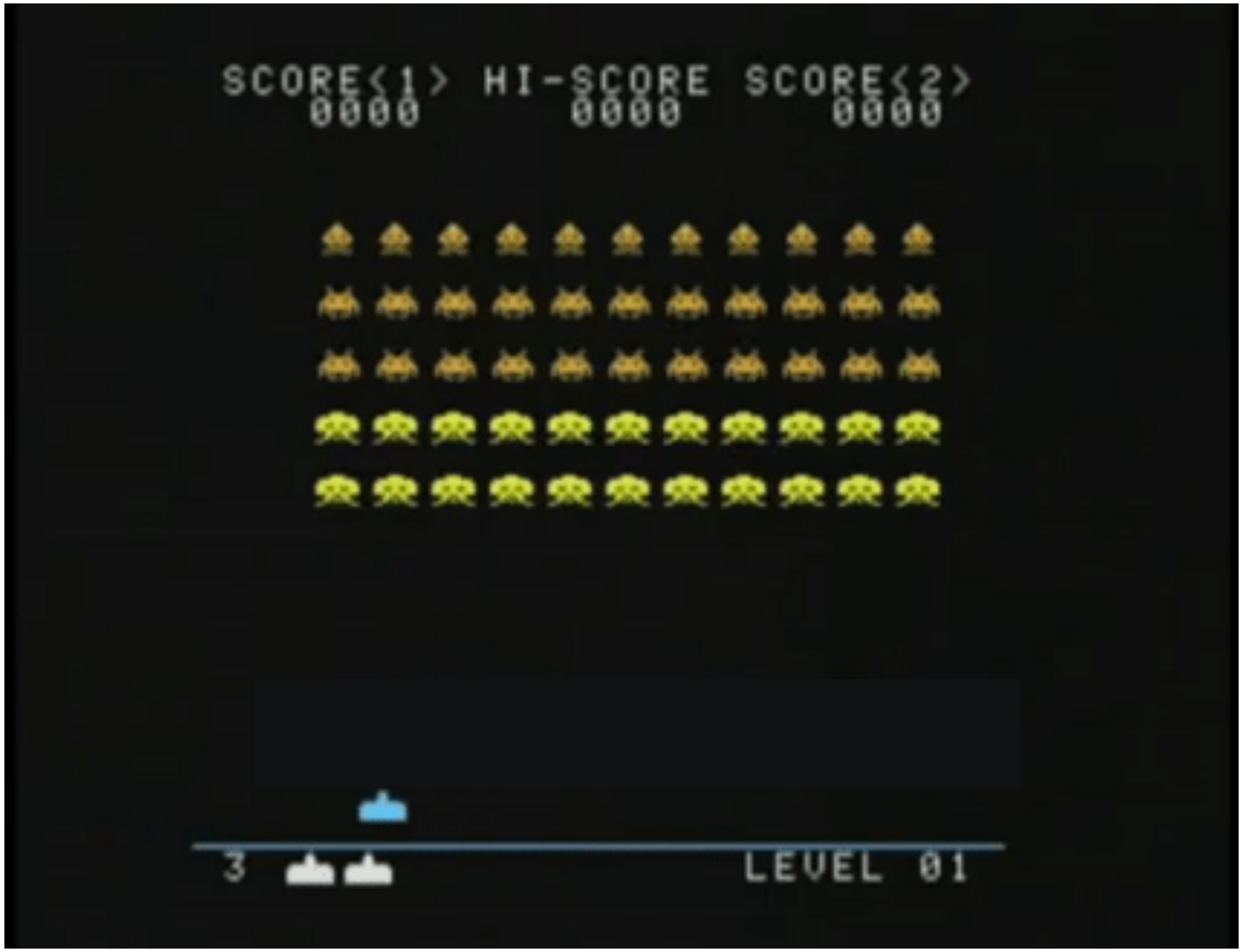
Strings

Final Project Example



How would you make this?
Take a deep breath…

Final Project Example

```
public class SpaceInvaders extends GraphicsProgram {

    private ArrayList<GOval> aliens;
    private GRect paddle;

    public void run() {
        // create objects and scores
        // add() to canvas
        setupGame();

        while(!gameOver()) {
            animateObjects();
            pause(100);
        }
    }

    private boolean keyPressed() {...}
    private boolean mousePressed() {...}
}
```

Setup

Animate

Interact

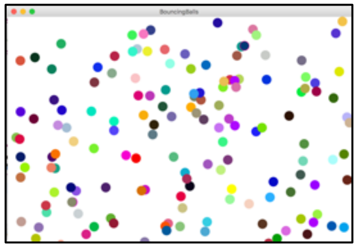
Start with a high-level look.

Final Project Example

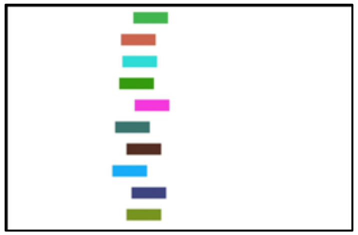
Animate

On every animation:

- Aliens move

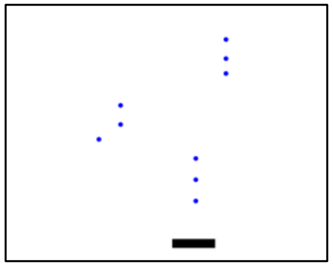


Bouncing Balls



Racing Cars

- Torpedos move



Rocket Paddle

Interact

← Key pressed:

- move ship left

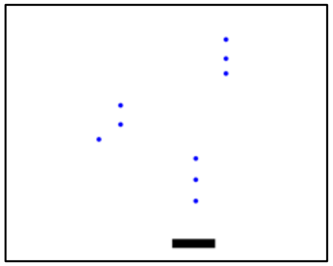
→ Key pressed:

- move ship right

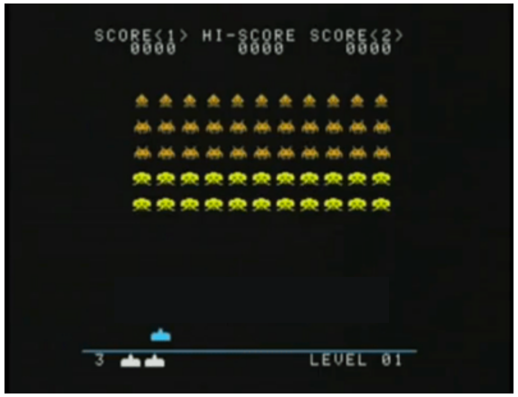
Keyboard Karel

Mouse pressed:

- Add torpedo



Rocket Paddle



Find worked examples!!

Learning Goals

1. Understand chars and Strings
2. Write methods acting on Strings
3. Learn something interesting

Text Applications

How is text represented?

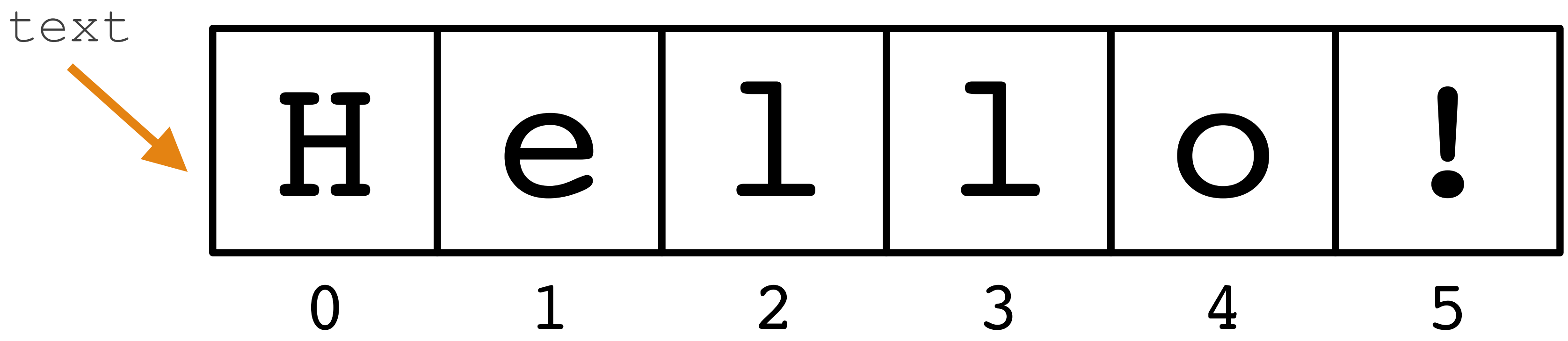
The variable type `String`

Text is stored using the variable type `String`.

A `String` is a sequence of characters.

```
public void run() {
    String text = "hello!";
    println(text);
}
```

The variable type `String`



- All characters in a string have an *index*.
- You can access a character in the string via its *index*.

```
char c = text.charAt(index);
```

- The *length* of a string is one larger than the last valid index in the string.

```
int len = text.length(); // 6
```

What String actually is

`String str` =

`char[] charArray` +

```
str.length();
str.charAt(i);
str.split(str);
str.contains(str);
```

Data storage

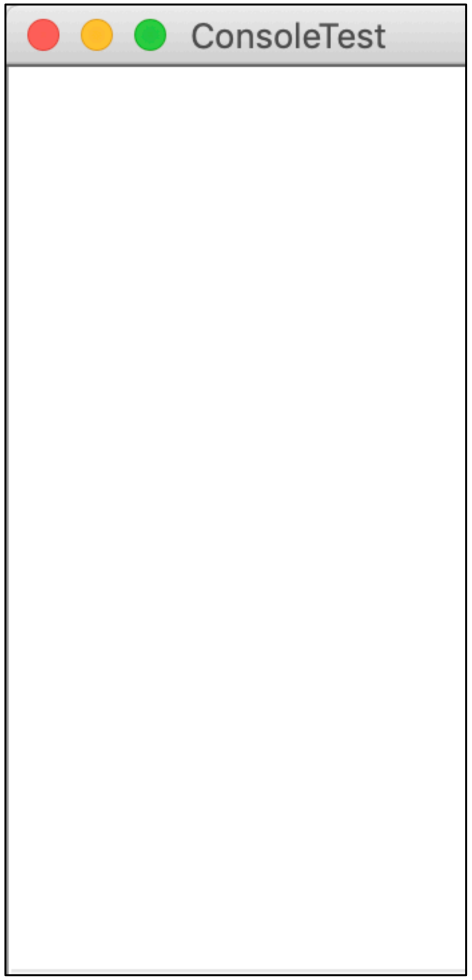
Useful methods

Why do both of these exist in the Java language?

