

Animation

An Interesting Website

www.boxcar2d.com

Returning Values

- A method may produce a value that can be read by its caller.
- To indicate that a method returns a value, specify the type returned in the method declaration:

```
private type name (parameters) {  
    /* ... method body ... */  
}
```

- A value can be returned with the **return** statement:

```
return value;
```

Subtleties of `return`

- If a method has non-`void` return type, it must always return a value.

```
private int thisIsWrong(int x) {  
    if (x == 5) {  
        return 0;  
    }  
}
```

What do we
return if `x != 5`?

Subtleties of `return`

- If a method has non-`void` return type, it must always return a value.

```
private int thisIsLegal(int x) {  
    if (x == 5) {  
        return 0;  
    } else {  
        return 1;  
    }  
}
```

Many Happy `return`s

- A method may have multiple return statements. The method ends as soon as `return` is executed.

```
private int thisIsLegal(int x) {  
    if (x == 5) {  
        return 0;  
    } else {  
        return 1;  
    }  
}
```

Many Happy `return`s

- A method may have multiple return statements. The method ends as soon as `return` is executed.

```
private int thisIsLegal(int x) {  
    if (x == 5) {  
        return 0;  
    }  
    return 1;  
}
```

The only way we can get here is if x is not equal to 5.

Scope

- Each variable has a **scope** where it can be accessed and how long it lives.

```
for (int i = 0; i < 5; i++) {
```

```
    int y = i * 4;
```

```
}
```

```
i = 3; // Error!
```

```
y = 2; // Error!
```

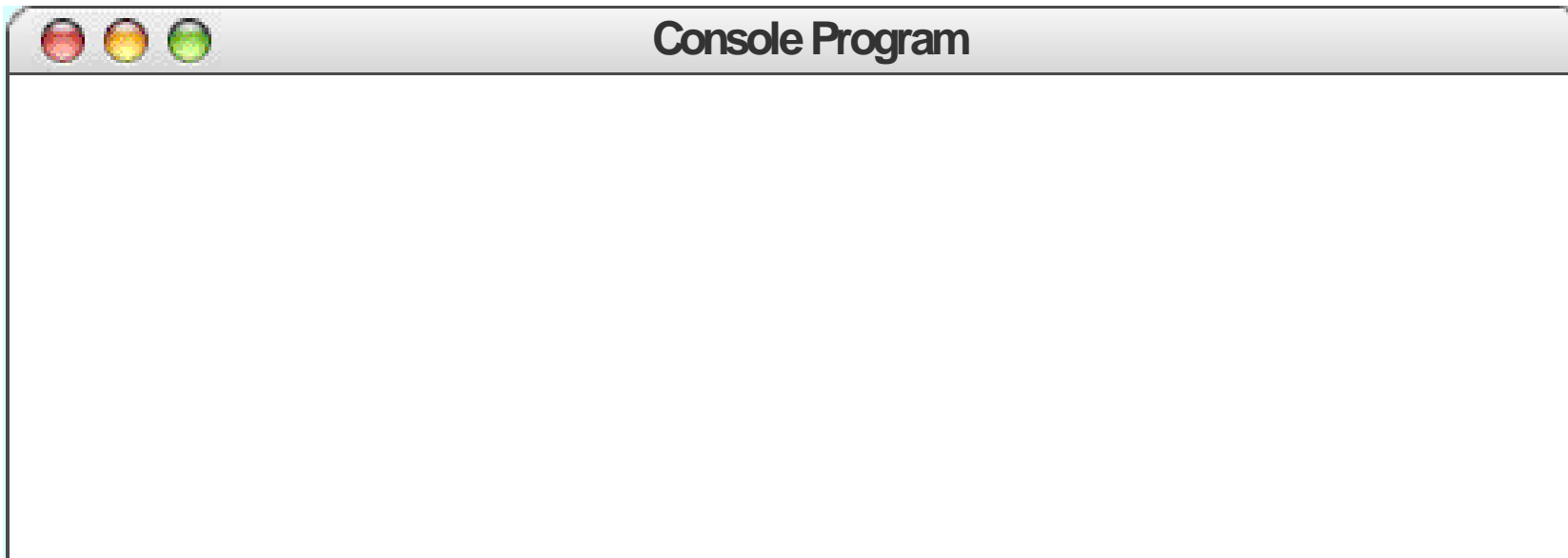

Scope of Method Calls

- A variable declared inside a method is called a **local variable**.
- Local variables can only be accessed inside of the method that declares them.

```
public void run() {  
    int x = 5;  
    someOtherMethod();  
}  
  
private void someOtherMethod() {  
    x = 4; // Error!  
}
```

```
public void run() {  
    for(int i = 0; i < MAX_NUM; i++) {  
        println(i + "! = " + factorial(i));  
    }  
}
```

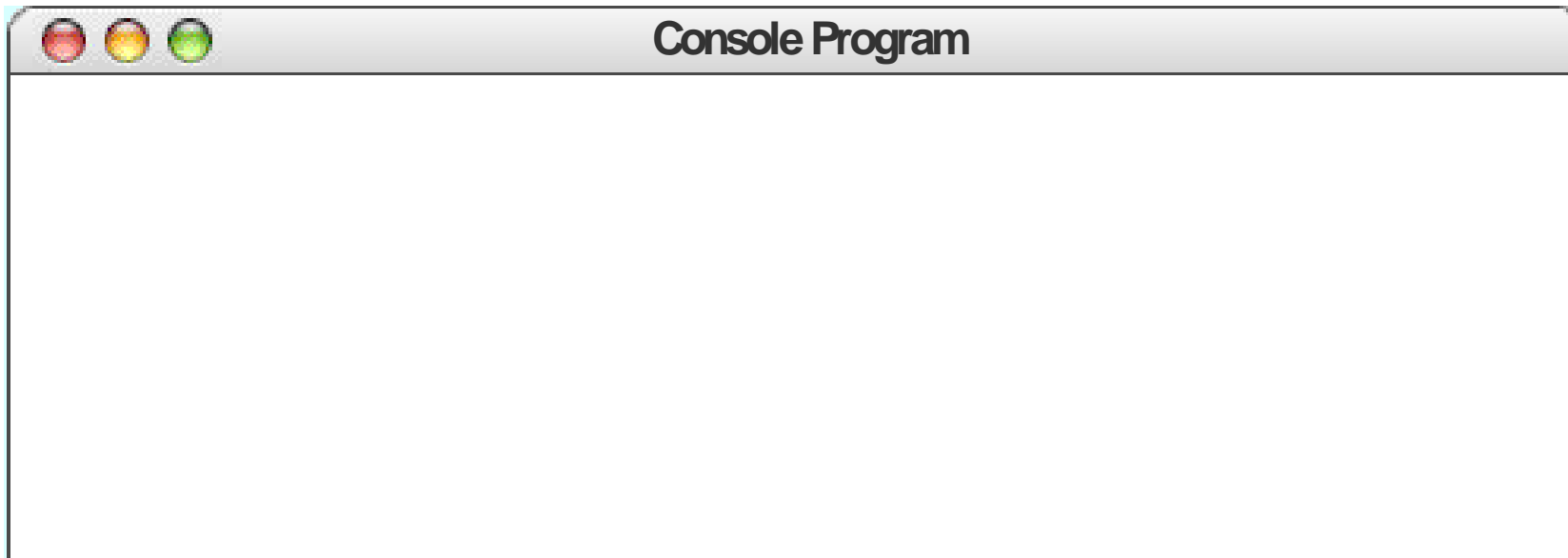
i



```
public void run() {  
    for(int i = 0; i < MAX_NUM; i++) {  
        println(i + "! = " + factorial(i));  
    }  
}
```

