Even More Strings!

Midterm Logistics

- Midterm next **Monday, February 11** from **7-10PM**.
  - We will email out room assignments soon.
- Covers material up through and including today's lecture.
- Open-book, open-note, closed computer.

Recap from Last Time

A man, a plan, a caret, a ban, a myriad, a sum, a lac, a liar, a hoop, a pint, a catalpa, a gas, an oil, a bird, a yell, a vat, a caw, a pax, a wag, a tax, a nay, a ram, a cap, a yam, a gay, a tsar, a wall, a car, a luger, a ward, a bin, a woman, a vassal, a wolf, a tuna, a nit, a pall, a fret, a watt, a bay, a daub, a tan, a cab, a datum, a gall, a hat, a tag, a zap, a say, a jaw, a lay, a wet, a gallop, a tug, a trot, a trap, a tram, a torr, a caper, a top, a tonk, a toll, a ball, a fair, a sax, a minim, a tenor, a bass, a passer, a capital, a rut, an amen, a ted, a cabal, a tang, a sun, an ass, a maw, a sag, a jam, a dam, a sub, a salt, an axon, a sail, an ad, a wadi, a radian, a room, a rood, a rip, a tad, a pariah, a revel, a reel, a reed, a pool, a plug, a pin, a peek, a parabola, a dog, a pat, a cud, a nu, a fan, a pal, a rum, a nod, an eta, a lag, an eel, a batik, a mug, a mot, a nap, a maxim, a mood, a leek, a grub, a gob, a gel, a drab, a citadel, a total, a cedar, a tap, a gag, a rat, a manor, a bar, a gal, a cola, a pap, a yaw, a tab, a raj, a gab, a nag, a pagan, a bag, a jar, a bat, a way, a papa, a local, a gar, a baron, a mat, a rag, a gap, a tar, a decal, a tot, a led, a tic, a bard, a leg, a bog, a burg, a keel, a doom, a mix, a map, an atom, a gum, a kit, a baleen, a gala, a ten, a don, a mural, a pan, a faun, a ducat, a pagoda, a lob, a rap, a keep, a nip, a gulp, a loop, a deer, a leer, a lever, a hair, a pad, a tapir, a door, a moor, an aid, a raid, a wad, an alias, an ox, an atlas, a bus, a madam, a jag, a saw, a mass, an anus, a gnat, a lab, a cadet, an em, a natural, a tip, a caress, a pass, a baronet, a minimax, a sari, a fall, a ballot, a knot, a pot, a rep, a carrot, a mart, a part, a tort, a gut, a poll, a gateway, a law, a jay, a sap, a zag, a tat, a hall, a gamut, a dab, a can, a tabu, a day, a batt, a waterfall, a patina, a nut, a flow, a lass, a van, a mow, a nib, a draw, a regular, a call, a war, a stay, a gam, a yap, a cam, a ray, an ax, a tag, a wax, a paw, a cat, a valley, a drib, a lion, a saga, a plat, a catnip, a pooh, a rail, a calamus, a dairyman, a bater, a canal – Panama!

Searching a String

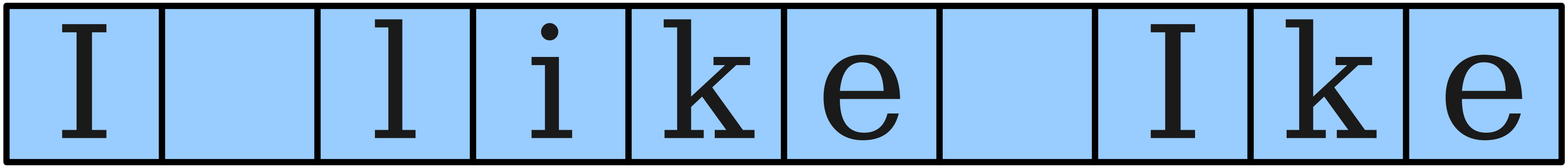
- You can search a string for a particular character or string by using the **`indexOf`** method:

***string*.indexOf(*pattern*)**

- **`indexOf`** returns the index of the first match if one exists.
- Otherwise, it returns -1 as a sentinel.