- **`char`**`[]` builds off Java's fundamental data storage
- `String` adds convenient methods to **`char`**`[]`

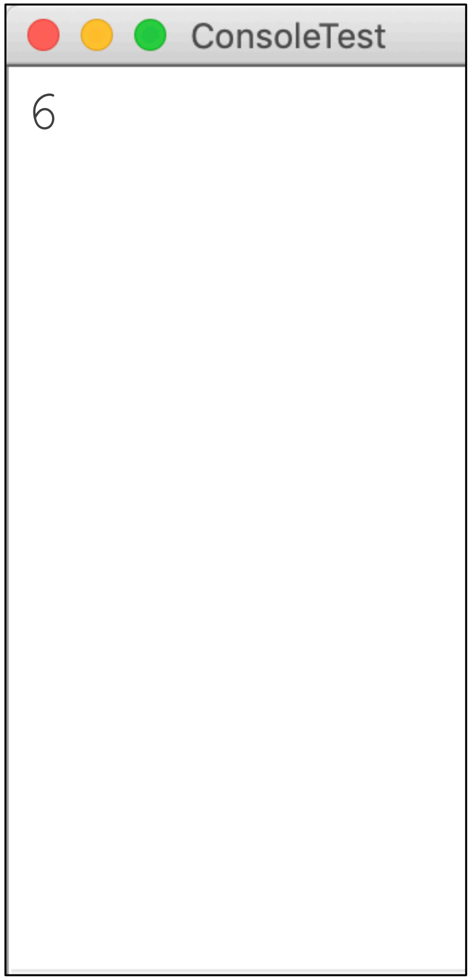
String Methods

```
public void run() {
    String example = "Hi mom";


    int length = example.length();
    println(length);


    char firstLetter = example.charAt(0);
    println(firstLetter);


    for(int i = 0; i < example.length(); i++) {
        char ch = example.charAt(i);
        println(ch);
    }
}
```

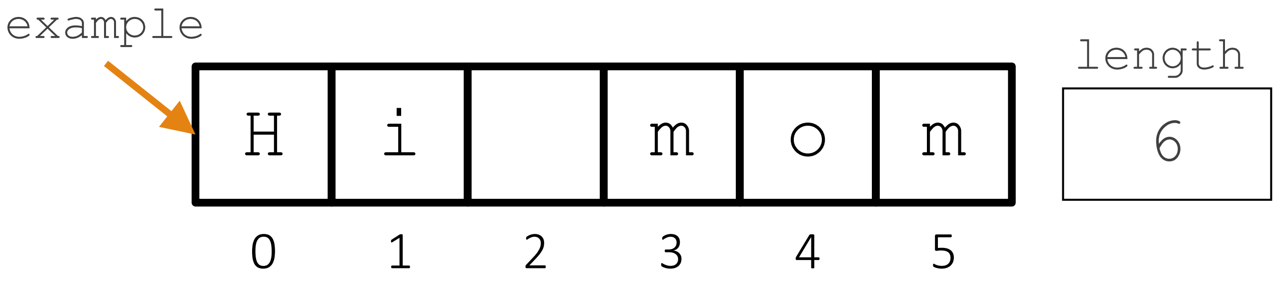


String Methods

```
public void run() {
     String example = "Hi mom";

     // example of length method
     int length = example.length();
     println(length); // prints 6

     // example of getCharAt
     char firstLetter = example.charAt(0);
     println(firstLetter); // prints 'H'

     // loop that prints letters one-by-one
     for(int i = 0; i < example.length(); i++) {
          char ch = example.charAt(i);
          println(ch);
     }
}
```

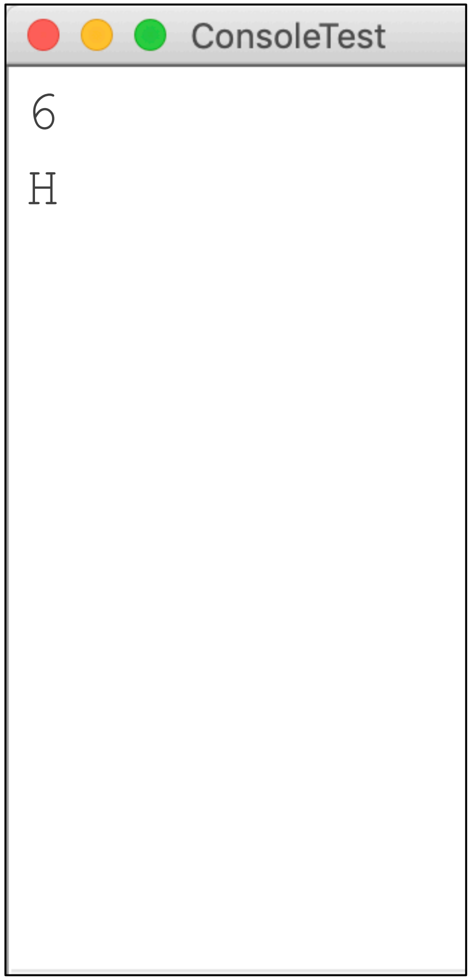


```
6
```

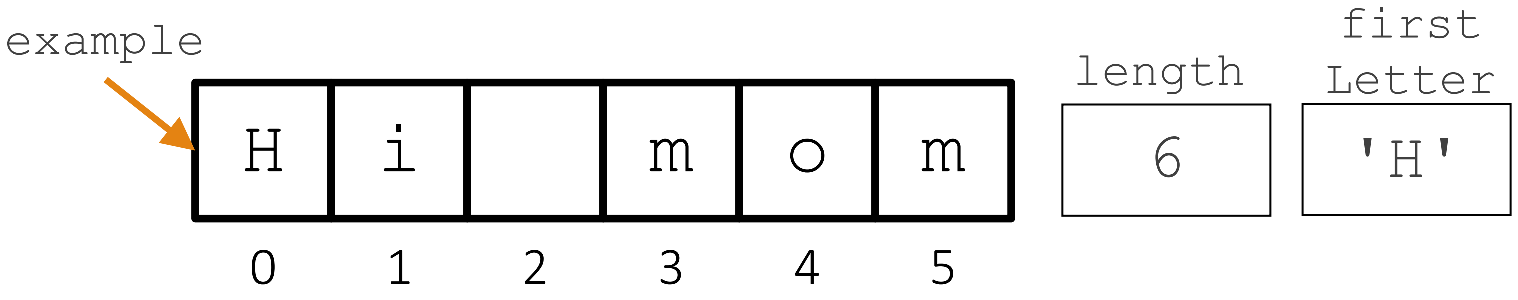


String Methods

```
public void run() {
     String example = "Hi mom";

     // example of length method
     int length = example.length();
     println(length); // prints 6

     // example of getCharAt
     char firstLetter = example.charAt(0);
     println(firstLetter); // prints 'H'

     // loop that prints letters one-by-one
     for(int i = 0; i < example.length(); i++) {
          char ch = example.charAt(i);
          println(ch);
     }
}
```

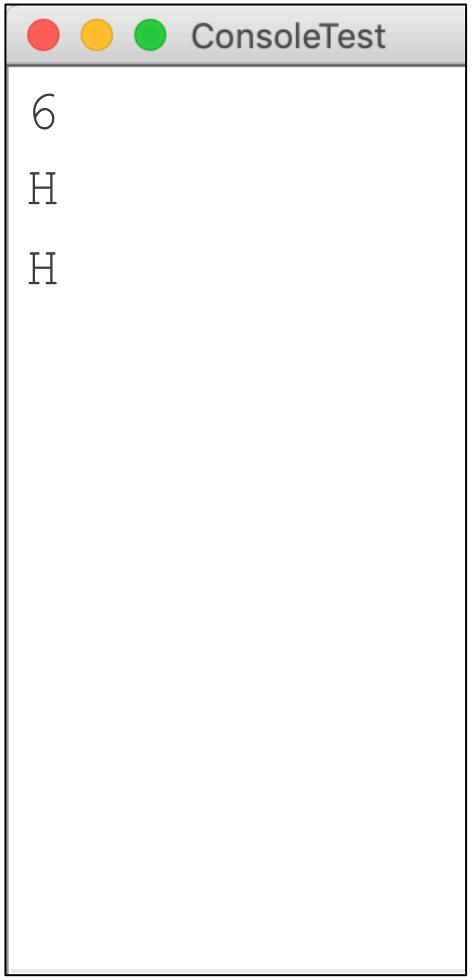


```
6
H
```

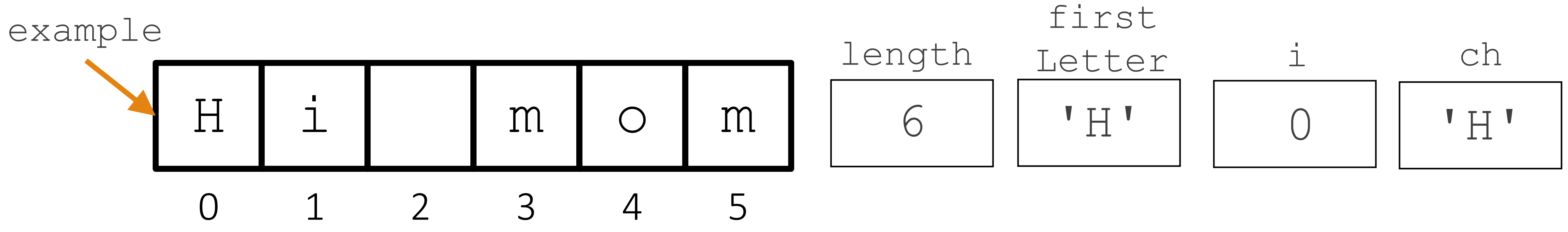


String Methods

```
public void run() {
    String example = "Hi mom";

    // example of length method
    int length = example.length();
    println(length); // prints 6

    // example of getCharAt
    char firstLetter = example.charAt(0);
    println(firstLetter); // prints 'H'

    // loop that prints letters one-by-one
    for(int i = 0; i < example.length(); i++) {
        char ch = example.charAt(i);
        println(ch);
    }
}
```

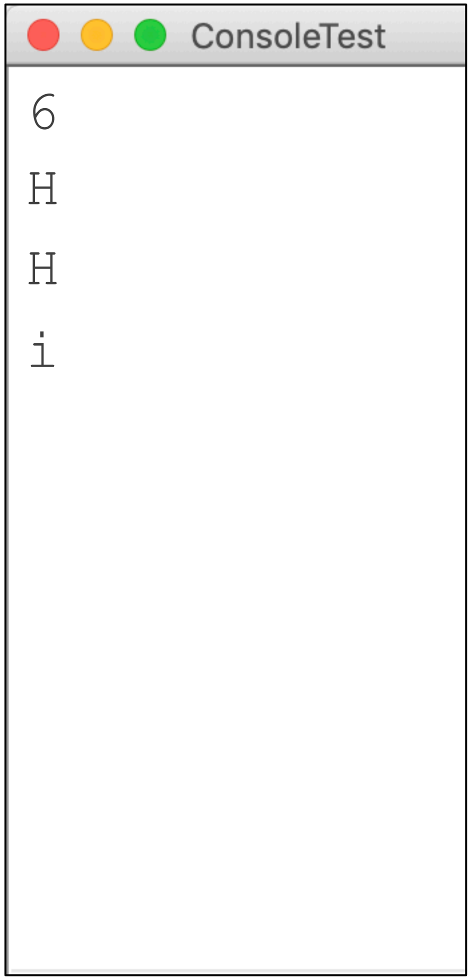


```
6
H
H
```

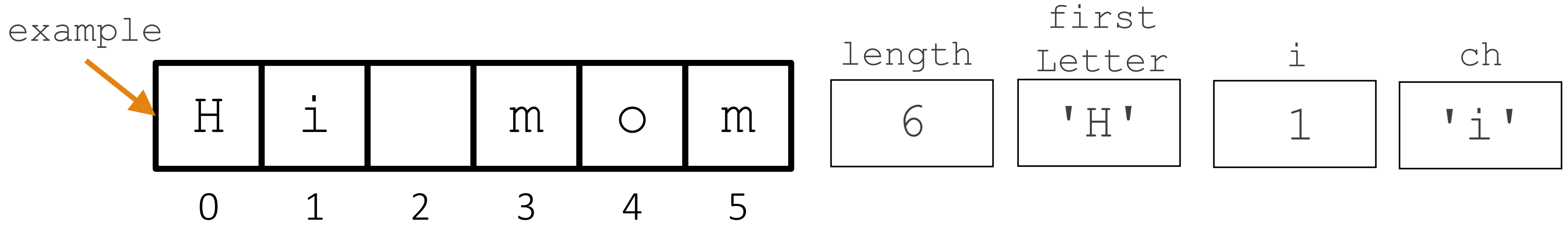


String Methods

```
public void run() {
    String example = "Hi mom";

    // example of length method
    int length = example.length();
    println(length); // prints 6

    // example of getCharAt
    char firstLetter = example.charAt(0);
    println(firstLetter); // prints 'H'

    // loop that prints letters one-by-one
    for(int i = 0; i < example.length(); i++) {
        char ch = example.charAt(i);
        println(ch);
    }
}
```