i

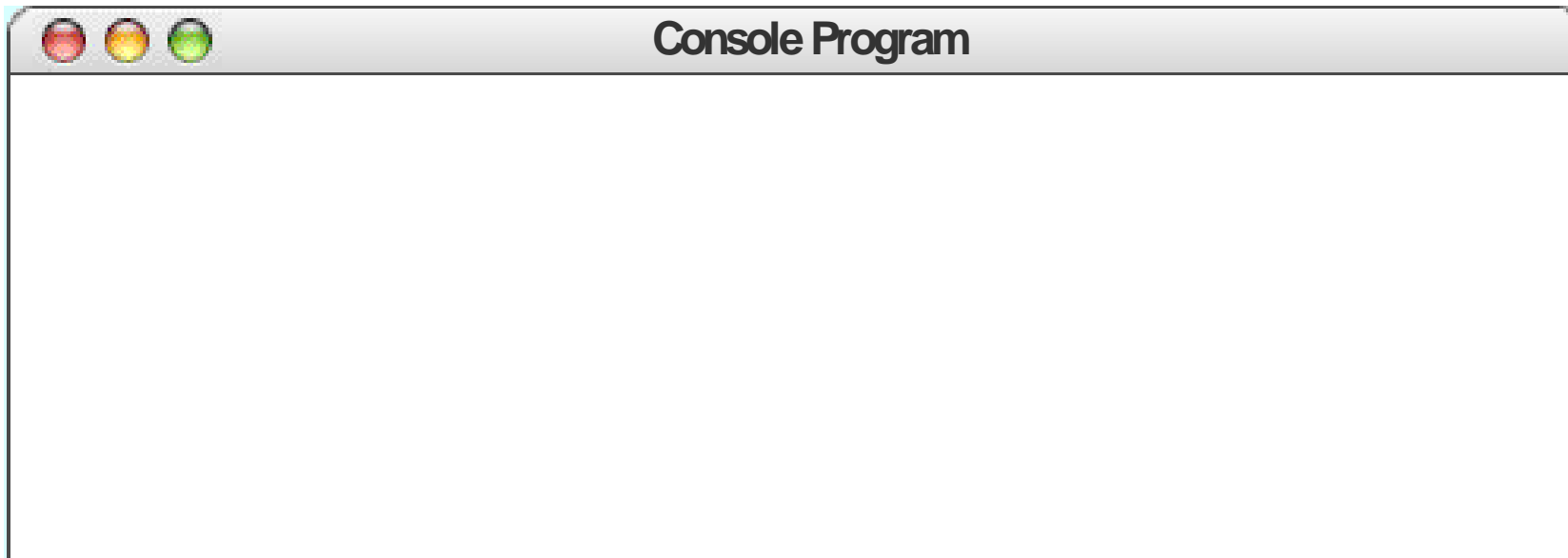
0



```
public void run() {  
    for(int i = 0; i < MAX_NUM; i++) {  
        println(i + "! = " + factorial(i));  
    }  
}
```

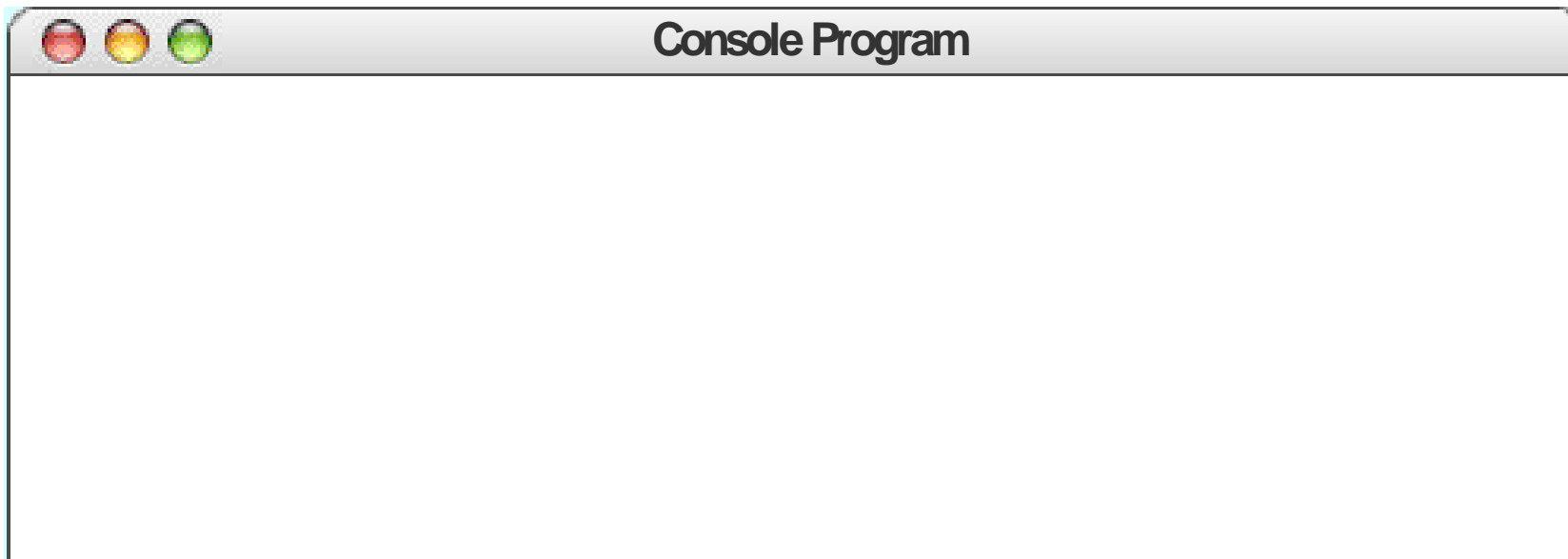
i

0



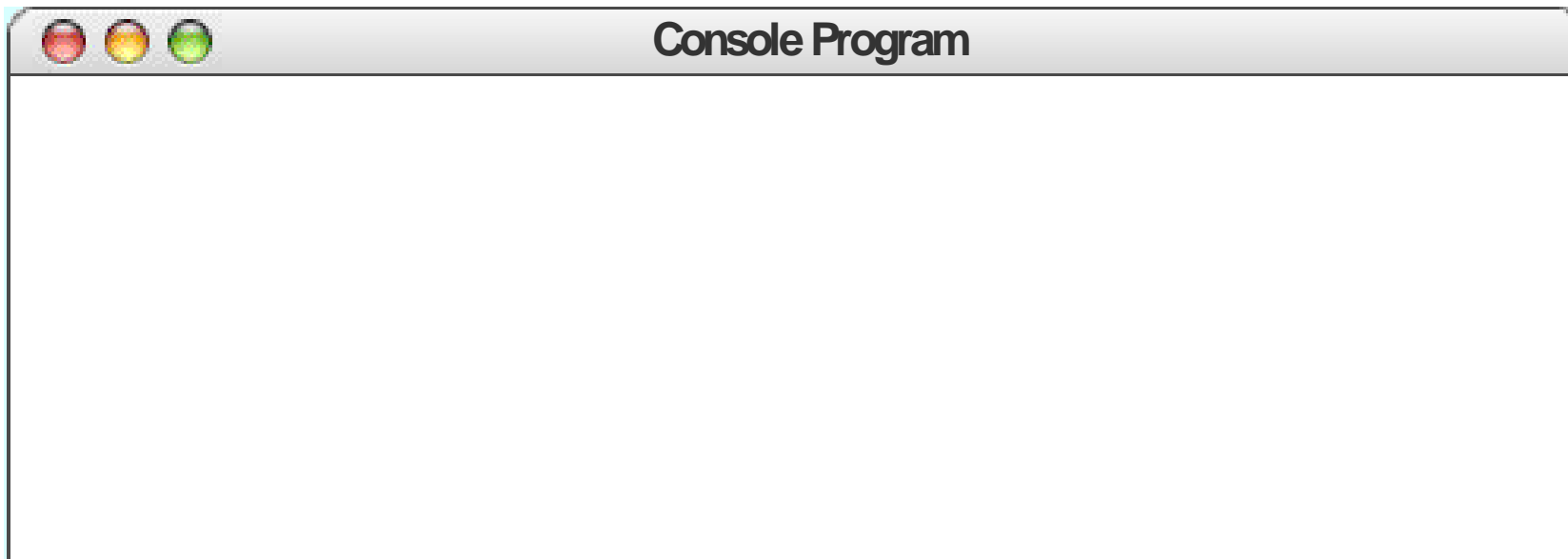
```
public void run() {  
    for(int i = 0; i < MAX NUM; i++) {  
        println(i + "! = " + factorial(i));  
    }  
}
```

