final double MAX_LEN = 200;
    private static final double MIN_LEN = 100;

    private RandomGenerator rgen = new RandomGenerator();




    public void run() {
        addMouseListeners();
    }

    public void mouseClicked(MouseEvent e) {
        GRect clickedRect = (GRect) getElementAt(e.getX(), e.getY());

        if (clickedRect != null) {
            clickedRect.setFillColor(rgen.nextColor());
        } else {



        }
    }

    public void mousePressed(MouseEvent e) {



    }

    public void mouseDragged(MouseEvent e) {









    }
}
```

```
public class RandomClickDrag extends GraphicsProgram {
    private static final double MAX_LEN = 200;
    private static final double MIN_LEN = 100;

    private RandomGenerator rgen = new RandomGenerator();




    public void run() {
        addMouseListeners();
    }

    public void mouseClicked(MouseEvent e) {
        GRect clickedRect = (GRect) getElementAt(e.getX(), e.getY());

        if (clickedRect != null) {
            clickedRect.setFillColor(rgen.nextColor());
        } else {
            GRect rect = new GRect(rgen.nextDouble(MIN_LEN, MAX_LEN),
                rgen.nextDouble(MIN_LEN, MAX_LEN));



        }
    }
    public void mousePressed(MouseEvent e) {



    }
    public void mouseDragged(MouseEvent e) {









    }
}
```

```
public class RandomClickDrag extends GraphicsProgram {
    private static final double MAX_LEN = 200;
    private static final double MIN_LEN = 100;

    private RandomGenerator rgen = new RandomGenerator();




    public void run() {
        addMouseListeners();
    }

    public void mouseClicked(MouseEvent e) {
        GRect clickedRect = (GRect) getElementAt(e.getX(), e.getY());

        if (clickedRect != null) {
            clickedRect.setFillColor(rgen.nextColor());
        } else {
            GRect rect = new GRect(rgen.nextDouble(MIN_LEN, MAX_LEN),
                rgen.nextDouble(MIN_LEN, MAX_LEN));
            rect.setLocation(e.getX() - (rect.getWidth() / 2),
                    e.getY() - (rect.getHeight() / 2));



        }
    }

    public void mousePressed(MouseEvent e) {



    }

    public void mouseDragged(MouseEvent e) {



    }
}
```

```
public class RandomClickDrag extends GraphicsProgram {
    private static final double MAX_LEN = 200;
    private static final double MIN_LEN = 100;

    private RandomGenerator rgen = new RandomGenerator();




    public void run() {
        addMouseListeners();
    }

    public void mouseClicked(MouseEvent e) {
        GRect clickedRect = (GRect) getElementAt(e.getX(), e.getY());

        if (clickedRect != null) {
            clickedRect.setFillColor(rgen.nextColor());
        } else {
            GRect rect = new GRect(rgen.nextDouble(MIN_LEN, MAX_LEN),
                rgen.nextDouble(MIN_LEN, MAX_LEN));
            rect.setLocation(e.getX() - (rect.getWidth() / 2),
                    e.getY() - (rect.getHeight() / 2));
            rect.setFilled(true);
            rect.setFillColor(rgen.nextColor());

        }
    }

    public void mousePressed(MouseEvent e) {



    }

    public void mouseDragged(MouseEvent e) {









    }

}
```

```
public class RandomClickDrag extends GraphicsProgram {
    private static final double MAX_LEN = 200;
    private static final double MIN_LEN = 100;

    private RandomGenerator rgen = new RandomGenerator();




    public void run() {
        addMouseListeners();
    }

    public void mouseClicked(MouseEvent e) {
        GRect clickedRect = (GRect) getElementAt(e.getX(), e.getY());