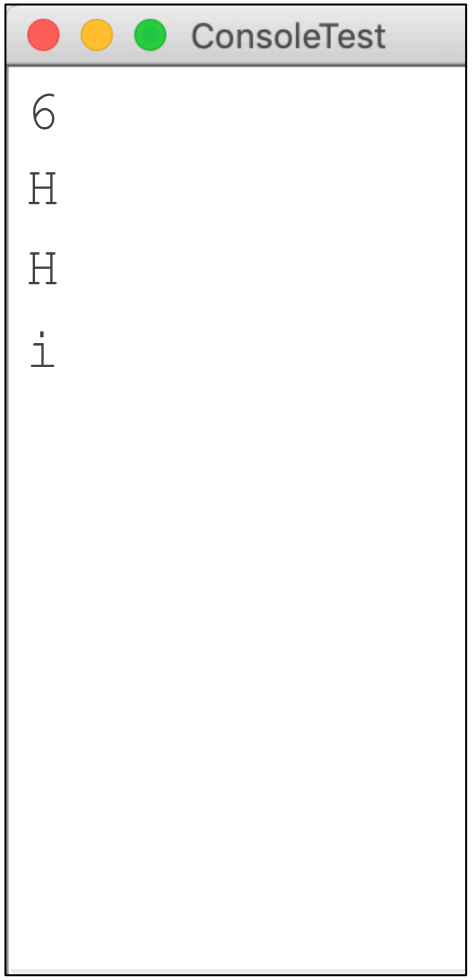


```
6
H
H
i
```

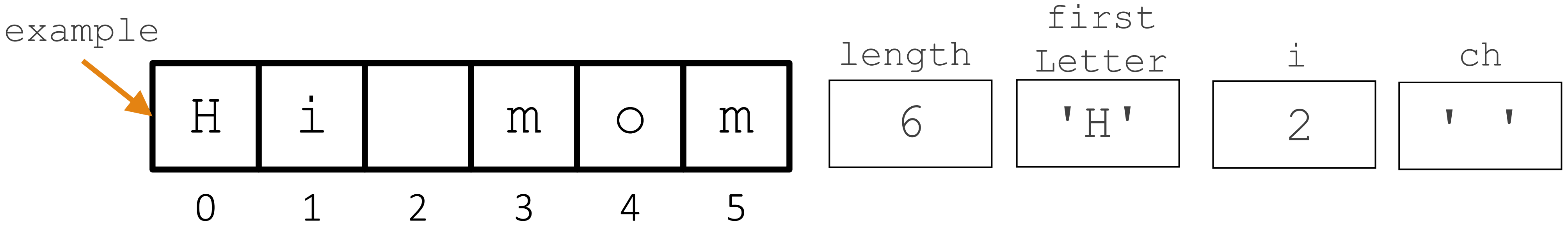


String Methods

```
public void run() {
    String example = "Hi mom";

    // example of length method
    int length = example.length();
    println(length); // prints 6

    // example of getCharAt
    char firstLetter = example.charAt(0);
    println(firstLetter); // prints 'H'

    // loop that prints letters one-by-one
    for(int i = 0; i < example.length(); i++) {
        char ch = example.charAt(i);
        println(ch);
    }
}
```

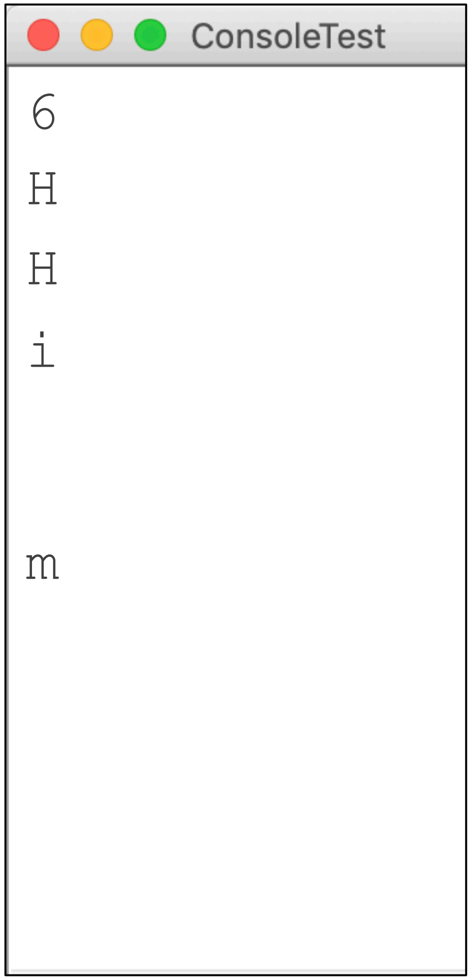


```
6
H
H
i
```

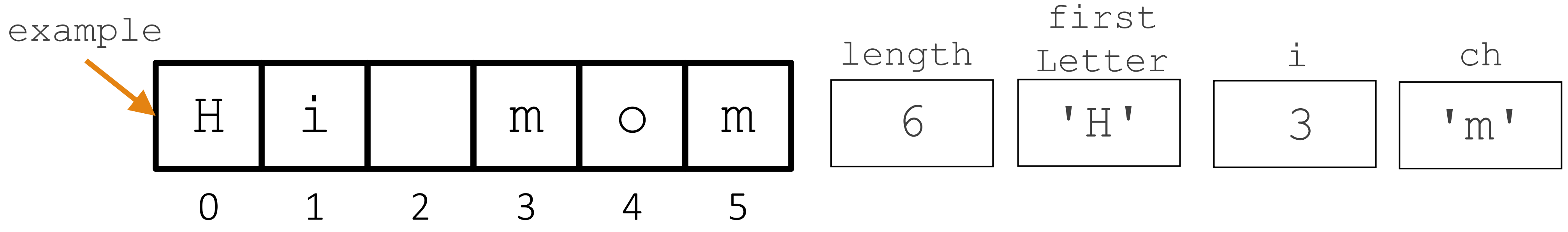


String Methods

```java
public void run() {
    String example = "Hi mom";

    // example of length method
    int length = example.length();
    println(length); // prints 6

    // example of getCharAt
    char firstLetter = example.charAt(0);
    println(firstLetter); // prints 'H'

    // loop that prints letters one-by-one
    for(int i = 0; i < example.length(); i++) {
        char ch = example.charAt(i);
        println(ch);
    }
}
```

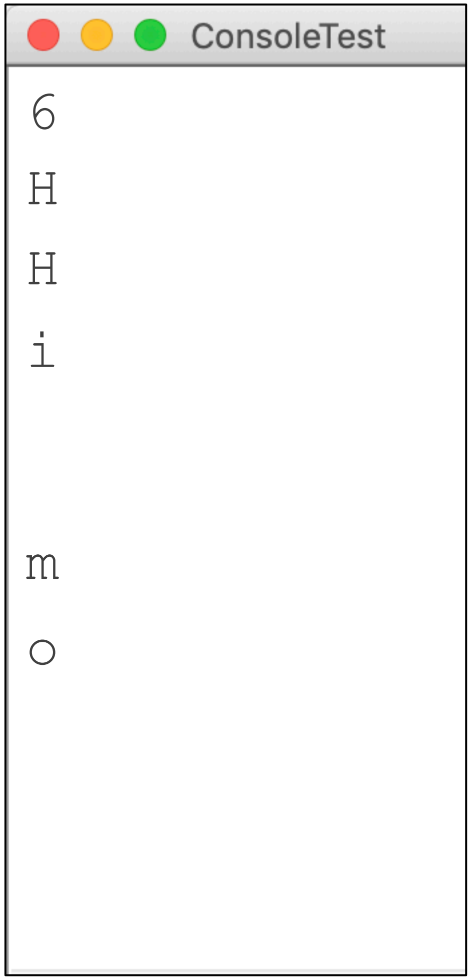


```
6
H
H
i


m
```



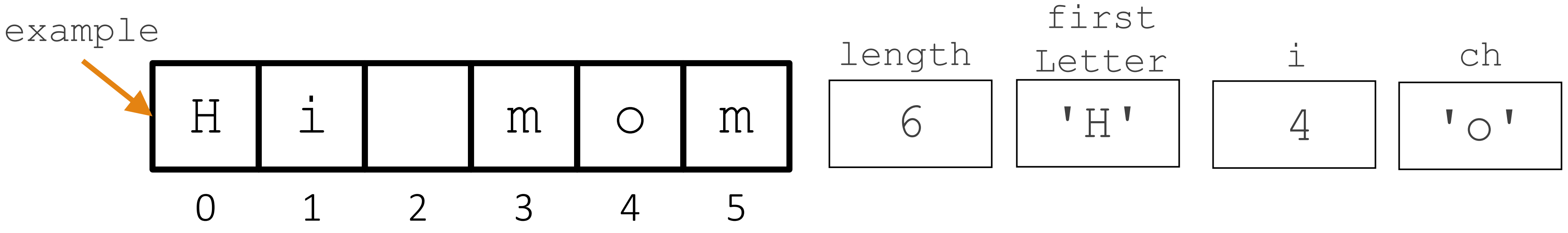
String Methods

```java
public void run() {
     String example = "Hi mom";

     // example of length method
     int length = example.length();
     println(length); // prints 6

     // example of getCharAt
     char firstLetter = example.charAt(0);
     println(firstLetter); // prints 'H’

     // loop that prints letters one-by-one
     for(int i = 0; i < example.length(); i++) {
          char ch = example.charAt(i);
          println(ch);
     }
}
```



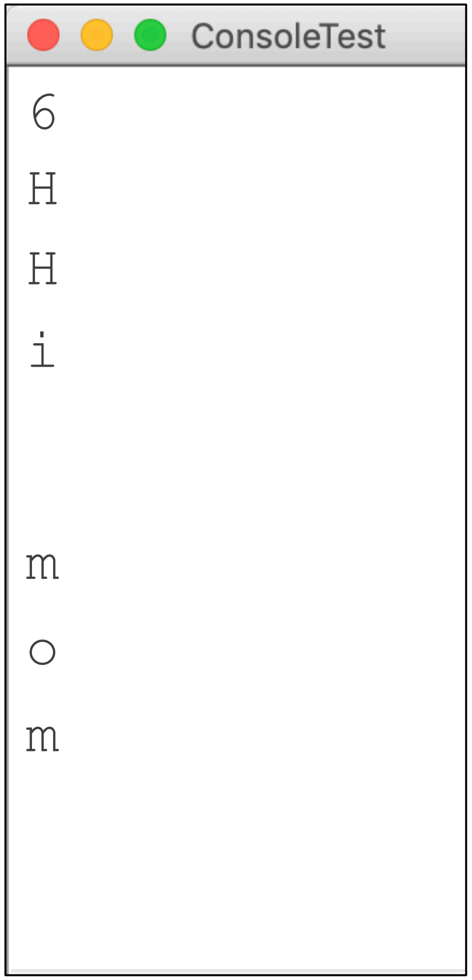
ConsoleTest

```
6
H
H
i

m
o
```

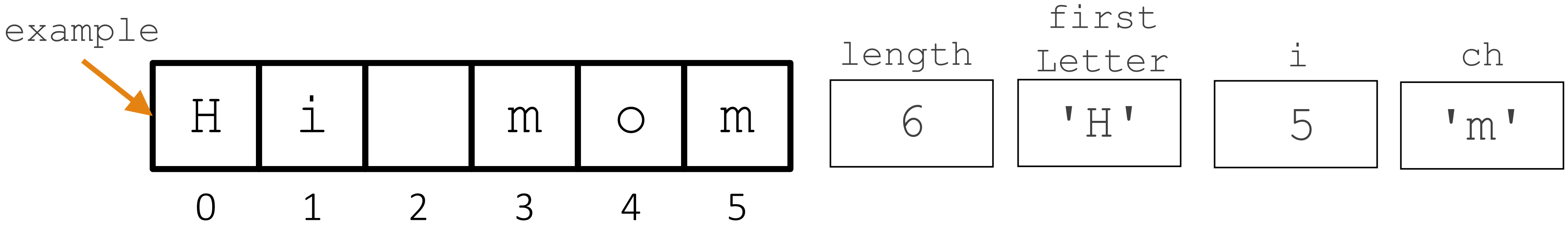


String Methods

```
public void run() {
     String example = "Hi mom";

     // example of length method
     int length = example.length();
     println(length); // prints 6

     // example of getCharAt
     char firstLetter = example.charAt(0);
     println(firstLetter); // prints 'H'

     // loop that prints letters one-by-one
     for(int i = 0; i < example.length(); i++) {
          char ch = example.charAt(i);
          println(ch);
     }
}
```