Replacing a Substring

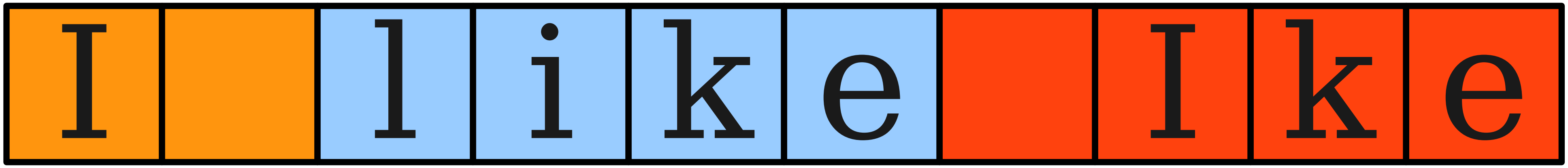
- Because strings are immutable, it can be difficult to replace pieces of a string.
- To replace a segment of a string:
  - Obtain the **substring** of the string up to the point to replace.
  - Obtain the substring of the string after the point to replace.



I		l	i	k	e		I	k	e

Replacing a Substring

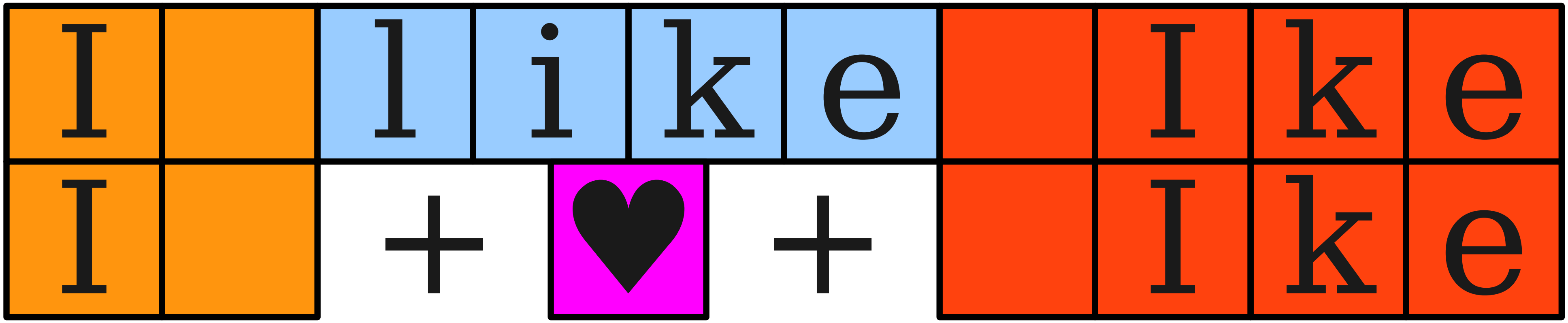
- Because strings are immutable, it can be difficult to replace pieces of a string.
- To replace a segment of a string:
    - Obtain the **substring** of the string up to the point to replace.
    - Obtain the substring of the string after the point to replace.



I		l	i	k	e		I	k	e

Replacing a Substring

- Because strings are immutable, it can be difficult to replace pieces of a string.
- To replace a segment of a string:
  - Obtain the **substring** of the string up to the point to replace.
  - Obtain the substring of the string after the point to replace.