i 0



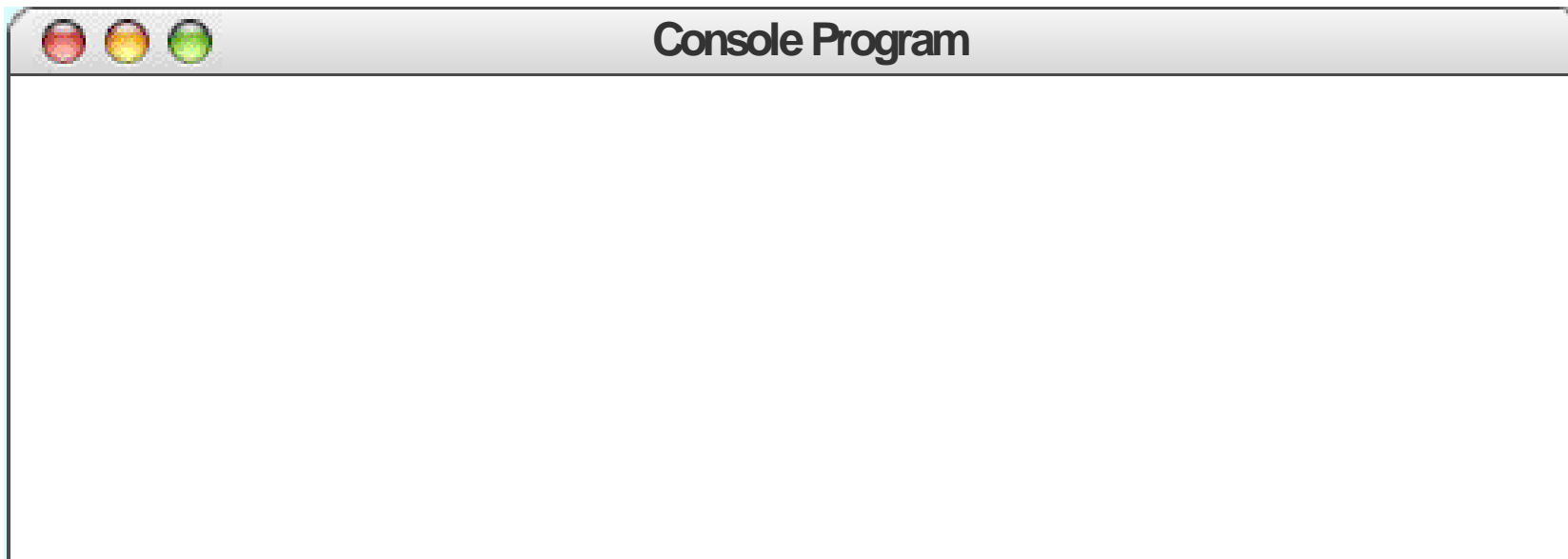
```
public void run() {  
    for(int i = 0; i < MAX_NUM; i++) {  
        println(i + "! = " + factorial(i));  
    }  
}
```

i 0



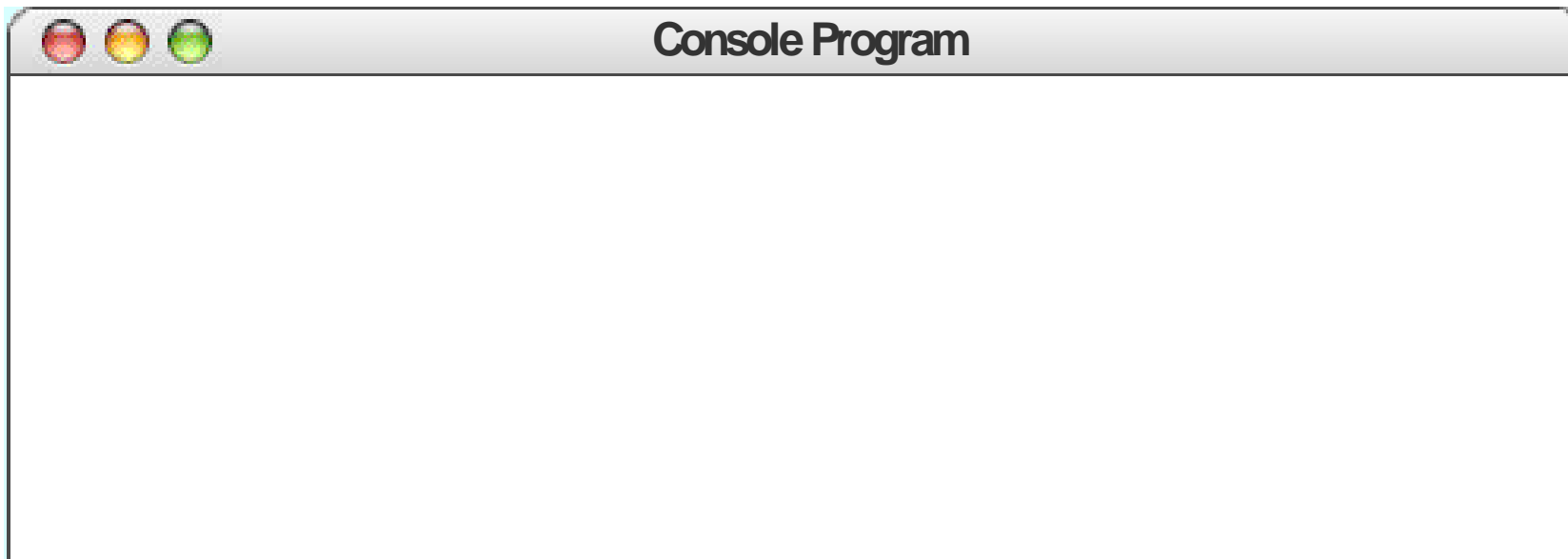
```
private int factorial(int n) {  
    int result = 1;  
    for (int i = 1; i <= n; i++) {  
        result *= i;  
    }  
    return result;  
}
```