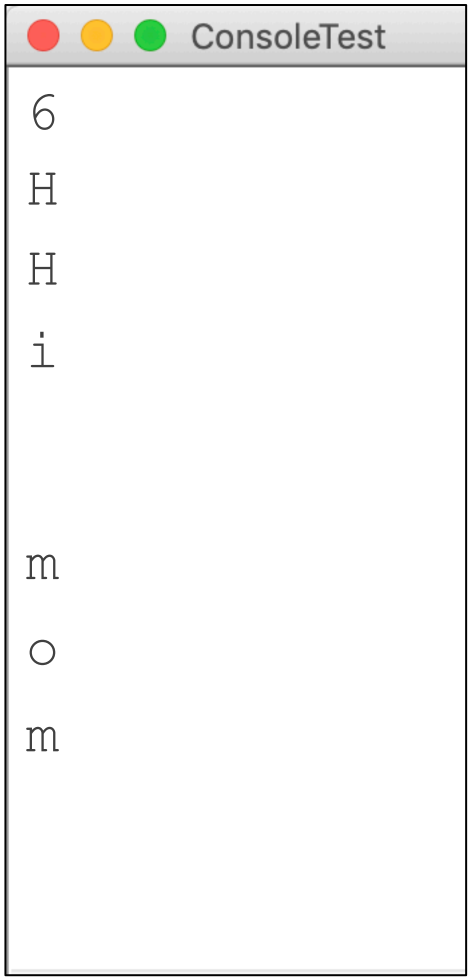


```
6
H
H
i



m
o
m
```

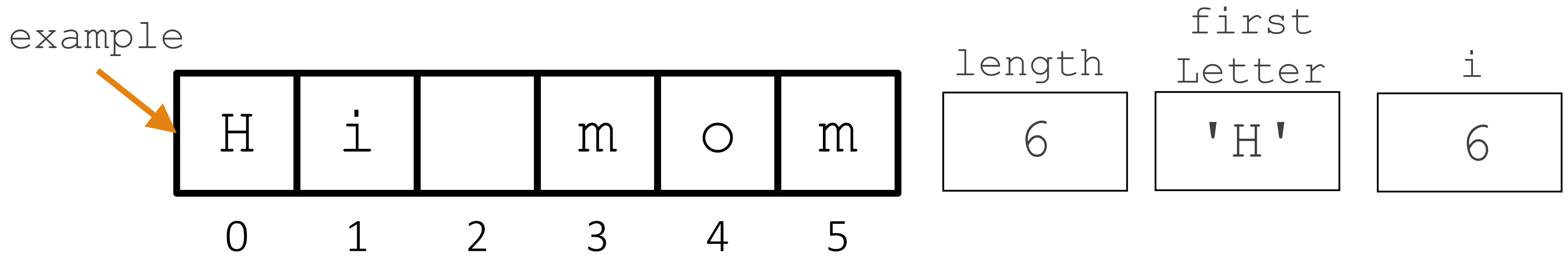


String Methods

```java
public void run() {
    String example = "Hi mom";

    // example of length method
    int length = example.length();
    println(length); // prints 6

    // example of getCharAt
    char firstLetter = example.charAt(0);
    println(firstLetter); // prints 'H'

    // loop that prints letters one-by-one
    for(int i = 0; i < example.length(); i++) {
        char ch = example.charAt(i);
        println(ch);
    }
}
```



```
6
H
H
i

m
o
m
```



How are characters represented?

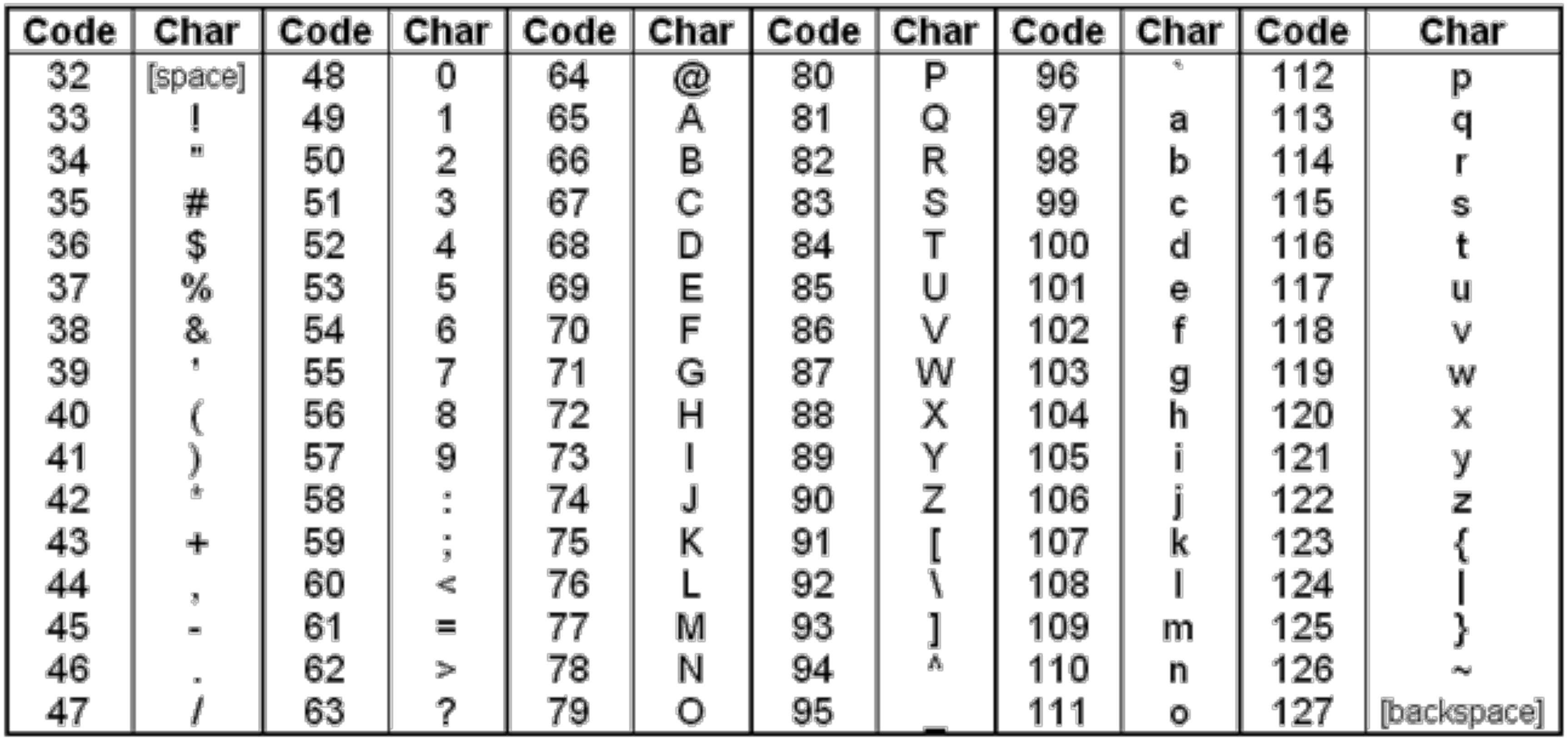
The variable type char

The primitive type **char** represents a single character or glyph.

Some examples:

```
char letterA = 'A';
char plus    = '+';
char zero    = '0';
char space   = ' ';
char newLine = '\n'; // special
char first = text.charAt(0);
```

ASCII



Code	Char	Code	Char	Code	Char	Code	Char	Code	Char	Code	Char
32	[space]	48	0	64	@	80	P	96	`	112	p
33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	'	55	7	71	G	87	W	103	g	119	w
40	(	56	8	72	H	88	X	104	h	120	x
41	)	57	9	73	I	89	Y	105	i	121	y
42	*	58	:	74	J	90	Z	106	j	122	z
43	+	59	;	75	K	91	[	107	k	123	{
44	,	60	<	76	L	92	\	108	l	124	|
45	-	61	=	77	M	93	]	109	m	125	}
46	.	62	>	78	N	94	^	110	n	126	~
47	/	63	?	79	O	95	_	111	o	127	[backspace]

* This is only the first half of the table

The letter A, for example, has the ASCII value 65

Using portions of slides by Eric Roberts

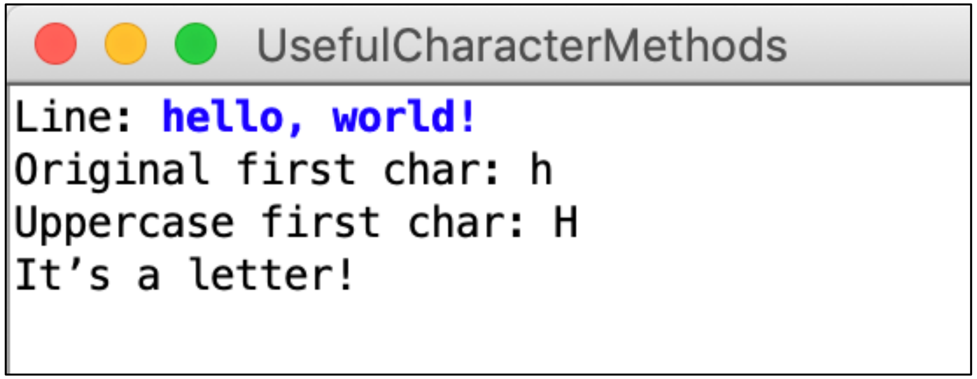
`'A'` → `'Z'` are sequential.

`'a'` → `'z'` are sequential.

`'0'` → `` `9' `` are sequential.

Useful Character methods

```
public void run() {
    String str = readLine("Line: ");

    char ch = str.charAt(0);
    println("Original first char: " + ch);

    ch = Character.toUpperCase(ch);
    println("Uppercase first char: " + ch);

    if (Character.isLetter(ch)) {
       println("It's a letter!");
    }
}
```

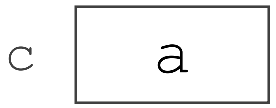


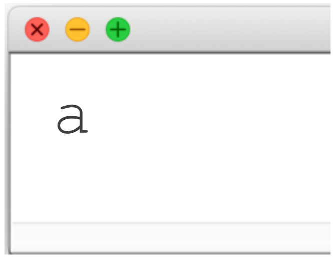
```
Line: hello, world!
Original first char: h
Uppercase first char: H
It's a letter!
```

Stay alert!

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

A

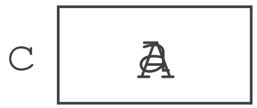


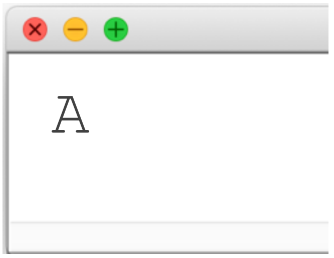


```
a
```

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

A





```
A
```

Useful Character methods

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

Using portions of slides by Eric Roberts

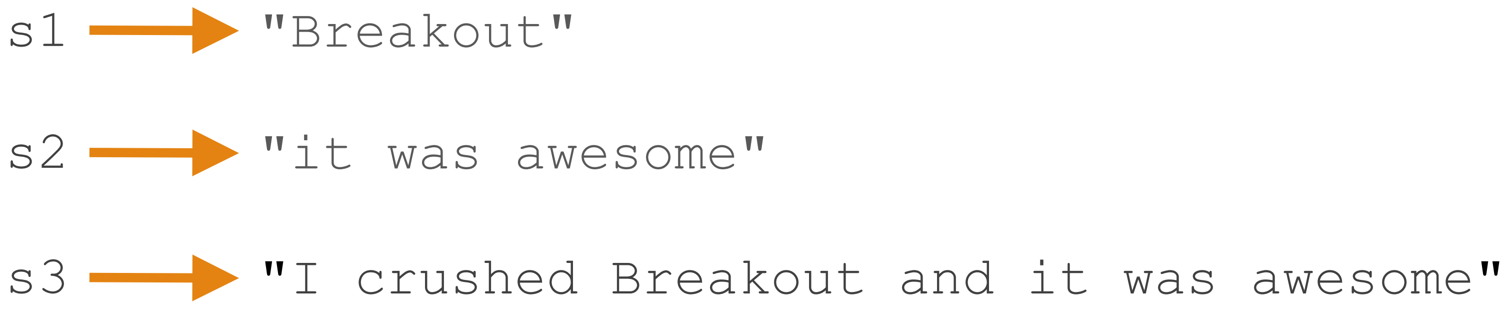
Strings have some unique properties

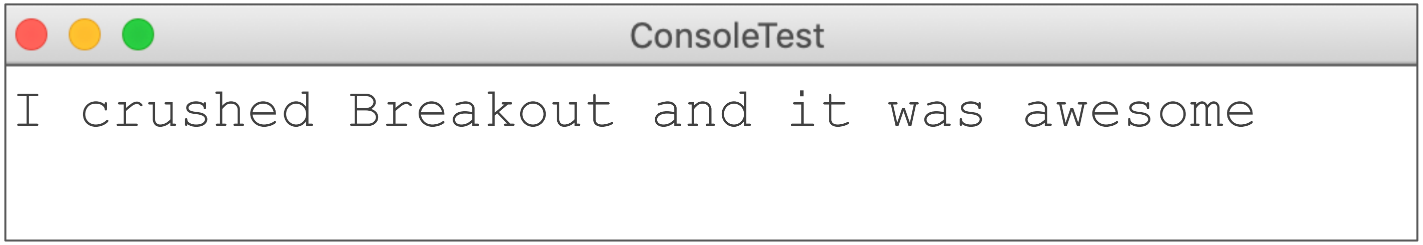
Strings are Immutable

- Java strings are *immutable*: once a string has been created **you cannot set characters**.
- To change a string:
  - *Create a new string* holding the new value you want it to have via concatenation (+).
  - Reassigning the **String** variable (that’s allowed).
- *Important consequence*: if you pass a **String** into a method, that method cannot modify that string.