        if (clickedRect != null) {
            clickedRect.setFillColor(rgen.nextColor());
        } else {
            GRect rect = new GRect(rgen.nextDouble(MIN_LEN, MAX_LEN),
                rgen.nextDouble(MIN_LEN, MAX_LEN));
            rect.setLocation(e.getX() - (rect.getWidth() / 2),
                    e.getY() - (rect.getHeight() / 2));
            rect.setFilled(true);
            rect.setFillColor(rgen.nextColor());
            add(rect);
        }
    }

    public void mousePressed(MouseEvent e) {



    }

    public void mouseDragged(MouseEvent e) {

    }
}
```

```
public class RandomClickDrag extends GraphicsProgram {
     private static final double MAX_LEN = 200;
     private static final double MIN_LEN = 100;

     private RandomGenerator rgen = new RandomGenerator();




     public void run() {
          addMouseListeners();
     }

     public void mouseClicked(MouseEvent e) {
          GRect clickedRect = (GRect) getElementAt(e.getX(), e.getY());

          if (clickedRect != null) {
               clickedRect.setFillColor(rgen.nextColor());
          } else {
               GRect rect = new GRect(rgen.nextDouble(MIN_LEN, MAX_LEN),
                    rgen.nextDouble(MIN_LEN, MAX_LEN));
               rect.setLocation(e.getX() - (rect.getWidth() / 2),
                         e.getY() - (rect.getHeight() / 2));
               rect.setFilled(true);
               rect.setFillColor(rgen.nextColor());
               add(rect);
          }
     }

     public void mousePressed(MouseEvent e) {
          currentRect = (GRect) getElementAt(e.getX(), e.getY());


     }

     public void mouseDragged(MouseEvent e) {




     }

}
```

```
public class RandomClickDrag extends GraphicsProgram {
     private static final double MAX_LEN = 200;
     private static final double MIN_LEN = 100;

     private RandomGenerator rgen = new RandomGenerator();
     private GRect currentRect;



     public void run() {
          addMouseListeners();
     }

     public void mouseClicked(MouseEvent e) {
          GRect clickedRect = (GRect) getElementAt(e.getX(), e.getY());

          if (clickedRect != null) {
               clickedRect.setFillColor(rgen.nextColor());
          } else {
               GRect rect = new GRect(rgen.nextDouble(MIN_LEN, MAX_LEN),
                    rgen.nextDouble(MIN_LEN, MAX_LEN));
               rect.setLocation(e.getX() - (rect.getWidth() / 2),
                         e.getY() - (rect.getHeight() / 2));
               rect.setFilled(true);
               rect.setFillColor(rgen.nextColor());
               add(rect);
          }
     }

     public void mousePressed(MouseEvent e) {
          currentRect = (GRect) getElementAt(e.getX(), e.getY());


     }

     public void mouseDragged(MouseEvent e) {



     }

}
```

```
public class RandomClickDrag extends GraphicsProgram {
     private static final double MAX_LEN = 200;
     private static final double MIN_LEN = 100;

     private RandomGenerator rgen = new RandomGenerator();
     private GRect currentRect;



     public void run() {
          addMouseListeners();
     }

     public void mouseClicked(MouseEvent e) {
          GRect clickedRect = (GRect) getElementAt(e.getX(), e.getY());

          if (clickedRect != null) {
               clickedRect.setFillColor(rgen.nextColor());
          } else {
               GRect rect = new GRect(rgen.nextDouble(MIN_LEN, MAX_LEN),
                    rgen.nextDouble(MIN_LEN, MAX_LEN));
               rect.setLocation(e.getX() - (rect.getWidth() / 2),
                         e.getY() - (rect.getHeight() / 2));
               rect.setFilled(true);
               rect.setFillColor(rgen.nextColor());
               add(rect);
          }
     }

     public void mousePressed(MouseEvent e) {
          currentRect = (GRect) getElementAt(e.getX(), e.getY());
          lastX = e.getX();
          lastY = e.getY();
     }

     public void mouseDragged(MouseEvent e) {



     }

}
```

```
public class RandomClickDrag extends GraphicsProgram {
    private static final double MAX_LEN = 200;
    private static final double MIN_LEN = 100;