n result i



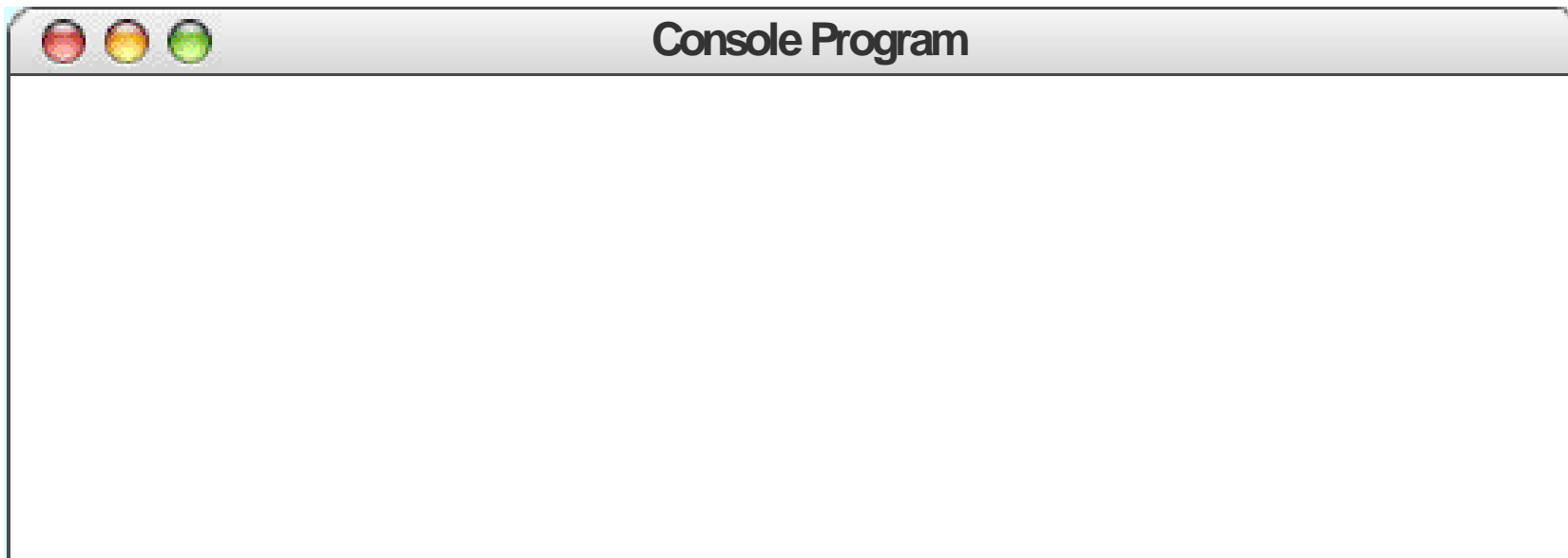
```
private int factorial(int n) {  
    int result = 1;  
    for (int i = 1; i <= n; i++) {  
        result *= i;  
    }  
    return result;  
}
```

n result i



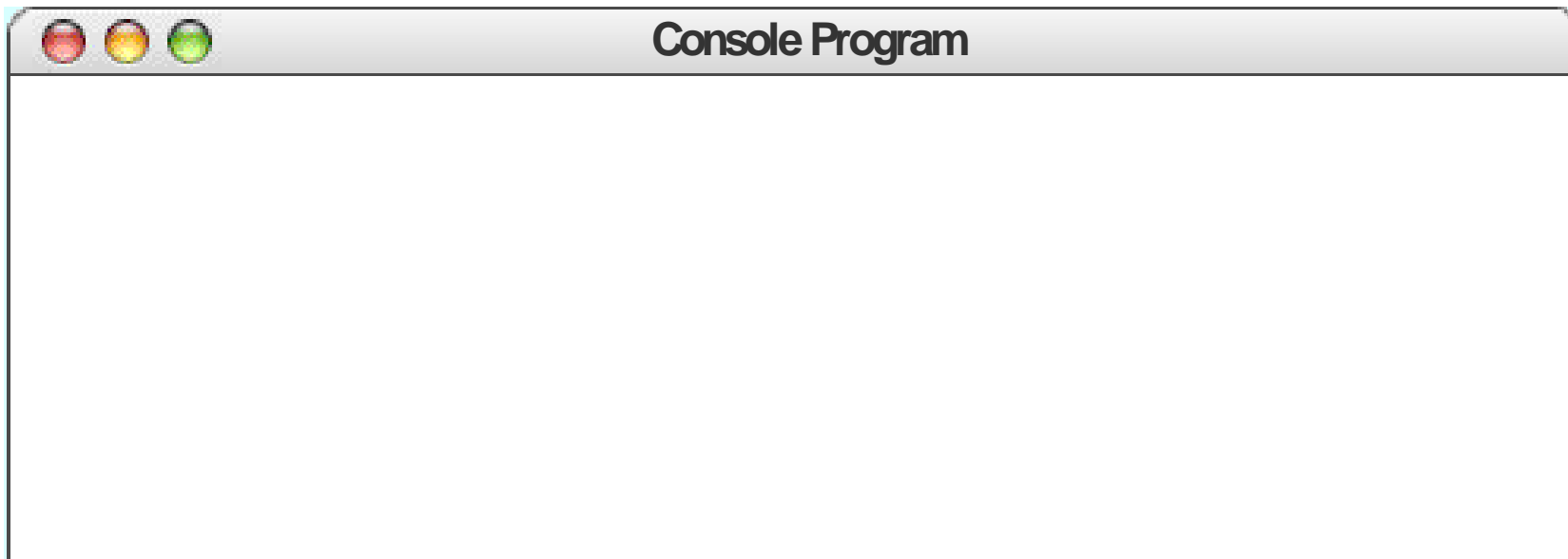

```
private int factorial(int n) {  
    int result = 1;  
    for (int i = 1; i <= n; i++) {  
        result *= i;  
    }  
    return result;  
}
```

n result i



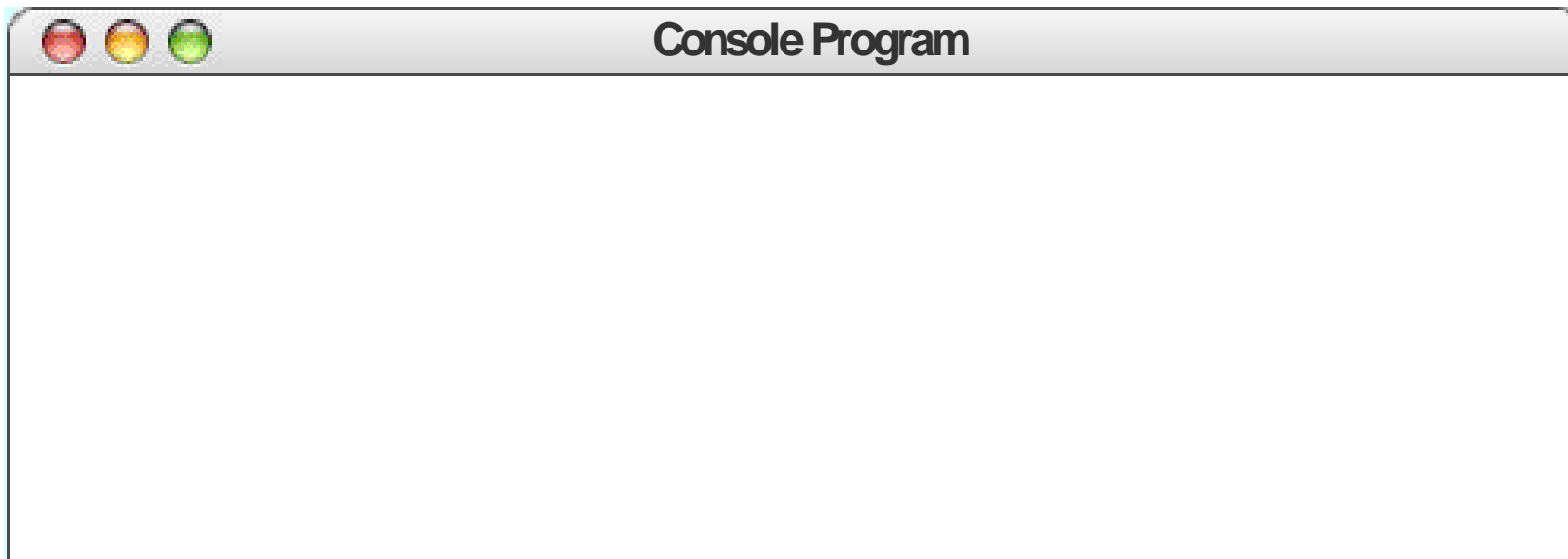
```
private int factorial(int n) {  
    int result = 1;  
    for (int i = 1; i <= n; i++) {  
        result *= i;  
    }  
    return result;  
}
```

n result i



```
private int factorial(int n) {  
    int result = 1;  
    for (int i = 1; i <= n; i++) {  
        result *= i;  
    }  
    return result;  
}
```