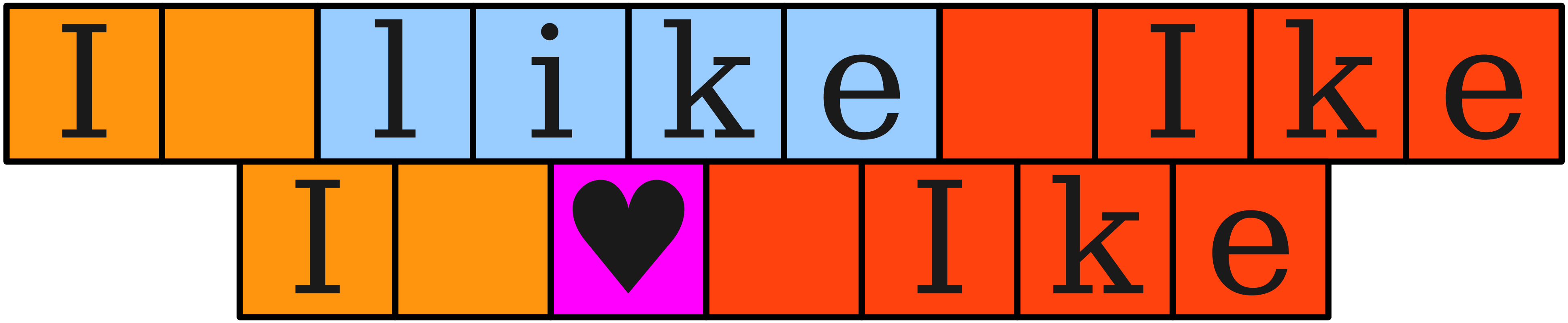


I like Ike

I + ♥ + Ike

Replacing a Substring

- Because strings are immutable, it can be difficult to replace pieces of a string.
- To replace a segment of a string:
  - Obtain the **substring** of the string up to the point to replace.
  - Obtain the substring of the string after the point to replace.