Strings are often made through concatenation

```java
public void run() {
    String s1 = "Breakout";
    String s2 = "it was awesome";
    String s3 = "I crushed " + s1 + " and " + s2;

    println(s3);
}
```



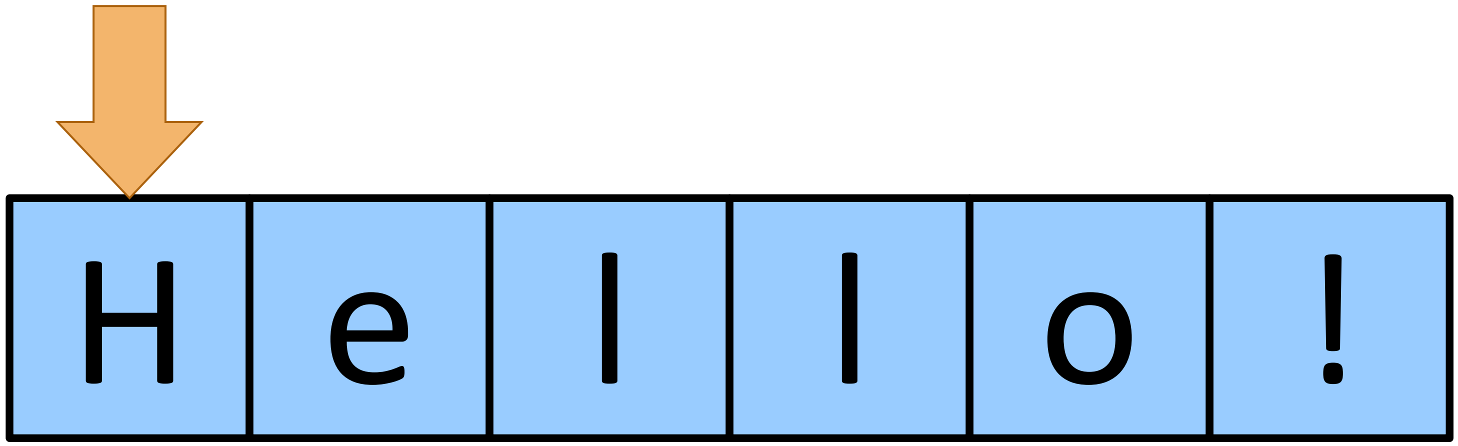


```
I crushed Breakout and it was awesome
```

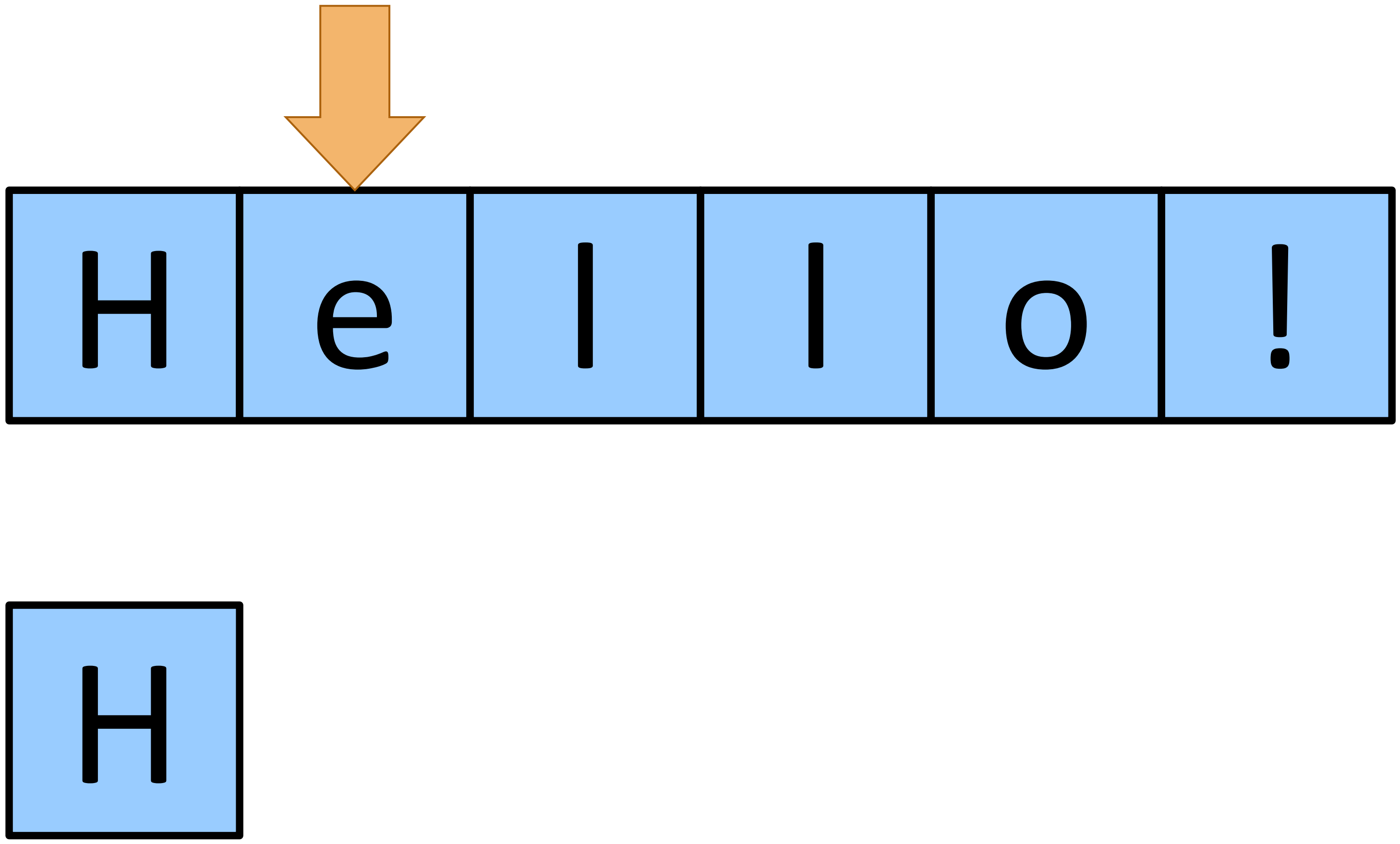
Reversing a String

Many string algorithms use the "loop and construct" pattern.