    private RandomGenerator rgen = new RandomGenerator();
    private GRect currentRect;
    private double lastX;
    private double lastY;

    public void run() {
        addMouseListeners();
    }

    public void mouseClicked(MouseEvent e) {
        GRect clickedRect = (GRect) getElementAt(e.getX(), e.getY());

        if (clickedRect != null) {
            clickedRect.setFillColor(rgen.nextColor());
        } else {
            GRect rect = new GRect(rgen.nextDouble(MIN_LEN, MAX_LEN),
                rgen.nextDouble(MIN_LEN, MAX_LEN));
            rect.setLocation(e.getX() - (rect.getWidth() / 2),
                    e.getY() - (rect.getHeight() / 2));
            rect.setFilled(true);
            rect.setFillColor(rgen.nextColor());
            add(rect);
        }
    }

    public void mousePressed(MouseEvent e) {
        currentRect = (GRect) getElementAt(e.getX(), e.getY());
        lastX = e.getX();
        lastY = e.getY();
    }

    public void mouseDragged(MouseEvent e) {



    }

}
```

```
public class RandomClickDrag extends GraphicsProgram {
    private static final double MAX_LEN = 200;
    private static final double MIN_LEN = 100;

    private RandomGenerator rgen = new RandomGenerator();
    private GRect currentRect;
    private double lastX;
    private double lastY;

    public void run() {
        addMouseListeners();
    }

    public void mouseClicked(MouseEvent e) {
        GRect clickedRect = (GRect) getElementAt(e.getX(), e.getY());

        if (clickedRect != null) {
            clickedRect.setFillColor(rgen.nextColor());
        } else {
            GRect rect = new GRect(rgen.nextDouble(MIN_LEN, MAX_LEN),
                rgen.nextDouble(MIN_LEN, MAX_LEN));
            rect.setLocation(e.getX() - (rect.getWidth() / 2),
                    e.getY() - (rect.getHeight() / 2));
            rect.setFilled(true);
            rect.setFillColor(rgen.nextColor());
            add(rect);
        }
    }

    public void mousePressed(MouseEvent e) {
        currentRect = (GRect) getElementAt(e.getX(), e.getY());
        lastX = e.getX();
        lastY = e.getY();
    }

    public void mouseDragged(MouseEvent e) {
        if (currentRect != null) {



        }
    }

}
```

```
public class RandomClickDrag extends GraphicsProgram {
    private static final double MAX_LEN = 200;
    private static final double MIN_LEN = 100;

    private RandomGenerator rgen = new RandomGenerator();
    private GRect currentRect;
    private double lastX;
    private double lastY;

    public void run() {
        addMouseListeners();
    }

    public void mouseClicked(MouseEvent e) {
        GRect clickedRect = (GRect) getElementAt(e.getX(), e.getY());

        if (clickedRect != null) {
            clickedRect.setFillColor(rgen.nextColor());
        } else {
            GRect rect = new GRect(rgen.nextDouble(MIN_LEN, MAX_LEN),
                rgen.nextDouble(MIN_LEN, MAX_LEN));
            rect.setLocation(e.getX() - (rect.getWidth() / 2),
                    e.getY() - (rect.getHeight() / 2));
            rect.setFilled(true);
            rect.setFillColor(rgen.nextColor());
            add(rect);
        }
    }

    public void mousePressed(MouseEvent e) {
        currentRect = (GRect) getElementAt(e.getX(), e.getY());
        lastX = e.getX();
        lastY = e.getY();
    }

    public void mouseDragged(MouseEvent e) {
        if (currentRect != null) {
            currentRect.move(e.getX() - lastX, e.getY() - lastY);