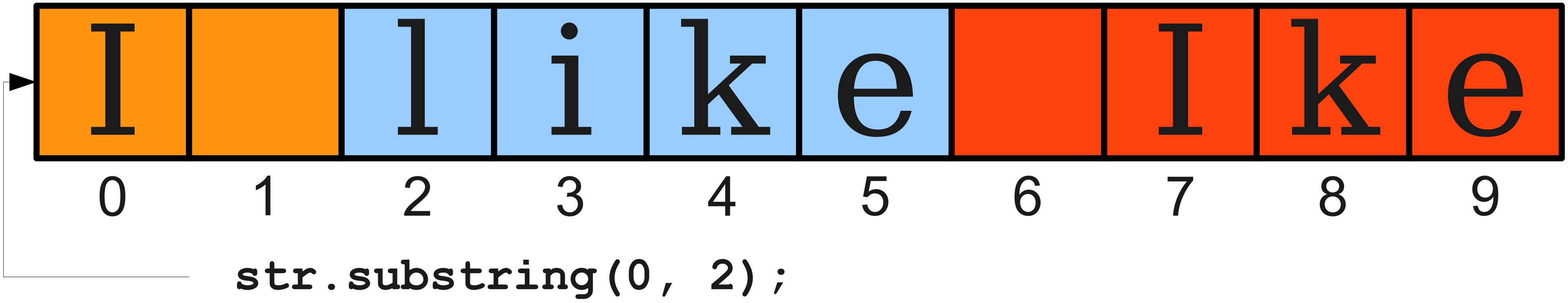
Obtaining Substrings

- To get all of the characters in the range [start, stop), use

string`.substring(`***start***`, `***stop***`)`

- To get all of the characters from some specified point forward, use

string`.substring(`***start***`)`



I		l	i	k	e		I	k	e
0	1	2	3	4	5	6	7	8	9

```
str.substring(0, 2);
```

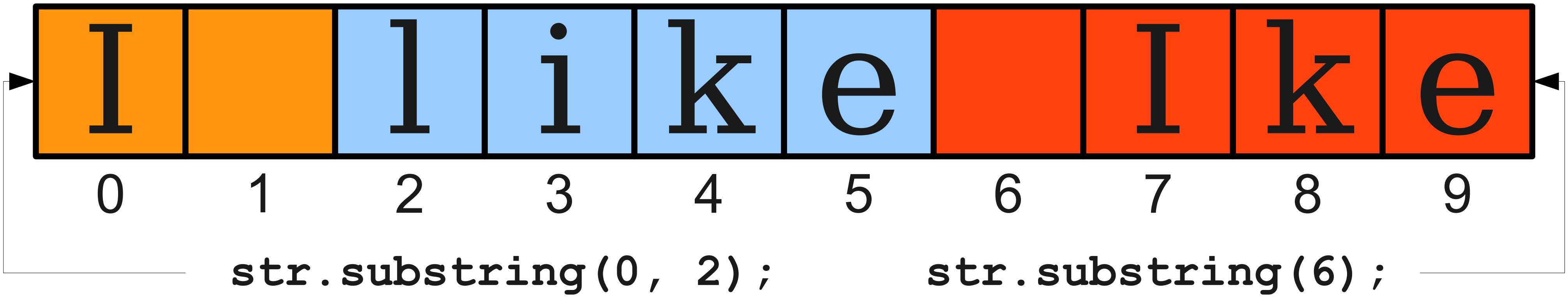
Obtaining Substrings

- To get all of the characters in the range [start, stop), use

```
string.substring(start, stop)
```

- To get all of the characters from some specified point forward, use

```
string.substring(start)
```



```
str.substring(0, 2);
```

```
str.substring(6);
```

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

Based on slides by Eric Roberts and Mehran Sahami

Classes

Objects Revisited

- An object is a combination of
  - **State** – persistent information, and
  - **Behavior** – the ability to operate on that state.

- **`GRect`** state:
  - Position
  - Size
  - Color
  - Is filled?
  - etc.

- **`GRect`** behavior:
  - Move
  - Change color
  - Change fill state
  - Tell position
  - etc.

Objects Revisited

- An object is a combination of
  - **State** – persistent information, and
  - **Behavior** – the ability to operate on that state.

- **`GPoint`** state:
  - Position

- **`GPoint`** behavior:
  - Move
  - Move by angle
  - Tell x
  - Tell y

Objects Revisited

- An object is a combination of
  - **State** – persistent information, and
  - **Behavior** – the ability to operate on that state.

- **`String`** state:
  - Character sequence

- **`String`** behavior:
  - Get characters
  - Produce substring
  - etc.

Classes and Objects

- Each object is an **instance** of some **class**.
- The class determines
  - what state each instance maintains.
  - what behaviors each instance possesses.
- Each instance determines
  - the specific values for that state information.

Creating our own Class

Image credit: http://store.controlconceptsusa.com/media/products/MC-Tally.jpg

Creating our own Class

- State:
  - The current number.

We use instance variables to keep track of state.

- Behavior:
  - Read the counter.
  - Increment the counter.

Creating our own Class

- State:
  - The current number.

We use instance variables to keep track of state.

- Behavior:
  - Read the counter.
  - Increment the counter.

We use methods to specify behavior.

Instance Variables Revisited

- Each instance of a class gets its own, unique copy of each instance variable.
- Different instances of the same object cannot read or write each others' instance variables.

Instance Variables Revisited

- Each instance of a class gets its own, unique copy of each instance variable.
- Different instances of the same object cannot read or write each others' instance variables.

Constructors

- A **constructor** is a special method defined in a class that is responsible for setting up class's instance variables to appropriate values.
- Syntax:

```
public NameOfClass(parameters)  {
  /* … body of constructor … */
}
```

- Inside a constructor:
  - Give initial values to instance variables.
  - Set up instance variables based on values specified in the parameters.
- Constructor called when instance created with `new`.

`toString()`

- To get a string representation of an object, Java uses a method

```
public String toString()
```

- If you define this method in your Java classes, you can customize what string will be produced.
- Otherwise, you get Icky Javaspeak string representations.