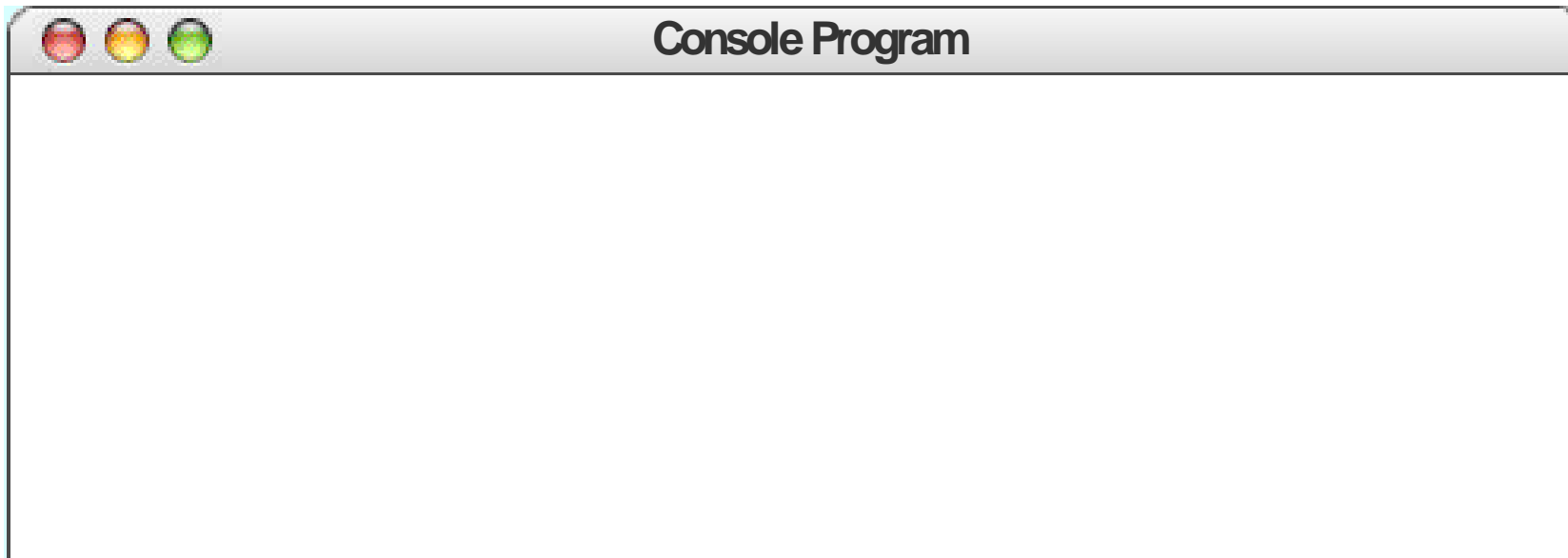
n result i



```
public void run() {  
    for(int i = 0; i < MAX_NUM; i++) {  
        println(i + "! = " + factorial(i));  
    }  
}
```

1

i 0



```
public void run() {  
    for(int i = 0; i < MAX NUM; i++) {  
        println(i + "! = " + factorial(i));  
    }  
}
```

