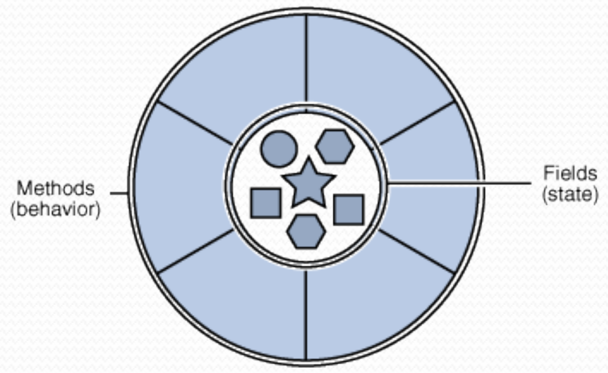
Building Java Programs

Chapter 4
Lecture 4-3: Strings; `printf`

reading: 3.3, 4.3

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 = "Glen Hansard";

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

- These methods are called using the dot notation:

```
String popStarz = "Prince vs. Michael";
System.out.println(popStarz.length());    // 18
```

`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 = "Mumford & Sons";
s.toUpperCase();
System.out.println(s);    // Mumford & Sons
```

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

```
String s = "Mumford & Sons";
s = s.toUpperCase();
System.out.println(s);    // MUMFORD & SONS
```

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? Bono
BONO has 4 letters and starts with B
```

- 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 outputs “The Name Game” with a person’s first and last name.

Example Output:

```
What is your name? James Joyce

James, James, bo-bames
Banana-fana fo-fames
Fee-fi-mo-mames
JAMES!

Joyce, Joyce, bo-boyce
Banana-fana fo-foyce
Fee-fi-mo-moyce
JOYCE!
```

Strings answer

```
// This program prints "The Name Game".
import java.util.*;

public class TheNameGame {
    public static void main(String[] args) {
        Scanner console = new Scanner(System.in);
        System.out.print("What is your name? ");
        String name = console.nextLine();

        int spaceIndex = name.indexOf(" ");
        String firstName = name.substring(0, spaceIndex);
        String lastName = name.substring(spaceIndex + 1);

        singSong(firstName);
        singSong(lastName);
    }
```

Strings answer (cont.)

```
 public static void singSong(String name) {
        System.out.println();
        String allButLast = name.substring(1);
        System.out.println(name + ", " + name + ", bo-b" + allButLast);
        System.out.println("Banana-fana fo-f" + allButLast);
        System.out.println("Fee-fi-mo-m" + allButLast);
        System.out.println(name.toUpperCase() + "!");
    }
}
```

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 song.

  - `==` 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
`equals(`**str**`)`	whether two strings contain the same characters
`equalsIgnoreCase(`**str**`)`	whether two strings contain the same characters, ignoring upper vs. lower case
`startsWith(`**str**`)`	whether one contains other's characters at start
`endsWith(`**str**`)`	whether one contains other's characters at end
`contains(`**str**`)`	whether the given string is found within this one

```
String name = console.next();
if (name.endsWith("Yeats")) {
    System.out.println("Say my glory was I had such friends.");
} else if (name.equalsIgnoreCase("OSCAR WILDE")) {
    System.out.println("A true friend stabs you in the front.");
}
```

`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

- `%`**W**`d` integer, **W** characters wide, right-aligned
- `%-`**W**`d` integer, **W** characters wide, *left*-aligned
- `%`**W**`f` 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

- `%.`**`D`**`f` real number, rounded to **D** digits after decimal
- `%`**`W`**`.`**`D`**`f` real number, **W** chars wide, **D** digits after decimal
- `%-`**`W`**`.`**`D`**`f` 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
```

$\underbrace{\texttt{   3.}\overbrace{\texttt{254}}^{3}}_{8}$

`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);
}
}
```