

Building Java Programs

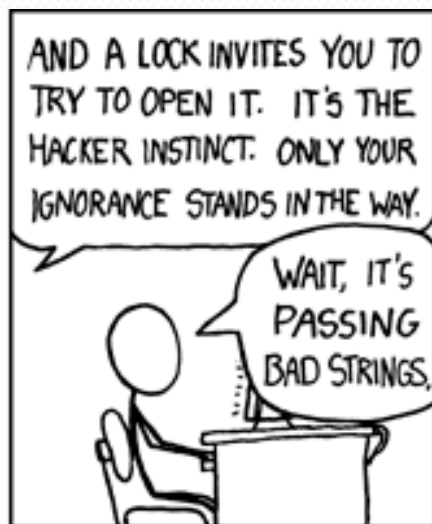
Chapter 4
Lecture 4-3: Strings, `char`

reading: 3.3, 4.3-4.4

self-check: Ch. 4 #12, 15

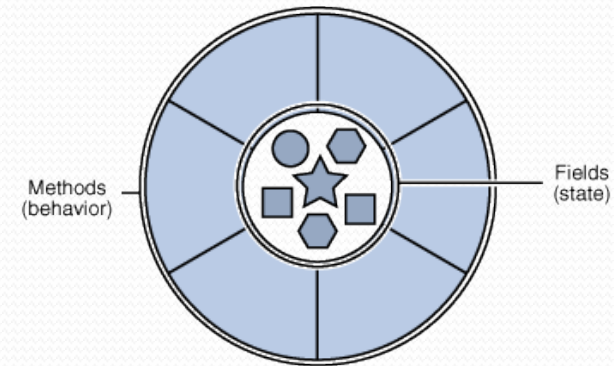
exercises: Ch. 4 #15, 16

videos: Ch. 3 #3



Objects (usage)

- **object:** An entity that contains data and behavior.
 - *data:* variables inside the object
 - *behavior:* methods inside the object
 - You interact with the methods; the data is hidden in the object.
 - A **class** is a type of objects.



- Constructing (creating) an object:
Type **objectName** = `new` **Type** (**parameters**) ;
- Calling an object's method:
objectName . **methodName** (**parameters**) ;

Strings

- **string**: An object storing a sequence of text characters.
 - Unlike most other objects, a `String` is not created with `new`.

```
String name = "text";  
String name = expression;
```

- Examples:

```
String name = "Marla Singer";  
  
int x = 3;  
int y = 5;  
String point = "(" + x + ", " + y + ")";
```

Indexes

- Characters of a string are numbered with 0-based *indexes*:

```
String name = "Ultimate";
```

index	0	1	2	3	4	5	6	7
character	U	l	t	i	m	a	t	e

- First character's index : 0
- Last character's index : 1 less than the string's length
- The individual characters are values of type `char` (seen later)

String methods

Method name	Description
<code>indexOf(str)</code>	index where the start of the given string appears in this string (-1 if not found)
<code>length()</code>	number of characters in this string
<code>substring(index1, index2)</code> or <code>substring(index1)</code>	the characters in this string from <i>index1</i> (inclusive) to <i>index2</i> (<u>exclusive</u>); if <i>index2</i> is omitted, grabs till end of string
<code>toLowerCase()</code>	a new string with all lowercase letters
<code>toUpperCase()</code>	a new string with all uppercase letters

- These methods are called using the dot notation:

```
String starz = "Yeezy & Hova";  
System.out.println(starz.length());    // 12
```

String method examples

```
// index      012345678901
String s1 = "Stuart Reges";
String s2 = "Marty Stepp";

System.out.println(s1.length());           // 12
System.out.println(s1.indexOf("e"));       // 8
System.out.println(s1.substring(7, 10));   // "Reg"

String s3 = s2.substring(1, 7);
System.out.println(s3.toLowerCase());     // "arty s"
```

- Given the following string:

```
// index      0123456789012345678901
String book = "Building Java Programs";
```

- How would you extract the word "Java" ?

Modifying strings

- Methods like `substring` and `toLowerCase` build and return a new string, rather than modifying the current string.

```
String s = "Aceyalone";  
s.toUpperCase();  
System.out.println(s);    // Aceyalone
```

- To modify a variable's value, you must reassign it:

```
String s = "Aceyalone";  
s = s.toUpperCase();  
System.out.println(s);    // ACEYALONE
```


Strings as user input

- Scanner's `next` method reads a word of input as a `String`.

```
Scanner console = new Scanner(System.in);
System.out.print("What is your name? ");
String name = console.next();
name = name.toUpperCase();
System.out.println(name + " has " + name.length() +
    " letters and starts with " + name.substring(0, 1));
```

Output:

```
What is your name? Nas
NAS has 3 letters and starts with N
```

- The `nextLine` method reads a line of input as a `String`.

```
System.out.print("What is your address? ");
String address = console.nextLine();
```