1

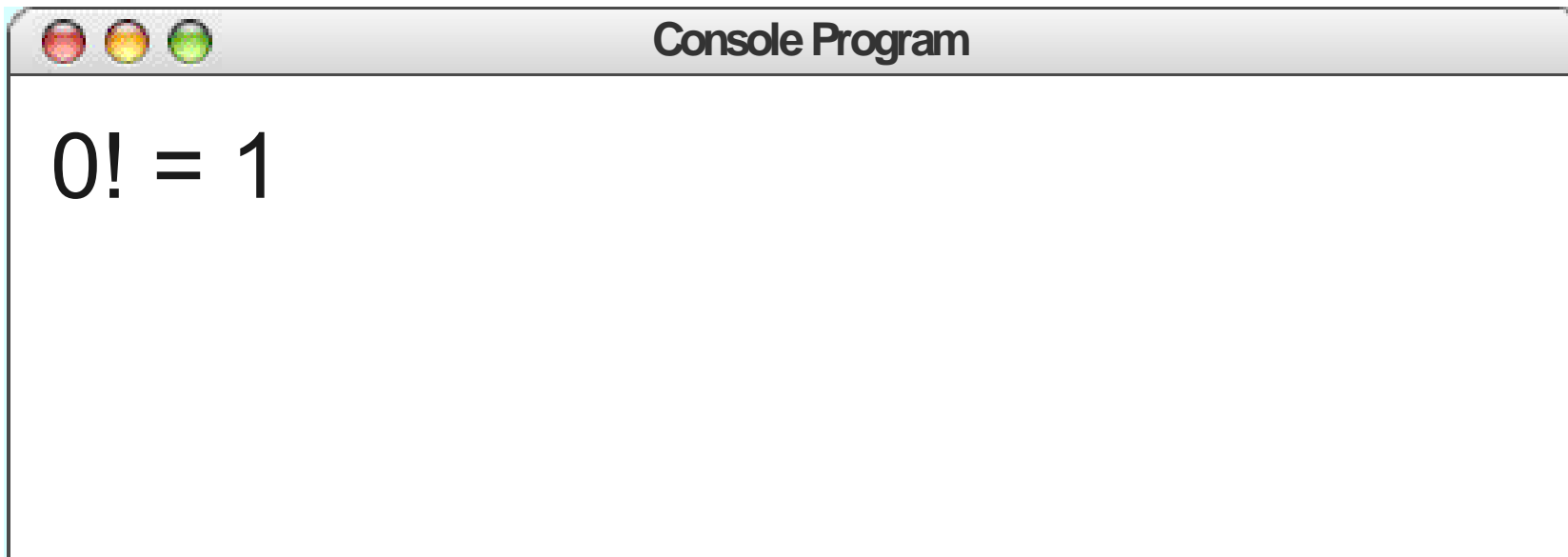
i 0

Console Program

0! = 1

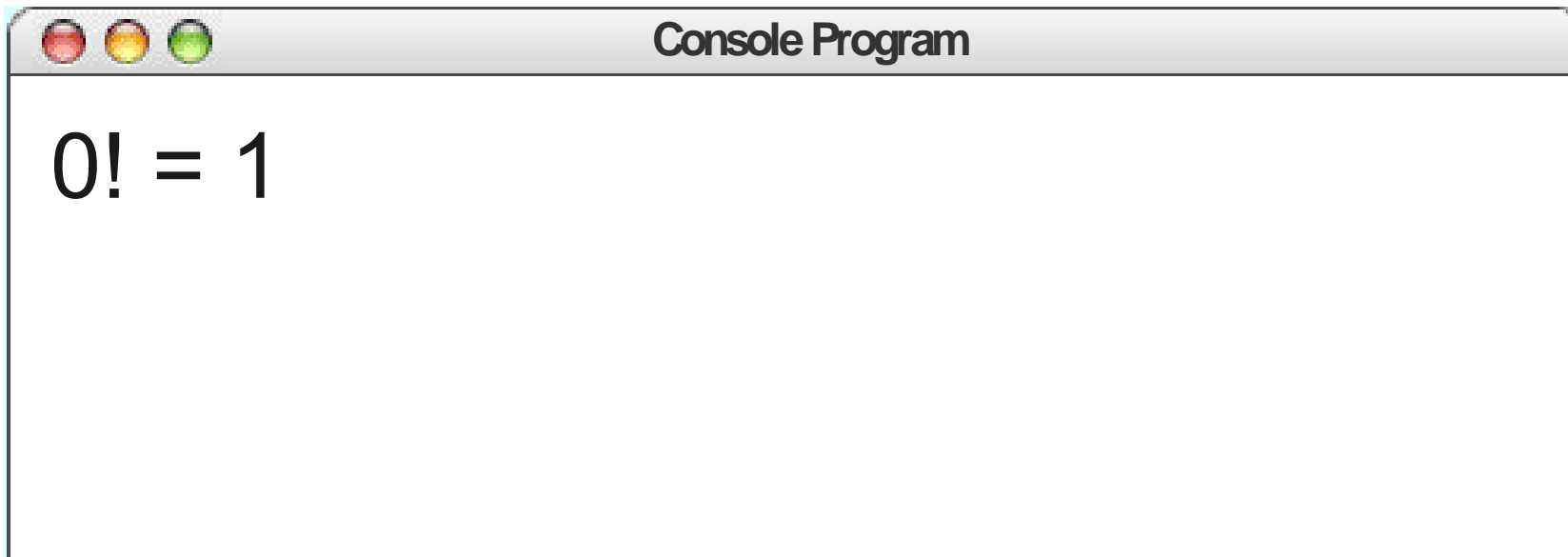
```
public void run() {  
    for(int i = 0; i < MAX_NUM; i++) {  
        println(i + "! = " + factorial(i));  
    }  
}
```

i 1



```
public void run() {  
    for(int i = 0; i < MAX_NUM; i++) {  
        println(i + "! = " + factorial(i));  
    }  
}
```

i 1



```
public void run() {  
    for(int i = 0; i < MAX NUM; i++) {  
        println(i + "! = " + factorial(i));  
    }  
}
```

i

1

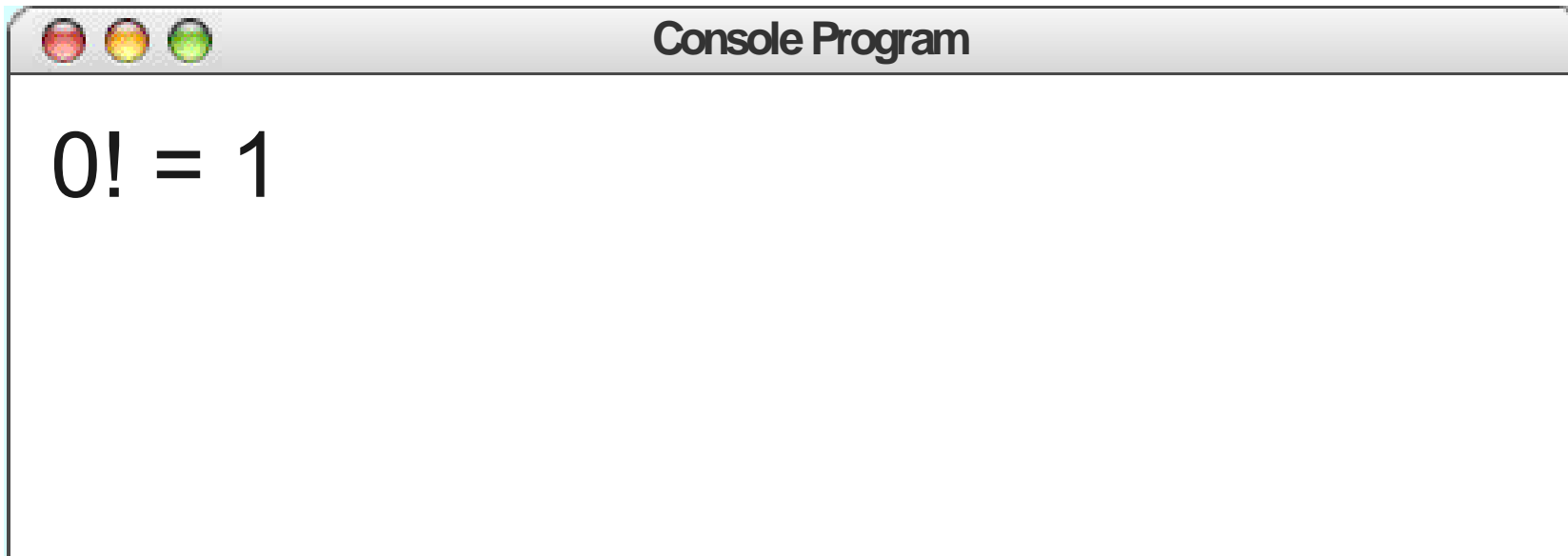


Console Program

0! = 1


```
public void run() {  
    for(int i = 0; i < MAX_NUM; i++) {  
        println(i + "! = " + factorial(i));  
    }  
}
```

i 1



```
private int factorial(int n) {  
    int result = 1;  
    for (int i = 1; i <= n; i++) {  
        result *= i;  
    }  
    return result;  
}
```

n result i



Console Program

0! = 1

```
private int factorial(int n) {  
    int result = 1;  
    for (int i = 1; i <= n; i++) {  
        result *= i;  
    }  
    return result;  
}
```

n result i



Console Program

0! = 1

```
private int factorial(int n) {  
    int result = 1;  
    for (int i = 1; i <= n; i++) {  
        result *= i;  
    }  
    return result;  
}
```

n result i

Console Program

0! = 1

```
private int factorial(int n) {  
    int result = 1;  
    for (int i = 1; i <= n; i++) {  
        result *= i;  
    }  
    return result;  
}
```

n result i



Console Program

0! = 1

```
private int factorial(int n) {  
    int result = 1;  
    for (int i = 1; i <= n; i++) {  
        result *= i;  
    }  
    return result;  
}
```

n result i



Console Program

0! = 1

```
private int factorial(int n) {  
    int result = 1;  
    for (int i = 1; i <= n; i++) {  
        result *= i;  
    }  
    return result;  
}
```

n result i

Console Program

0! = 1

```
private int factorial(int n) {  
    int result = 1;  
    for (int i = 1; i <= n; i++) {  
        result *= i;  
    }  
    return result;  
}
```

n result i

Console Program

0! = 1


```
private int factorial(int n) {  
    int result = 1;  
    for (int i = 1; i <= n; i++) {  
        result *= i;  
    }  
    return result;  
}
```

n result i

Console Program

0! = 1

```
public void run() {  
    for(int i = 0; i < MAX_NUM; i++) {  
        println(i + "! = " + factorial(i));  
    }  
}
```