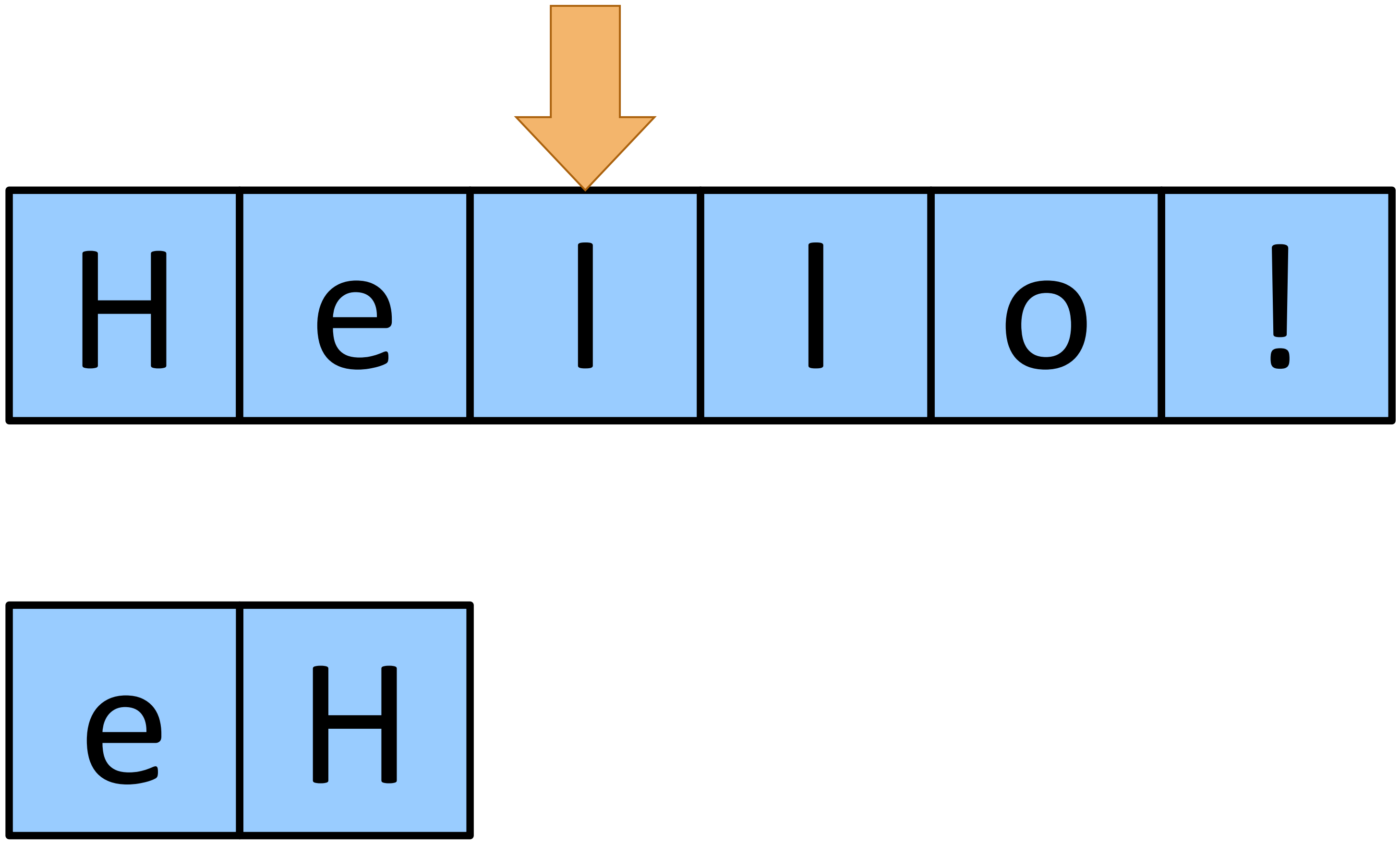
Reversing a String



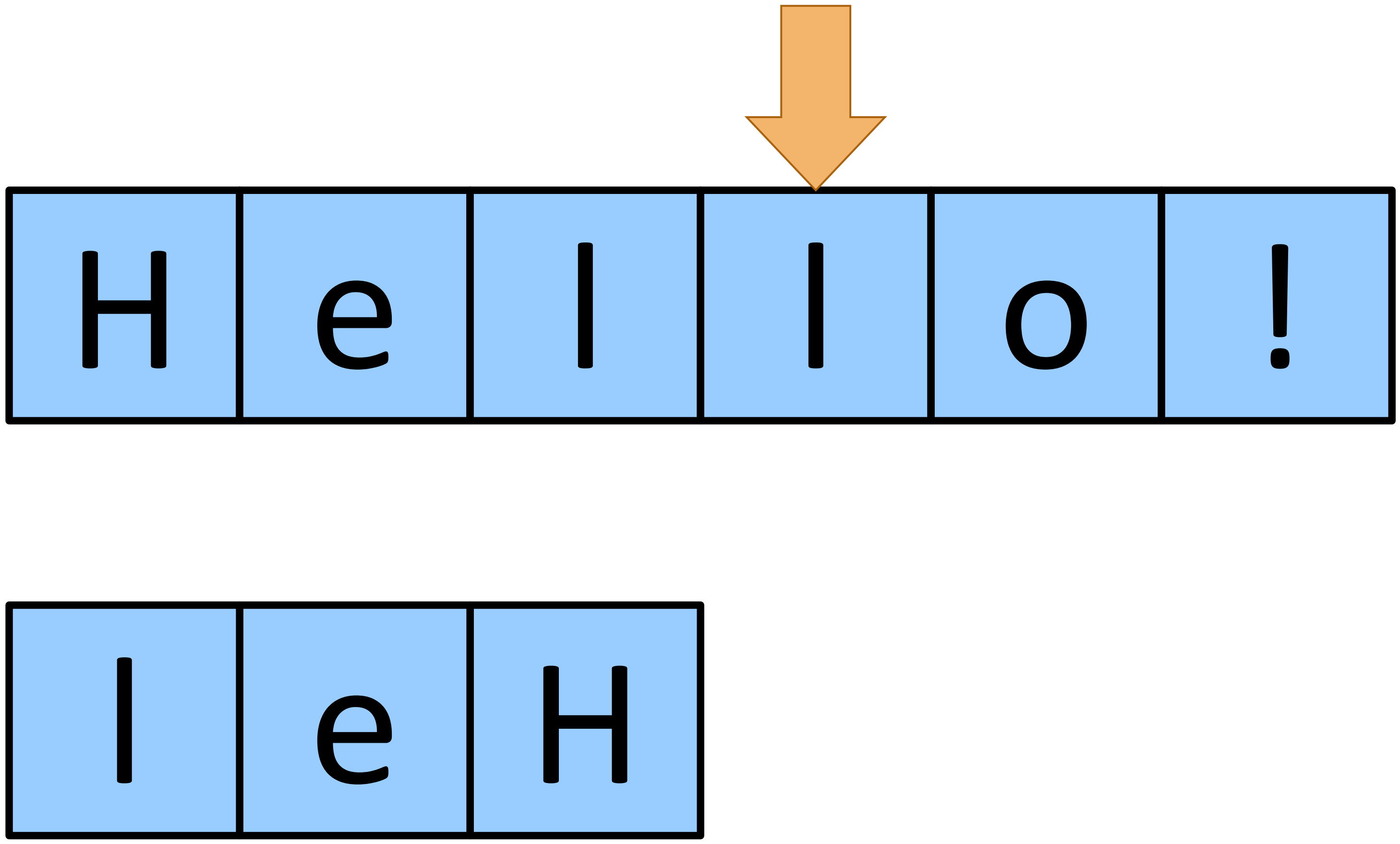
Reversing a String



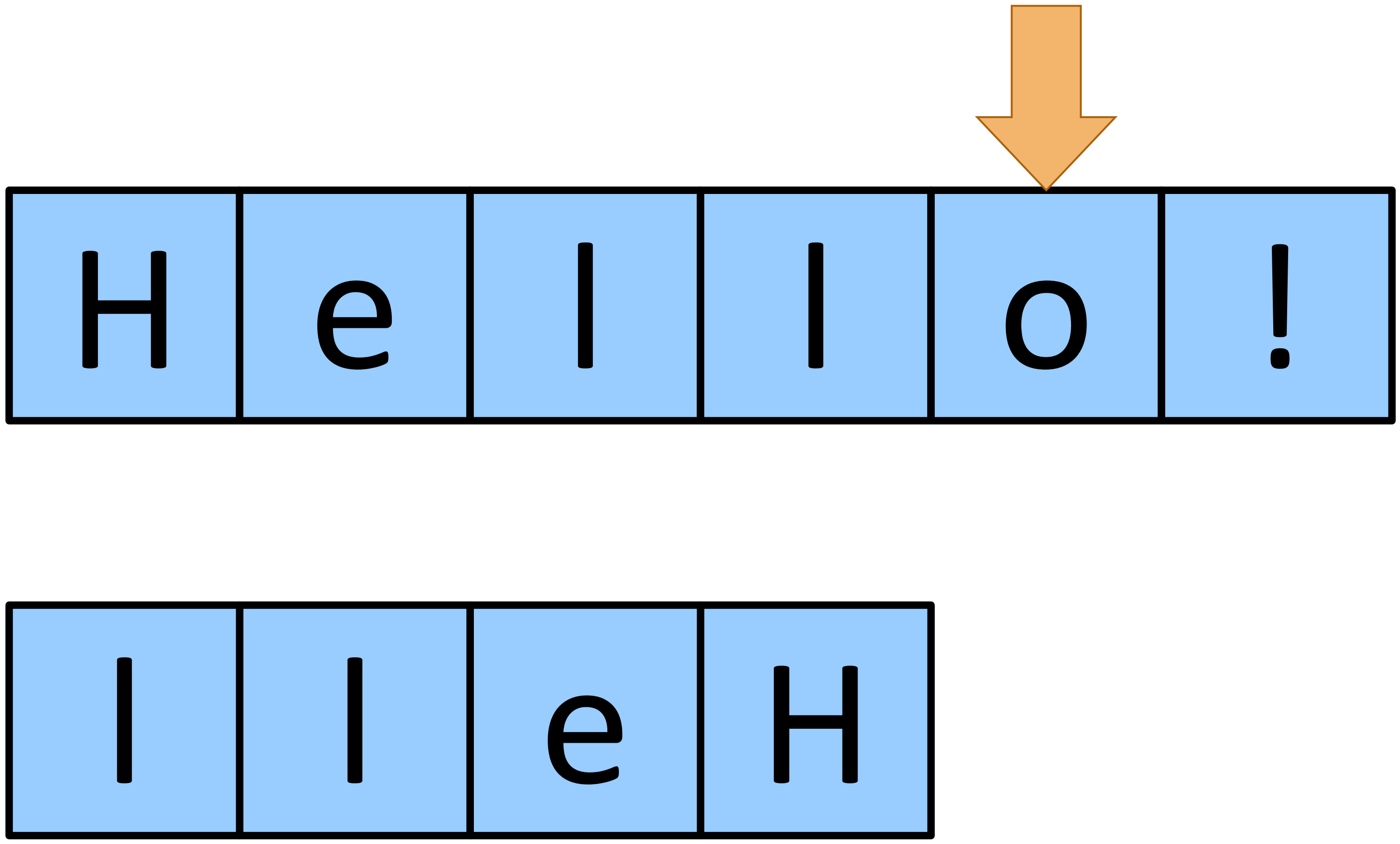
Reversing a String



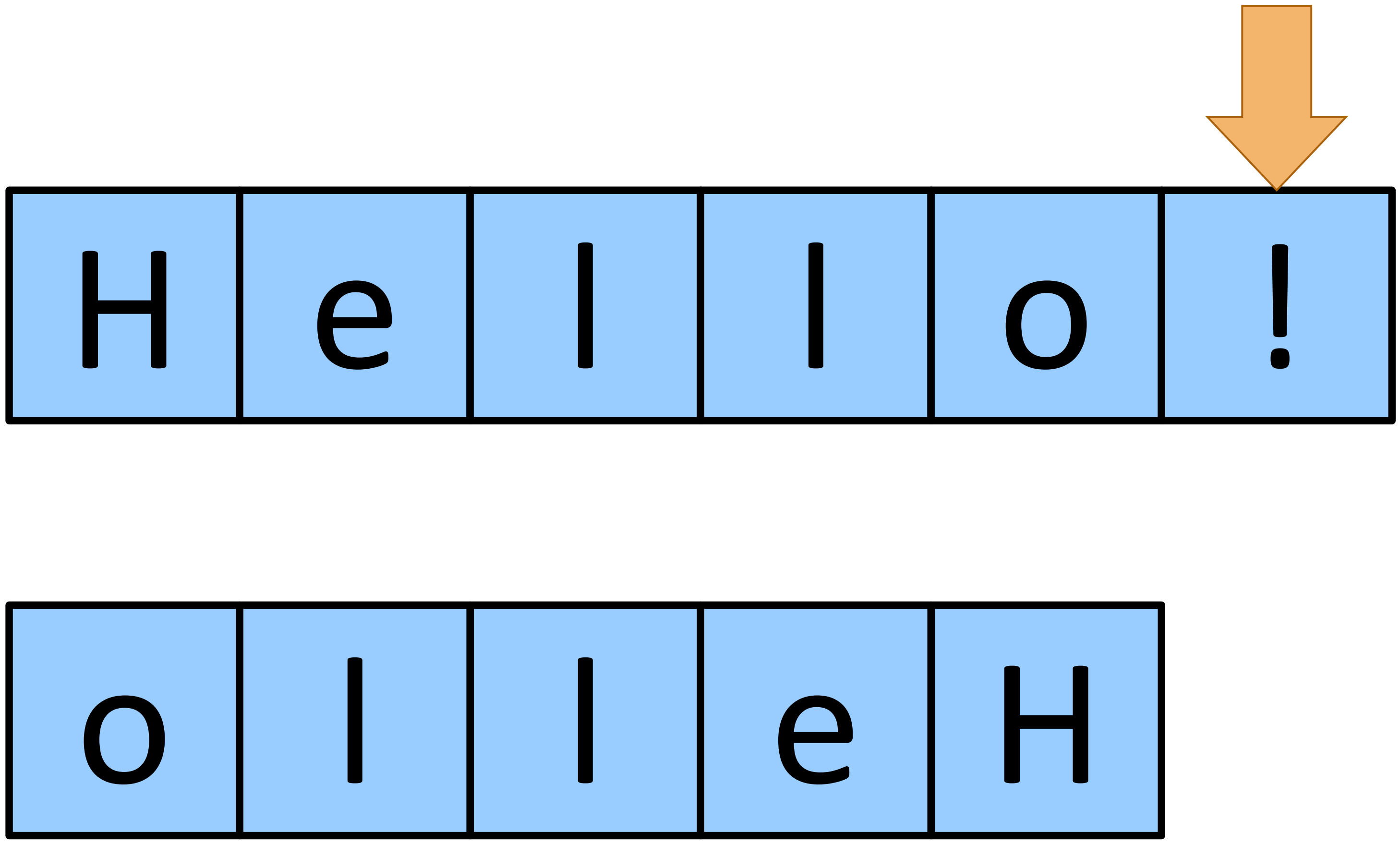
Reversing a String



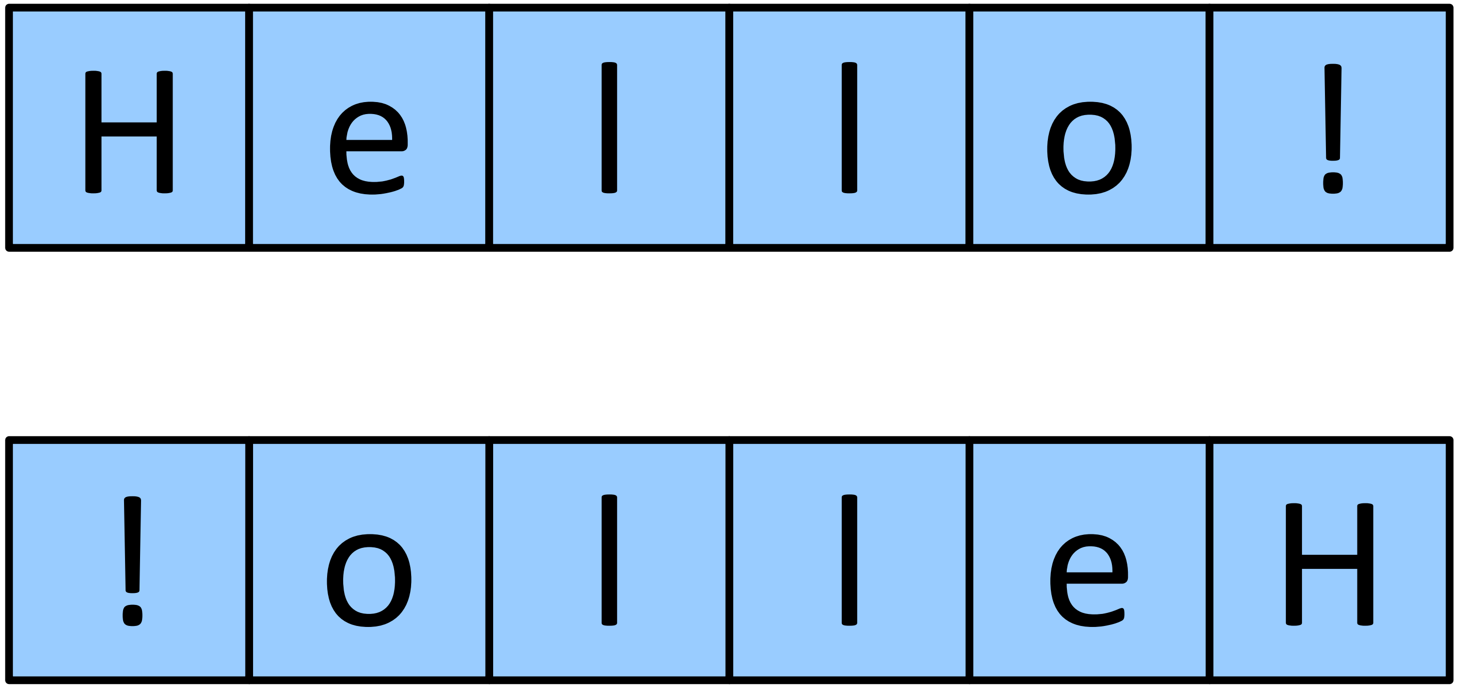
Reversing a String



Reversing a String



Reversing a String



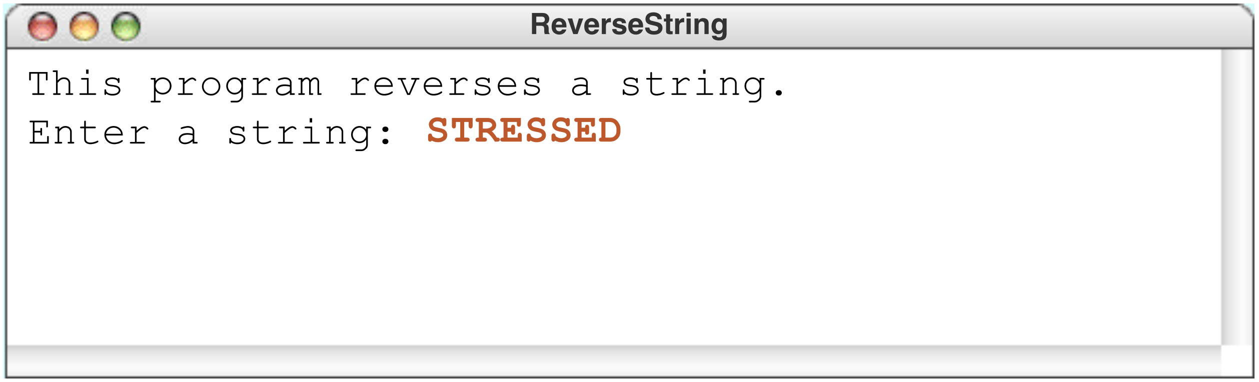
reverseString

```
public void run() {
   println("This program reverses a string.");
   String str = readLine("Enter a string: ");
   String rev = reverseString(str);
   println(str + " spelled backwards is " + rev);
}
```

rev

str

STRESSED

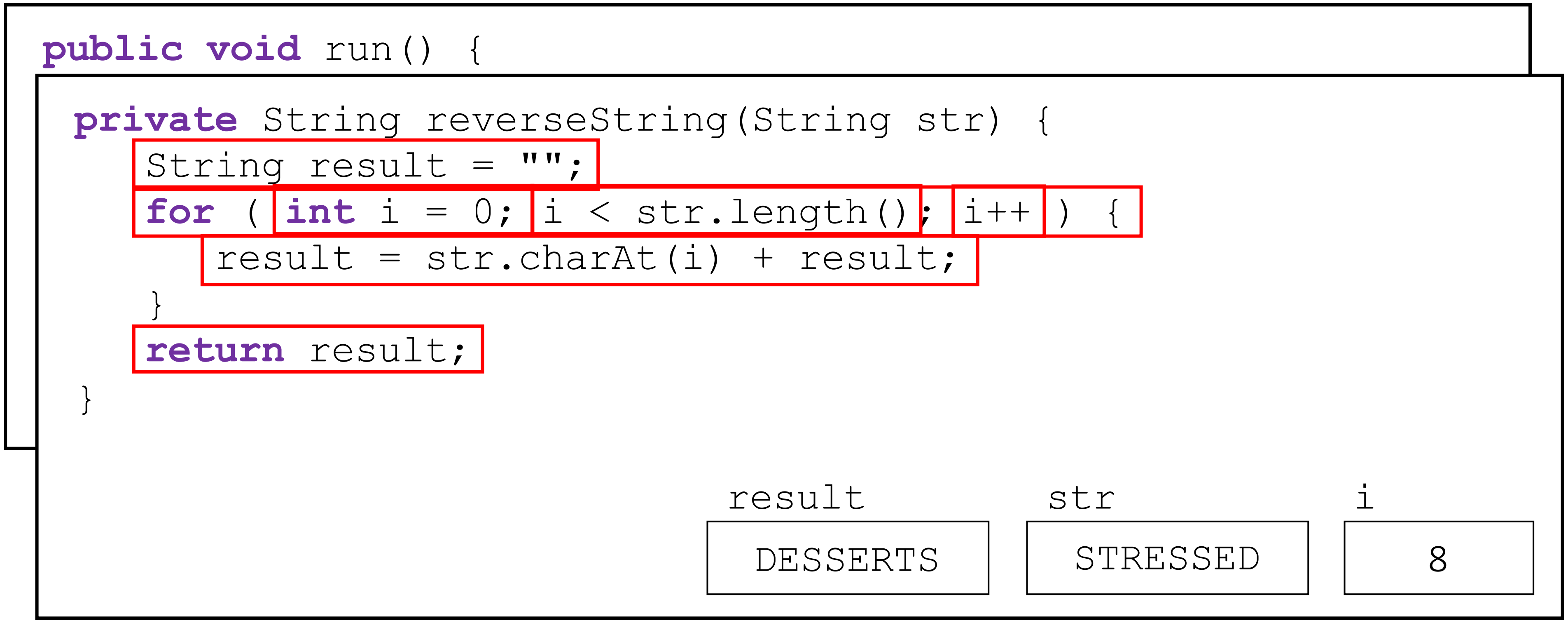


```
This program reverses a string.
Enter a string: STRESSED
```

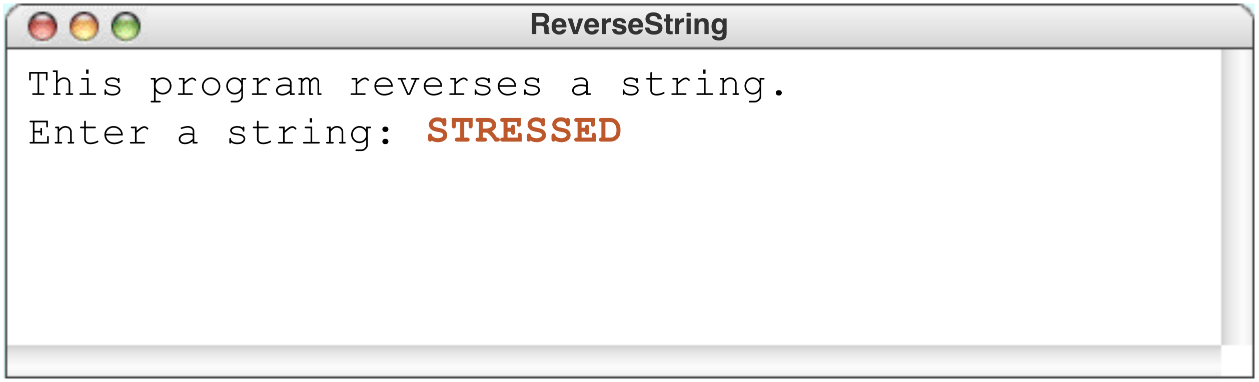
skip simulation

Using portions of slides by Eric Roberts

reverseString



```
public void run() {
   private String reverseString(String str) {
      String result = "";
      for ( int i = 0; i < str.length(); i++ ) {
         result = str.charAt(i) + result;
      }
      return result;
   }
```