Strings question

- Write a program that reads two people's first names and suggests a name for their child

Example Output:

Parent 1 first name? **Danielle**

Parent 2 first name? **John**

Child Gender? **f**

Suggested baby name: JODANI

Parent 1 first name? **Danielle**

Parent 2 first name? **John**

Child Gender? **Male**

Suggested baby name: DANIJO

Name border

HELENE
HELEN
HELE
HEL
HE
H
HE
HEL
HELE
HELEN
HELENE
MARTIN
MARTI
MART
MAR
MA
M
MA
MAR
MART
MARTI
MARTIN

- Prompt the user for full name
- Draw out the pattern to the left
- This should be resizable. Size 1 is shown and size 2 would have the first name twice followed by last name twice

Strings answer

```
// Suggests a baby name based on parents' names.
```

```
import java.util.*;

public class BabyNamer {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        System.out.print("Parent 1 first name? ");
        String name1 = s.next();
        System.out.print("Parent 2 first name? ");
        String name2 = s.next();
        System.out.print("Child Gender? ");
        String gender = s.next();

        String halfName1 = getHalfName(name1);
        String halfName2 = getHalfName(name2);

        String name = "";
        if(gender.toLowerCase().startsWith("m")){
            name = halfName1 + halfName2;
        } else {
            name = halfName2 + halfName1;
        }
        System.out.println("Suggested name: " + name.toUpperCase());
    }
    ...
}
```

Strings answer (cont.)

```
public static String getHalfName(String name) {  
    int halfIndex = name.length() / 2;  
    String half = name.substring(0, halfIndex);  
    return half;  
}
```

Comparing strings

- Relational operators such as `<` and `==` fail on objects.

```
Scanner console = new Scanner(System.in);
System.out.print("What is your name? ");
String name = console.next();
if (name == "Lance") {
    System.out.println("Pain is temporary.");
    System.out.println("Quitting lasts forever.");
}
```

- This code will compile, but it will not print the quote.
- `==` compares objects by *references* (seen later), so it often gives `false` even when two `String`s have the same letters.

The equals method

- Objects are compared using a method named `equals`.

```
Scanner console = new Scanner(System.in);
System.out.print("What is your name? ");
String name = console.next();
if (name.equals("Lance")) {
    System.out.println("Pain is temporary.");
    System.out.println("Quitting lasts forever.");
}
```

- Technically this is a method that returns a value of type `boolean`, the type used in logical tests.

String test methods

Method	Description
<code>equals (str)</code>	whether two strings contain the same characters
<code>equalsIgnoreCase (str)</code>	whether two strings contain the same characters, ignoring upper vs. lower case
<code>startsWith (str)</code>	whether one contains other's characters at start
<code>endsWith (str)</code>	whether one contains other's characters at end
<code>contains (str)</code>	whether the given string is found within this one

```
String name = console.next();
if(name.endsWith("Kweli")) {
    System.out.println("Pay attention, you gotta listen to hear.");
} else if(name.equalsIgnoreCase("NaS")) {
    System.out.println("I never sleep 'cause sleep is the cousin of
        death.");
}
```


Type char

- `char` : A primitive type representing single characters.
 - Each character inside a `String` is stored as a `char` value.
 - Literal `char` values are surrounded with apostrophe (single-quote) marks, such as `'a'` or `'4'` or `'\n'` or `'\''`
 - It is legal to have variables, parameters, returns of type `char`

```
char letter = 'S';  
System.out.println(letter);           // S
```

- `char` values can be concatenated with strings.

```
char initial = 'P';  
System.out.println(initial + " Diddy"); // P Diddy
```

The charAt method

- The chars in a String can be accessed using the charAt method.

```
String food = "cookie";  
char firstLetter = food.charAt(0); // 'c'  
System.out.println(firstLetter + " is for " + food);  
System.out.println("That's good enough for me!");
```

- You can use a for loop to print or examine each character.

```
String major = "CSE";  
for (int i = 0; i < major.length(); i++) {  
    char c = major.charAt(i);  
    System.out.println(c);  
}
```

Output:

C
S
E

char VS. String

- "h" is a String
'h' is a char (the two behave differently)

- String is an object; it contains methods

```
String s = "h";  
s = s.toUpperCase(); // 'H'  
int len = s.length(); // 1  
char first = s.charAt(0); // 'H'
```

- char is primitive; you can't call methods on it

```
char c = 'h';  
c = c.toUpperCase(); // ERROR: "cannot be dereferenced"
```

- What is `s + 1`? What is `c + 1`?
- What is `s + s`? What is `c + c`?

char VS. int

- All `char` values are assigned numbers internally by the computer, called *ASCII* values.

- Examples:

'A' is 65, 'B' is 66, ' ' is 32

'a' is 97, 'b' is 98, '*' is 42

- Mixing `char` and `int` causes automatic conversion to `int`.