1

i 1



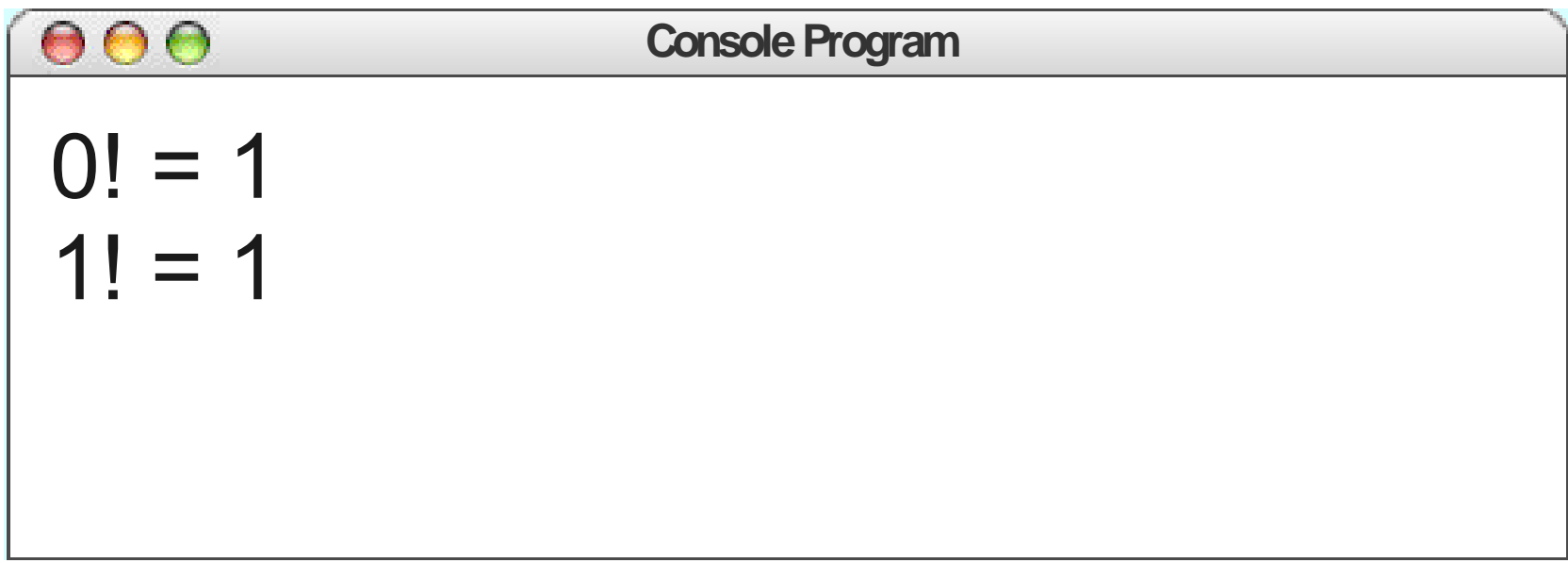
Console Program

0! = 1

```
public void run() {  
    for(int i = 0; i < MAX NUM; i++) {  
        println(i + "! = " + factorial(i));  
    }  
}
```

1

i 1



```
public void run() {  
    for(int i = 0; i < MAX_NUM; i++) {  
        println(i + "! = " + factorial(i));  
    }  
}
```

i

2



Console Program

0! = 1

1! = 1

```
public void run() {  
    for(int i = 0; i < MAX_NUM; i++) {  
        println(i + "! = " + factorial(i));  
    }  
}
```

i 2

Console Program

0! = 1

1! = 1

```
public void run() {  
    for(int i = 0; i < MAX NUM; i++) {  
        println(i + "! = " + factorial(i));  
    }  
}
```

i

2



Console Program

0! = 1

1! = 1

```
public void run() {  
    for(int i = 0; i < MAX_NUM; i++) {  
        println(i + "! = " + factorial(i));  
    }  
}
```

i 2



Console Program

```
0! = 1  
1! = 1
```

```
public void run() {  
    for(int i = 0; i < MAX_NUM; i++) {  
        println(i + "! = " + factorial(i));  
    }  
}
```

2

i

2



Console Program

0! = 1

1! = 1


```
public void run() {  
    for(int i = 0; i < MAX NUM; i++) {  
        println(i + "! = " + factorial(i));  
    }  
}
```

2

i

2



Console Program

```
0! = 1  
1! = 1  
2! = 2
```

```
public void run() {  
    for(int i = 0; i < MAX_NUM; i++) {  
        println(i + "! = " + factorial(i));  
    }  
}
```

i

3



Console Program

```
0! = 1  
1! = 1  
2! = 2
```

```
public void run() {  
    for(int i = 0; i < MAX_NUM; i++) {  
        println(i + "! = " + factorial(i));  
    }  
}
```

i 3



Console Program

```
0! = 1  
1! = 1  
2! = 2
```

```
public void run() {  
    for(int i = 0; i < MAX NUM; i++) {  
        println(i + "! = " + factorial(i));  
    }  
}
```

i

3



Console Program

```
0! = 1  
1! = 1  
2! = 2
```

```
public void run() {  
    for(int i = 0; i < MAX_NUM; i++) {  
        println(i + "! = " + factorial(i));  
    }  
}
```

i

3



Console Program

```
0! = 1  
1! = 1  
2! = 2
```

```
public void run() {  
    for(int i = 0; i < MAX_NUM; i++) {  
        println(i + "! = " + factorial(i));  
    }  
}
```

6

i

3



Console Program

```
0! = 1  
1! = 1  
2! = 2
```

```
public void run() {  
    for(int i = 0; i < MAX NUM; i++) {  
        println(i + "! = " + factorial(i));  
    }  
}
```

6

i

3



Console Program

0! = 1

1! = 1

2! = 2

3! = 6

```
public void run() {  
    for(int i = 0; i < MAX_NUM; i++) {  
        println(i + "! = " + factorial(i));  
    }  
}
```

i 4



Console Program

0! = 1

1! = 1

2! = 2

3! = 6


```
public void run() {  
    for(int i = 0; i < MAX_NUM; i++) {  
        println(i + "! = " + factorial(i));  
    }  
}
```

i 4



Console Program

0! = 1

1! = 1

2! = 2

3! = 6

Retiring Young

Pass-by-Value

- Java methods pass their parameters by **value**.
- The method gets a *copy* of its parameters, not the actual parameters themselves.

```
private void myMethod(int x) {  
    x = 137;  
}  
  
public void run() {  
    int x = 42;  
    myMethod(x);  
    println("The value of x is " + x);  
}
```

This statement
prints 42,
not 137.

Time-Out For Announcements!

Assignment 2

- Assignment 2 is due on Friday.
- **Recommendation:** Complete all parts of the assignment by Wednesday. Start testing the first five parts of the assignment.
- LaIR open through Thursday, 6PM – Midnight.

Back to CS106A!

Animation

Operations on the GObject Class

The following operations apply to all GObjects:

object.setColor(color)

Sets the color of the object to the specified color constant.

object.setLocation(x, y)

Changes the location of the object to the point (x, y).

object.move(dx, dy)

Moves the object on the screen by adding *dx* and *dy* to its current coordinates.

Standard color names defined in the `java.awt` package:

`Color.BLACK`

`Color.RED`

`Color.BLUE`

`Color.DARK_GRAY`

`Color.YELLOW`

`Color.MAGENTA`

`Color.GRAY`

`Color.GREEN`

`Color.ORANGE`

`Color.LIGHT_GRAY`

`Color.CYAN`

`Color.PINK`

`Color.WHITE`

Operations on the GObject Class

The following operations apply to all `GObject`s:

`object.setColor(color)`

Sets the color of the object to the specified color constant.