        }
    }

}
```

```
public class RandomClickDrag extends GraphicsProgram {
    private static final double MAX_LEN = 200;
    private static final double MIN_LEN = 100;

    private RandomGenerator rgen = new RandomGenerator();
    private GRect currentRect;
    private double lastX;
    private double lastY;

    public void run() {
        addMouseListeners();
    }

    public void mouseClicked(MouseEvent e) {
        GRect clickedRect = (GRect) getElementAt(e.getX(), e.getY());

        if (clickedRect != null) {
            clickedRect.setFillColor(rgen.nextColor());
        } else {
            GRect rect = new GRect(rgen.nextDouble(MIN_LEN, MAX_LEN),
                rgen.nextDouble(MIN_LEN, MAX_LEN));
            rect.setLocation(e.getX() - (rect.getWidth() / 2),
                    e.getY() - (rect.getHeight() / 2));
            rect.setFilled(true);
            rect.setFillColor(rgen.nextColor());
            add(rect);
        }
    }

    public void mousePressed(MouseEvent e) {
        currentRect = (GRect) getElementAt(e.getX(), e.getY());
        lastX = e.getX();
        lastY = e.getY();
    }

    public void mouseDragged(MouseEvent e) {
        if (currentRect != null) {
            currentRect.move(e.getX() - lastX, e.getY() - lastY);

            lastX = e.getX();
            lastY = e.getY();
        }
    }

}
```

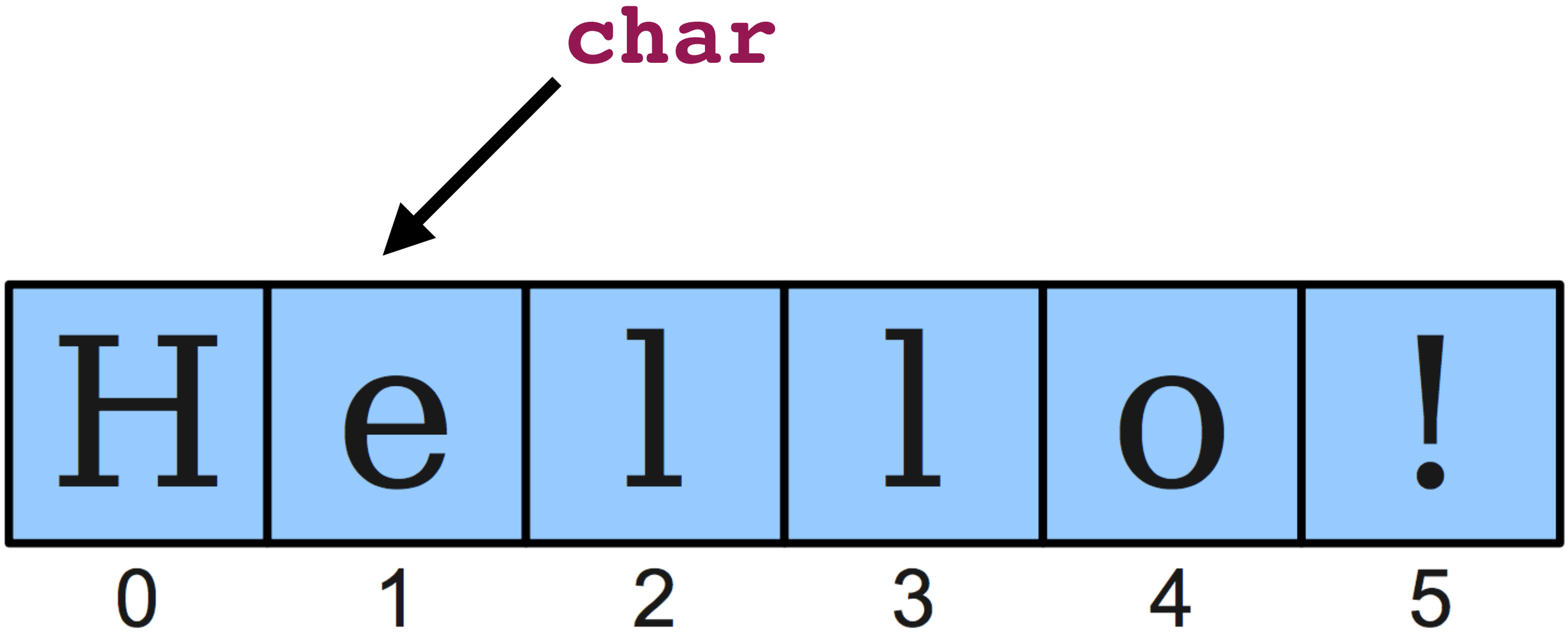
The plan

- General info
- Karel
- Expression evaluation
- Program tracing
- Simple Java and randomness
- Graphics and MouseEvents
- String manipulation

H	e	l	l	o	!
0	1	2	3	4	5

H	e	l	l	o	!
0	1	2	3	4	5

string`.charAt(`*index*`)`



***string*`.charAt(`*index*`)`**

Testing properties of `char`

`boolean Character.isDigit(char ch)` Determines if the specified character is a digit.
`boolean Character.isLetter(char ch)` Determines if the specified character is a letter.
`boolean Character.isLetterOrDigit(char ch)` Determines if the specified character is a letter or a digit.
`boolean Character.isLowerCase(char ch)` Determines if the specified character is a lowercase letter.
`boolean Character.isUpperCase(char ch)` Determines if the specified character is an uppercase letter.
`boolean Character.isWhitespace(char ch)` Determines if the specified character is **whitespace** (spaces and tabs).
`char Character.toLowerCase(char ch)` Converts **`ch`** to its lowercase equivalent, if any. If not, **`ch`** is returned unchanged.
`char Character.toUpperCase(char ch)` Converts **`ch`** to its uppercase equivalent, if any. If not, **`ch`** is returned unchanged.