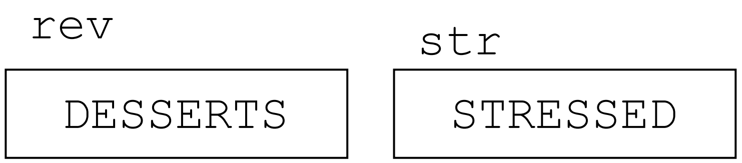
```
This program reverses a string.
Enter a string: STRESSED
```

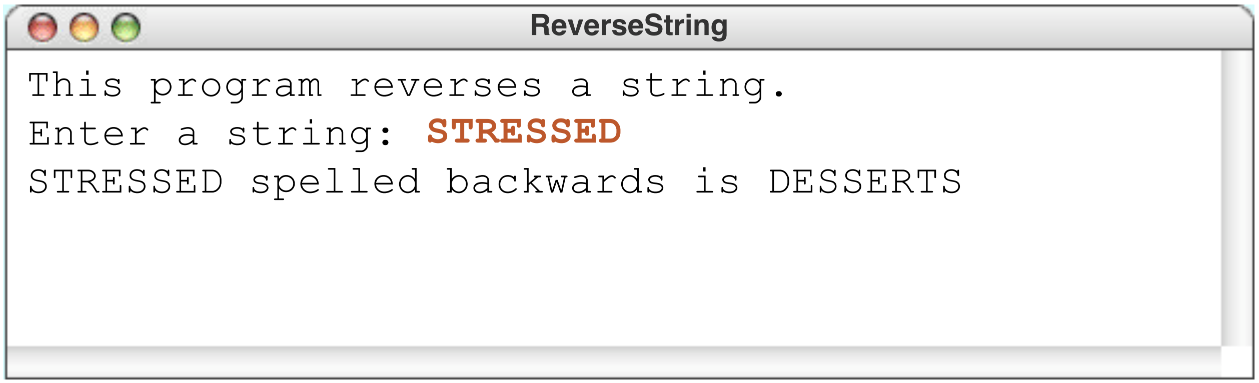
skip simulation

Using portions of slides by Eric Roberts

reverseString

```
public void run() {
   println("This program reverses a string.");
   String str = readLine("Enter a string: ");
   String rev = reverseString(str);
   println(str + " spelled backwards is " + rev);
}
```





```
This program reverses a string.
Enter a string: STRESSED
STRESSED spelled backwards is DESSERTS
```

skip simulation

Using portions of slides by Eric Roberts

Palindrome

A *palindrome* is a string that reads the same forwards and backwards.

For example:

- racecar
- kayak
- Mr. Owl ate my metal worm.
- Go hang a salami! I'm a lasagna hog.
- küçük
- Ey Edip, Adana'da pide ye.

How would we use `reverseString()` to check if a word is a palindrome?

```
private String reverseString(String str) {
   String result = "";
   for ( int i = 0; i < str.length(); i++ ) {
      result = str.charAt(i) + result;
   }
   return result;
}
```

Let’s Code it!