'a' + 10 is 107, 'A' + 'A' is 130

- To convert an `int` into the equivalent `char`, type-cast it.

(char) ('a' + 2) is 'c'

Comparing char values

- You can compare `char` values with relational operators:

`'a' < 'b'` and `'X' == 'X'` and `'Q' != 'q'`

- An example that prints the alphabet:

```
for (char c = 'a'; c <= 'z'; c++) {  
    System.out.print(c);  
}
```

- You can test the value of a string's character:

```
String word = console.next();  
if (word.charAt(word.length() - 1) == 's') {  
    System.out.println(word + " is plural.");  
}
```

String/char question

- A *Caesar cipher* is a simple encryption where a message is encoded by shifting each letter by a given amount.
 - e.g. with a shift of 3, $A \rightarrow D$, $H \rightarrow K$, $X \rightarrow A$, and $Z \rightarrow C$
- Write a program that reads a message from the user and performs a Caesar cipher on its letters:

Your secret message: **Brad thinks Angelina is cute**

Your secret key: 3

The encoded message: eudg wklqnv dqjholqd lv fxwh

Strings answer 1

```
// This program reads a message and a secret key from the user and  
// encrypts the message using a Caesar cipher, shifting each letter.
```

```
import java.util.*;
```

```
public class SecretMessage {  
    public static void main(String[] args) {  
        Scanner console = new Scanner(System.in);  
  
        System.out.print("Your secret message: ");  
        String message = console.nextLine();  
        message = message.toLowerCase();  
  
        System.out.print("Your secret key: ");  
        int key = console.nextInt();  
  
        encode(message, key);  
    }  
}
```

```
...
```

Strings answer 2

```
// This method encodes the given text string using a Caesar
// cipher, shifting each letter by the given number of places.
public static void encode(String text, int shift) {
    System.out.print("The encoded message: ");
    for (int i = 0; i < text.length(); i++) {
        char letter = text.charAt(i);

        // shift only letters (leave other characters alone)
        if (letter >= 'a' && letter <= 'z') {
            letter = (char) (letter + shift);

            // may need to wrap around
            if (letter > 'z') {
                letter = (char) (letter - 26);
            } else if (letter < 'a') {
                letter = (char) (letter + 26);
            }
        }
        System.out.print(letter);
    }
    System.out.println();
}
}
```


(Optional) printf

reading: 4.3

Formatting text with `printf`

```
System.out.printf("format string", parameters);
```

- A format string can contain *placeholders* to insert parameters:

- `%d` integer
- `%f` real number
- `%s` string

- these placeholders are used instead of + concatenation

- Example:

```
int x = 3;
int y = -17;
System.out.printf("x is %d and y is %d!\n", x, y);
// x is 3 and y is -17!
```

- `printf` does not drop to the next line unless you write `\n`

printf width

- `%Wd` integer, **W** characters wide, right-aligned
- `%-Wd` integer, **W** characters wide, *left*-aligned
- `%Wf` real number, **W** characters wide, right-aligned
- ...

```
for (int i = 1; i <= 3; i++) {  
    for (int j = 1; j <= 10; j++) {  
        System.out.printf("%4d", (i * j));  
    }  
    System.out.println();    // to end the line  
}
```

Output:

1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30

printf precision

- `%.Df` real number, rounded to **D** digits after decimal
- `%.W.Df` real number, **W** chars wide, **D** digits after decimal
- `%-W.Df` real number, **W** wide (left-align), **D** after decimal

```
double gpa = 3.253764;  
System.out.printf("your GPA is %.1f\n", gpa);  
System.out.printf("more precisely: %8.3f\n", gpa);
```

Output:

```
your GPA is 3.3  
more precisely: 3.254
```

3
8

printf question

- Modify our `Receipt` program to better format its output.
 - Display results in the format below, with 2 digits after .
- Example log of execution:

How many people ate? 4

Person #1: How much did your dinner cost? 20.00

Person #2: How much did your dinner cost? 15

Person #3: How much did your dinner cost? 25.0

Person #4: How much did your dinner cost? 10.00

Subtotal: \$70.00

Tax: \$5.60

Tip: \$10.50

Total: \$86.10

printf answer (partial)

...

```
// Calculates total owed, assuming 8% tax and 15% tip
```

```
public static void results(double subtotal) {  
    double tax = subtotal * .08;  
    double tip = subtotal * .15;  
    double total = subtotal + tax + tip;  
  
    // System.out.println("Subtotal: $" + subtotal);  
    // System.out.println("Tax: $" + tax);  
    // System.out.println("Tip: $" + tip);  
    // System.out.println("Total: $" + total);  
  
    System.out.printf("Subtotal: $%.2f\n", subtotal);  
    System.out.printf("Tax: $%.2f\n", tax);  
    System.out.printf("Tip: $%.2f\n", tip);  
    System.out.printf("Total: $%.2f\n", total);  
}  
}
```