```
char ch =
Character.toUpperCase('a');
```

```
char ch =
Character.toUpperCase('a');

// ch is 'A'
```

```
char ch =
Character.toUpperCase('a');

// ch is 'A'
```

```
boolean bool =
Character.isLetter('!');
```

```
char ch =
Character.toUpperCase('a');

// ch is 'A'
```

```
boolean bool =
Character.isLetter('!');

// bool is false
```

Useful `String` methods

`int length()` Returns the length of the string
`char charAt(int index)` Returns the character at the specified index. Note: Strings indexed starting at 0.
`String substring(int p1, int p2)` Returns the substring beginning at **p1** and extending up to but not including **p2**
`String substring(int p1)` Returns substring beginning at **p1** and extending through end of string.
`boolean equals(String s2)` Returns true if string **s2** is equal to the receiver string. This is case sensitive.
`int compareTo(String s2)` Returns integer whose sign indicates how strings compare in lexicographic order
`int indexOf(char ch)` *or* **`int indexOf(String s)`** Returns index of first occurrence of the character or the string, or -1 if not found
`String toLowerCase()` *or* **`String toUpperCase()`** Returns a lowercase or uppercase version of the receiver string

```
String testStr = “hello”;
testStr = testStr + “!”;
println(testStr);
```

```
String testStr = “hello”;
testStr = testStr + “!”;
println(testStr);
// output is “hello!”
```

```
String testStr = “hello”;
testStr += “!”;
println(testStr);
```

```
String testStr = “hello”;
testStr += “!”;
println(testStr);
// output is still “hello!”
```

```
String testStr = “hello”;
testStr += “!”;
println(testStr);
// output is still “hello!”
```

```
String testStr = “hello”;
testStr = “!” + testStr;
println(testStr);
```

```
String testStr = “hello”;
testStr += “!”;
println(testStr);
// output is still “hello!”
```

```
String testStr = “hello”;
testStr = “!” + testStr;
println(testStr);
// output is “!hello”
```

A warm-up

```
public void run() {
    String str = "1 3 535 2340 29";
    println(removeWhitespace1(str));
    println(removeWhitespace2(str));
}
```