Equality

✗

```
private boolean isPalindrome(String original) {
    String reversed = reverseString(original);
    return reversed == original;
}
```

✓

```
private boolean isPalindrome(String original) {
    String reversed = reverseString(original);
    return reversed.equals(original);
}
```

Use `str1.equals(str2)` to compare strings, not `str1 == str2`

Some test cases

- Let's test our program on some examples:
  - racecar
  - kayak
  - go hang a salami! i'm a lasagna hog.
  - Ey Edip, Adana'da pide ye.
- Will it work?

```java
private boolean isPalindrome(String original) {
    String reversed = reverseString(original);
    return reversed.equals(original);
}
```

More Palindromes

Here are some palindromes in other languages:

- 여보, 안경 안보여 (Honey, I can't see my glasses)
- 上海自來水來自海上 (Shanghai tap water originates from "above" the ocean)
- कड़क (a loud thunderous sound)
- بلح تعلق تحت قلعة حلب (Dates hang underneath a castle in Halab)

The comedian Dmitri Martin also has a routine about palindromes; check it out at https://www.youtube.com/watch?v=0hUHDIOazIU

Useful String methods

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

Using portions of slides by Eric Roberts

Useful String Methods

```
String original = "How now brown cow";

// replace all occurrences of " " with "!"
String replaced = original.replaceAll(" ", "!");
println(replaced);
```

```
"How!now!brown!cow"
```

```
// returns true if original contains the string "cow"
println(original.contains("cow"));
```

```
true
```

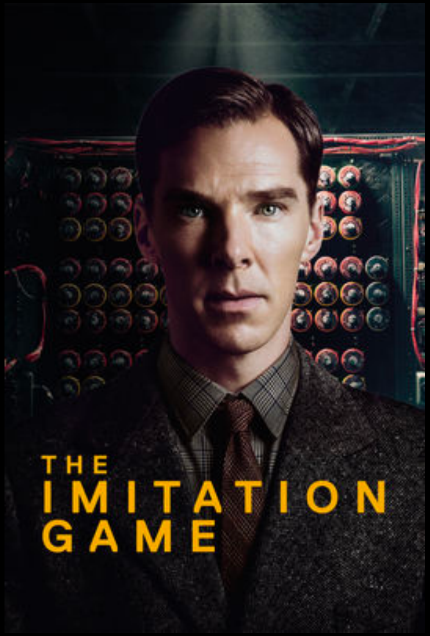
```
// String → String[], split on spaces " "
String[] strArray = original.split(" ");
for(int i = 0; i < strArray.length; i++) {
    println(strArray[i]);
}
```

```
{"How","now",
"brown","cow"}
```

```
// String[] → String, joined by ”@"
String concatenated = String.join("@", strArray);
println(concatenated); // How@now@brown@cow
```

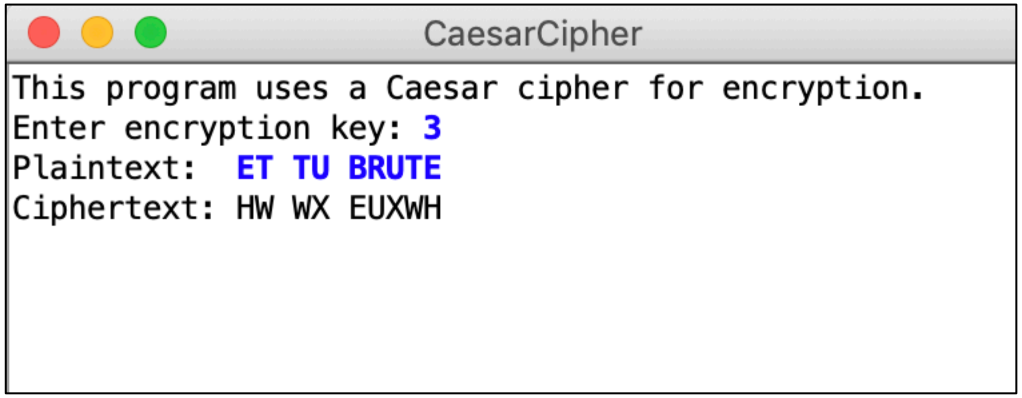
```
"How@now@brown@cow"
```

The Caesar Cipher



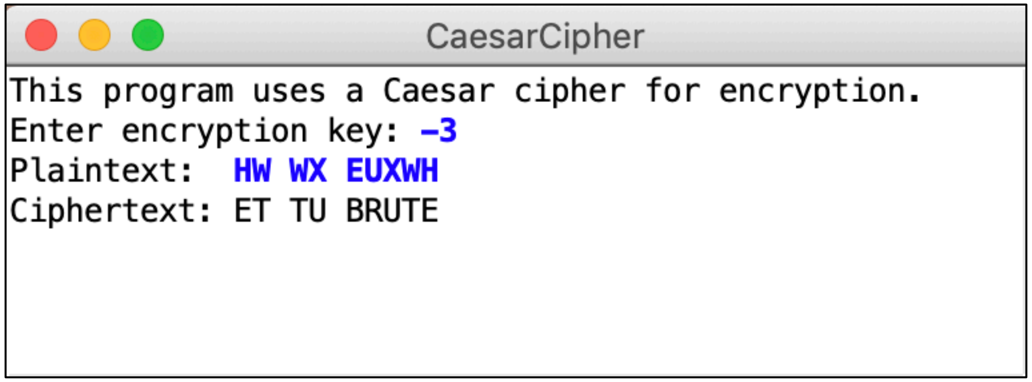
Julius Caesar
Jül Sezar

The Caesar Cipher



```
This program uses a Caesar cipher for encryption.
Enter encryption key: 3
Plaintext:  ET TU BRUTE
Ciphertext: HW WX EUXWH
```

Encrypt: shift by `key = 3`



```
This program uses a Caesar cipher for encryption.
Enter encryption key: -3
Plaintext:  HW WX EUXWH
Ciphertext: ET TU BRUTE
```

Decrypt: shift by `key = -3`

English alphabet (26 letters) `ABCDEFGHIJKLMNOPQRSTUVWXYZ`

🤔

How the cipher works

ASCII uppercase:	65	66	67	68	69	70	…	87	88	89	90
	A	B	C	D	E	F	…	W	X	Y	Z
Shifted by key=3	68	69	70	71	72	73	…	90	65	66	67
	D	E	F	G	H	I	…	Z	A	B	C

Cipher key=3

```
char ch = 'A';
char chShift = ch + 3;
println("plain : " + ch);
println("cipher: " + chShift);
```

Type mismatch: cannot convert from int to char

Java autoconverts **char** to **int**!!

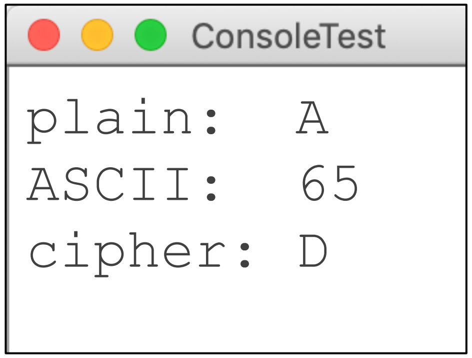
How the cipher works

ASCII uppercase:	65	66	67	68	69	70	…	87	88	89	90
	A	B	C	D	E	F	…	W	X	Y	Z

Cipher key=3

Shifted by key=3	68	69	70	71	72	73	…	90	65	66	67
	D	E	F	G	H	I	…	Z	A	B	C

```
char ch = 'A';
int chIntShift = ch + key;
char chShift = (char) chIntShift;
println("plain:  " + ch);
println("ASCII:  " + (int) ch);
println("cipher: " + chShift);
```



```
plain:   A
ASCII:   65
cipher:  D
```

Java autoconverts **char** to **int**!!

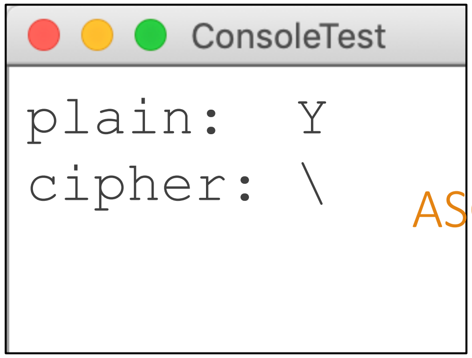
How the cipher works

ASCII	65	66	67	68	69	70	…	87	88	89	90
uppercase:	A	B	C	D	E	F	…	W	X	Y	Z

Cipher key=3

Shifted	68	69	70	71	72	73	…	90	65	66	67
by key=3	D	E	F	G	H	I	…	Z	A	B	C

```
char ch = 'Y';
int chIntShift = ch + key;
char chShift = (char) chIntShift;
println("plain:  " + ch);
println("cipher: " + chShift);
```


ConsoleTest

```
plain:   Y
cipher:  \
```

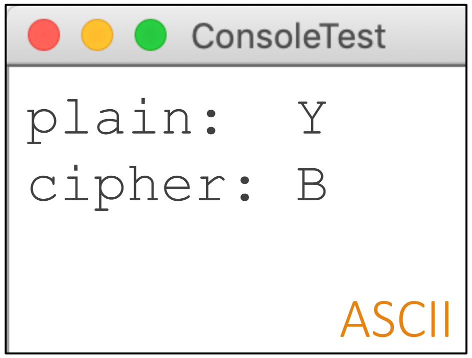
ASCII 92

How the cipher works

ASCII	65	66	67	68	69	70	… 87	88	89	90
uppercase:	A	B	C	D	E	F	… W	X	Y	Z
Shifted	68	69	70	71	72	73	… 90	65	66	67
by key=3	D	E	F	G	H	I	… Z	A	B	C

Cipher key=3

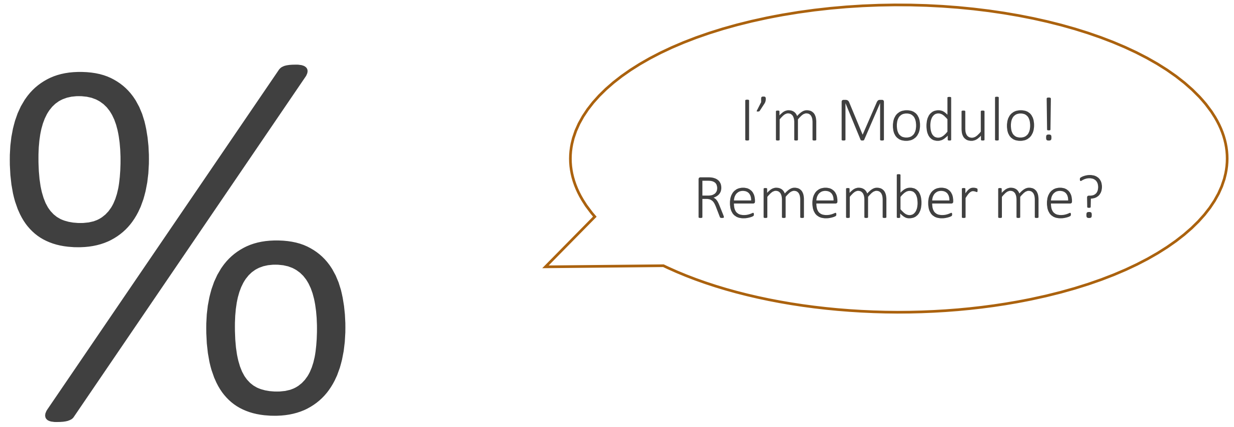
```
char ch = 'Y';
int chIntShift = ch + key;
if(chIntShift > 90) { chIntShift -= 26; }
char chShift = (char) chIntShift;
println("plain : " + ch);
println("cipher: " + chShift);
```



```
plain:  Y
cipher: B
```

ASCII
66: 92-26

How the cipher works



```
int chIntShift = 'A'  +  (ch - 'A' + key) % 26;
char chShift = (char) chIntShift;
```

```
 'Y' - 'A'                = 89 - 65= 24

 'Y' - 'A' + key          = 24 + 3 = 27

('Y' - 'A' + key) % 26              = 1

'A' + ('Y' - 'A' + key) % 26        = 66
```

ASCII	65	66	67	68	69	70	71	…	88	89	90
uppercase:	A	B	C	D	E	F	G	…	X	Y	Z

Allowing all ASCII symbols

ASCII	65	66	67	68	69	70	71	…	88	89	90
uppercase:	A	B	C	D	E	F	G	…	X	Y	Z

ASCII	97	98	99	100	101	102	103	…	120	121	122
lowercase:	a	b	c	d	e	f	g	…	x	y	z

```
if (Character.isUpperCase(ch)) {
    // shift by key
    // ensure resulting ASCII is between 65 and 90
} else if (Character.isLowerCase(ch)) {
    // shift by key
    // ensure resulting ASCII is between 97 and 122
} else {
    // do nothing
}
```

Learning Goals

1. Understand chars and Strings
2. Write methods acting on Strings
3. Learn something interesting

Your goals today

(1) Array exercise: MinMaxMean
(+ ArrayList exercises)
Submit by end of Lab 2,
even if you are not done

(2) Final Project (start by Lab 3)
Console/Graphics,
Games/Stories,
Puzzles/Adventures,
Math/Medicine/Science,
...The ArrayList goes on!
Due by 6pm tomorrow