`object.setLocation(x, y)`

Changes the location of the object to the point (x, y) .

`object.move(dx, dy)`

Moves the object on the screen by adding dx and dy to its current coordinates.

Standard color names defined in the `java.awt` package:

`Color.BLACK`

`Color.RED`

`Color.BLUE`

`Color.DARK_GRAY`

`Color.YELLOW`

`Color.MAGENTA`

`Color.GRAY`

`Color.GREEN`

`Color.ORANGE`

`Color.LIGHT_GRAY`

`Color.CYAN`

`Color.PINK`

`Color.WHITE`

Animation

- By repositioning objects after they have been added to the canvas, we can create animations.
- General pattern for animation:

```
while (not-done-condition) {  
    update graphics  
    pause (pause-time) ;  
}
```

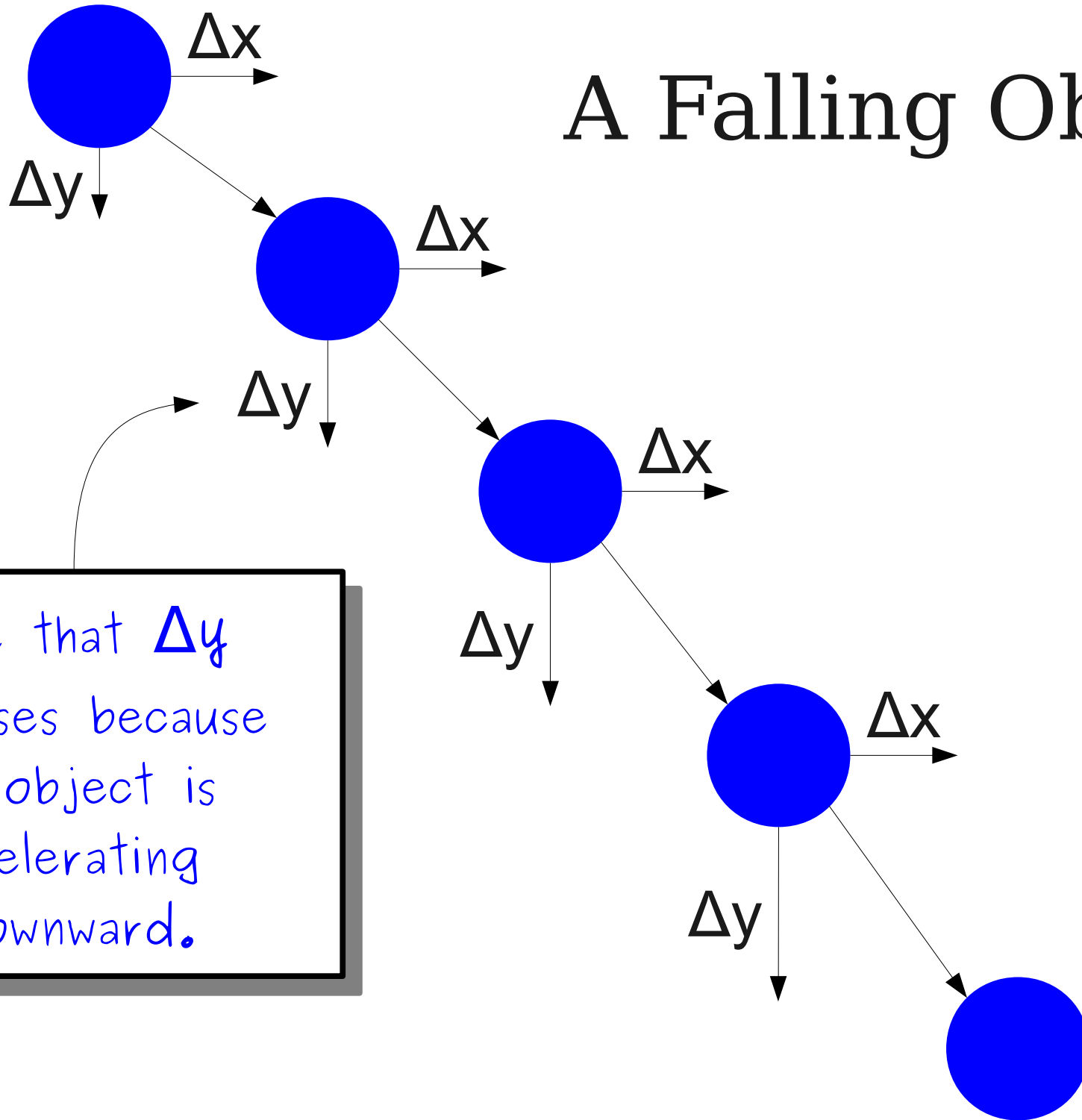
Physics Simulation





http://physbam.stanford.edu/~fedkiw/animations/motion_smoke.avi

A Falling Object

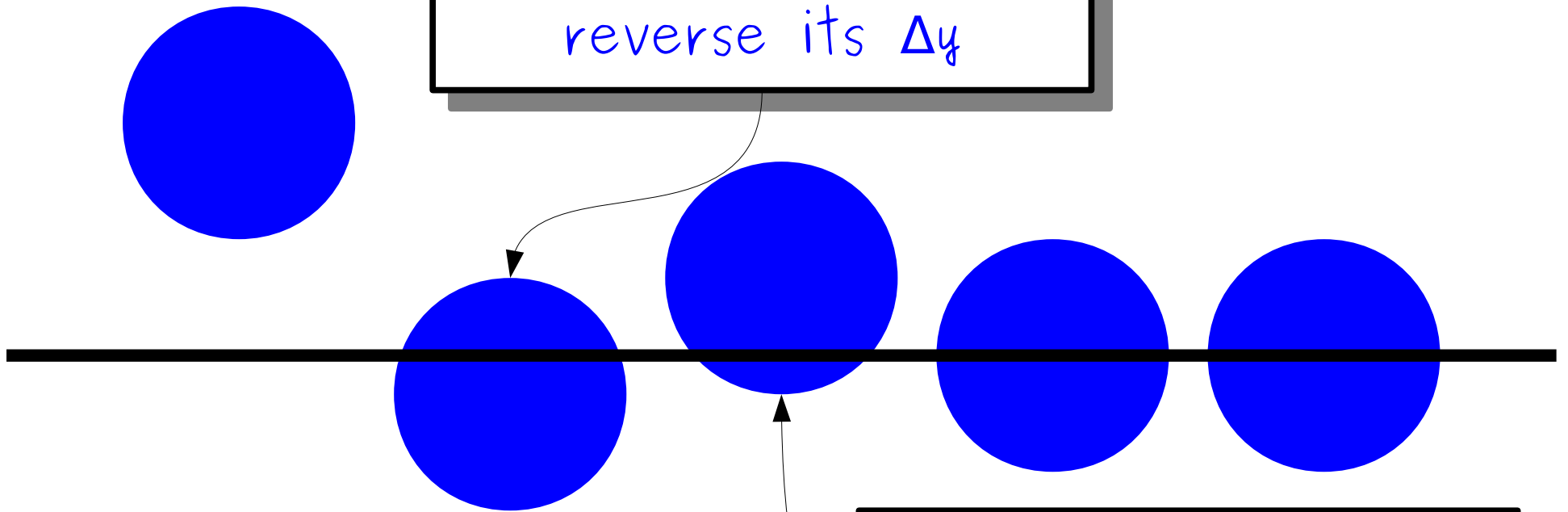


Note that Δy increases because the object is accelerating downward.

Let's Code It Up!

A Sticky Situation

The ball is below the ground, so we reverse its Δy



It's still below the ground, so we reverse its Δy again.