```
private String removeWhitespace1(String str) {



}
```

A warm-up

```
public void run() {
    String str = "1 3 535 2340 29";
    println(removeWhitespace1(str));
    println(removeWhitespace2(str));
}
```

```
private String removeWhitespace1(String str) {
    String result = "";




    return result;
}
```

A warm-up

```
public void run() {
    String str = "1 3 535 2340 29";
    println(removeWhitespace1(str));
    println(removeWhitespace2(str));
}
```

```
private String removeWhitespace1(String str) {
    String result = "";
    for (int i = 0; i < str.length(); i++) {


            result += curCh;


    }
    return result;
}
```

A warm-up

```
public void run() {
    String str = "1 3 535 2340 29";
    println(removeWhitespace1(str));
    println(removeWhitespace2(str));
}
```

```
private String removeWhitespace1(String str) {
    String result = "";
    for (int i = 0; i < str.length(); i++) {
        char curCh = str.charAt(i);

            result += curCh;

    }
    return result;
}
```

A warm-up

```
public void run() {
    String str = "1 3 535 2340 29";
    println(removeWhitespace1(str));
    println(removeWhitespace2(str));
}
```

```
private String removeWhitespace1(String str) {
    String result = "";
    for (int i = 0; i < str.length(); i++) {
        char curCh = str.charAt(i);
        if (!Character.isWhitespace(curCh)) {
            result += curCh;
        }
    }
    return result;
}
```

```java
private String removeWhitespace2(String str) {
    String result = "";
    for (int i = str.length() - 1; i >= 0; i--) {
        char curCh = str.charAt(i);
        if (!Character.isWhitespace(curCh)) {
            result = curCh + result;
        }
    }
    return result;
}
```

```java
private String removeWhitespace1(String str) {
    String result = "";
    for (int i = 0; i < str.length(); i++) {
        char curCh = str.charAt(i);
        if (!Character.isWhitespace(curCh)) {
            result += curCh;
        }
    }
    return result;
}
```

```java
private String removeWhitespace2(String str) {
    String result = "";
    for (int i = str.length() - 1; i >= 0; i--) {
        char curCh = str.charAt(i);
        if (!Character.isWhitespace(curCh)) {
            result = curCh + result;
        }
    }
    return result;
}
```

```java
private String removeWhitespace1(String str) {
    String result = "";
    for (int i = 0; i < str.length(); i++) {
        char curCh = str.charAt(i);
        if (!Character.isWhitespace(curCh)) {
            result += curCh;
        }
    }
    return result;
}
```

More `String` problem tips

More `String` problem tips

- First consider how you would do the problem as a human

More `String` problem tips

- First consider how you would do the problem as a human
  - What do you need to remember between each character?

More `String` problem tips

- First consider how you would do the problem as a human
    - What do you need to remember between each character?
- Watch out for out-of-bounds!

More `String` problem tips

- First consider how you would do the problem as a human
    - What do you need to remember between each character?
- Watch out for out-of-bounds!
- Decompose the problem if necessary

Write a method **`removeDoubledLetters`** that takes a string as its argument and returns a new string with all doubled letters in the string replaced by a single letter. For example, if you call

```
removeDoubledLetters("tresidder")
```

your method should return the string **`"tresider"`**. Similarly, if you call

```
removeDoubledLetters("bookkeeper")
```

your method should return **`"bokeper"`**.

In writing your solution, you should keep in mind the following:

- You do not need to write a complete program. All you need is the definition of the method **`removeDoubledLetters`** that returns the desired result.
- You may assume that all letters in the string are lower case so that you don't have to worry about changes in capitalization.
- You may assume that no letter appears more than twice in a row. (It is likely that your program will work even if this restriction were not included; we've included it explicitly only so that you don't even have to think about this case.)

```
/**
 * Removes any doubled letters from a string.
 */
   private String removeDoubledLetters(String str)  {

}
```

```
/**
 * Removes any doubled letters from a string.
 */
   private String removeDoubledLetters(String str) {
      String result = "";

      return result;
   }
```

```
/**
 * Removes any doubled letters from a string.
 */
    private String removeDoubledLetters(String str) {
        String result = "";
        for (int i = 0; i < str.length(); i++) {
            
            
            
        }
        return result;
    }
```

```
/**
 * Removes any doubled letters from a string.
 */
   private String removeDoubledLetters(String str) {
      String result = "";
      for (int i = 0; i < str.length(); i++) {
         char ch = str.charAt(i);



      }
      return result;
   }
```

```
/**
 * Removes any doubled letters from a string.
 */
   private String removeDoubledLetters(String str) {
      String result = "";
      for (int i = 0; i < str.length(); i++) {
         char ch = str.charAt(i);
         if (i == 0 || ch != str.charAt(i - 1)) {

         }
      }
      return result;
   }
```

```
/**
 * Removes any doubled letters from a string.
 */
   private String removeDoubledLetters(String str) {
      String result = "";
      for (int i = 0; i < str.length(); i++) {
         char ch = str.charAt(i);
         if (i == 0 || ch != str.charAt(i - 1)) {
            result += ch;
         }
      }
      return result;
   }
```

How do I love thee? Let me count the ways.

How do I love thee? Let me count the ways.

Elizabeth Browning, *Sonnet 43*

How do I love thee? Let me count the ways.

Elizabeth Browning, *Sonnet 43* 1850

Computers are, of course, very good at counting things. Write a method

```
private int countLove(String str)
```

that takes a string as its argument and returns the number of times the word *love* appears in that string, ignoring differences in case, but making sure that *love* is not just part of a longer word like *clover, glove, pullover,* or *slovenly*. For example, if you were to call

```
countLove("Love in the clover.")
```

your method should return 1. The word **`Love`** counts as a match because your method should ignore the fact that **`Love`** starts with an uppercase **`L`**. The word **`clover`** doesn't match because the letters **`love`** are merely part of a larger word.

`int length()`
Returns the length of the string

`char charAt(int index)`
Returns the character at the specified index. Note: Strings indexed starting at 0.

`String substring(int p1, int p2)`
Returns the substring beginning at **p1** and extending up to but not including **p2**

`String substring(int p1)`
Returns substring beginning at **p1** and extending through end of string.

`boolean equals(String s2)`
Returns true if string **s2** is equal to the receiver string. This is case sensitive.

`int compareTo(String s2)`
Returns integer whose sign indicates how strings compare in lexicographic order

`int indexOf(char ch)` *or* **`int indexOf(String s)`**
Returns index of first occurrence of the character or the string, or -1 if not found

`String toLowerCase()` *or* **`String toUpperCase()`**
Returns a lowercase or uppercase version of the receiver string

```java
/**
 * Counts the occurrences of the word "love"
 * in a string.
 */
   private int countLove(String str) {

}
```

```java
/**
 * Counts the occurrences of the word "love"
 * in a string.
 */
   private int countLove(String str) {
       String lowerCaseString = str.toLowerCase();
```

```java
}
```

```java
/**
 * Counts the occurrences of the word "love"
 * in a string.
 */
   private int countLove(String str) {
       String lowerCaseString = str.toLowerCase();

       int start = lowerCaseString.indexOf("love");
```

```java
}
```

```java
/**
 * Counts the occurrences of the word "love"
 * in a string.
 */
   private int countLove(String str) {
       String lowerCaseString = str.toLowerCase();

       int start = lowerCaseString.indexOf("love");
       while (start != -1) {

       }

}
```

```java
/**
 * Counts the occurrences of the word "love"
 * in a string.
 */
   private int countLove(String str) {
      String lowerCaseString = str.toLowerCase();

      int start = lowerCaseString.indexOf("love");
      while (start != -1) {

         start = lowerCaseString.indexOf("love", start + 4);
      }

}
```

```
/**
 * Counts the occurrences of the word "love"
 * in a string.
 */
   private int countLove(String str) {
      String lowerCaseString = str.toLowerCase();

      int start = lowerCaseString.indexOf("love");
      while (start != -1) {
         if (isSeparator(lowerCaseString, start - 1) &&
             isSeparator(lowerCaseString, start + 4)) {

         }
         start = lowerCaseString.indexOf("love", start + 4);
      }

   }
```

```
/**
 * Counts the occurrences of the word "love"
 * in a string.
 */
   private int countLove(String str) {
       String lowerCaseString = str.toLowerCase();
       int count = 0;
       int start = lowerCaseString.indexOf("love");
       while (start != -1) {
           if (isSeparator(lowerCaseString, start - 1) &&
               isSeparator(lowerCaseString, start + 4)) {

           }
           start = lowerCaseString.indexOf("love", start + 4);
       }

   }
```

```
/**
 * Counts the occurrences of the word "love"
 * in a string.
 */
   private int countLove(String str) {
       String lowerCaseString = str.toLowerCase();
       int count = 0;
       int start = lowerCaseString.indexOf("love");
       while (start != -1) {
           if (isSeparator(lowerCaseString, start - 1) &&
               isSeparator(lowerCaseString, start + 4)) {
              count++;
           }
           start = lowerCaseString.indexOf("love", start + 4);
       }

   }
```

```
/**
 * Counts the occurrences of the word "love"
 * in a string.
 */
   private int countLove(String str) {
       String lowerCaseString = str.toLowerCase();
       int count = 0;
       int start = lowerCaseString.indexOf("love");
       while (start != -1) {
           if (isSeparator(lowerCaseString, start - 1) &&
               isSeparator(lowerCaseString, start + 4)) {
              count++;
           }
           start = lowerCaseString.indexOf("love", start + 4);
       }
       return count;
   }
```

```java
/**
 * Counts the occurrences of the word "love"
 * in a string.
 */
   private int countLove(String str) {
       String lowerCaseString = str.toLowerCase();
       int count = 0;
       int start = lowerCaseString.indexOf("love");
       while (start != -1) {
           if (isSeparator(lowerCaseString, start - 1) &&
                isSeparator(lowerCaseString, start + 4)) {
               count++;
           }
           start = lowerCaseString.indexOf("love", start + 4);
       }
       return count;
   }

/**
 * Checks to see if the ith char of str is a separator.
 */
   private boolean isSeparator(String str, int i) {


   }
```

```
/**
 * Counts the occurrences of the word "love"
 * in a string.
 */
   private int countLove(String str) {
       String lowerCaseString = str.toLowerCase();
       int count = 0;
       int start = lowerCaseString.indexOf("love");
       while (start != -1) {
           if (isSeparator(lowerCaseString, start - 1) &&
               isSeparator(lowerCaseString, start + 4)) {
              count++;
           }
           start = lowerCaseString.indexOf("love", start + 4);
       }
       return count;
   }

/**
 * Checks to see if the ith char of str is a separator.
 */
   private boolean isSeparator(String str, int i) {

       return !Character.isLetter(str.charAt(i));
   }
```

```java
/**
 * Counts the occurrences of the word "love"
 * in a string.
 */
   private int countLove(String str) {
      String lowerCaseString = str.toLowerCase();
      int count = 0;
      int start = lowerCaseString.indexOf("love");
      while (start != -1) {
         if (isSeparator(lowerCaseString, start - 1) &&
             isSeparator(lowerCaseString, start + 4)) {
            count++;
         }
         start = lowerCaseString.indexOf("love", start + 4);
      }
      return count;
   }

/**
 * Checks to see if the ith char of str is a separator.
 */
   private boolean isSeparator(String str, int i) {
      if (i < 0 || i >= str.length()) return true;
      return !Character.isLetter(str.charAt(i));
   }
```

Wish You a Good Luck