Principles of Programming

Module 5: Methods

Methods as Black Boxes, Implementing Method

Today’s Objectives

- Topics
  - Review Test
  - 5.1 Methods as Black Boxes
  - 5.2 Implementing Methods

Last Lecture *on a Page*

PowerTable.java
RandomDemo.java
Dice.java
CalculatePI.java

1: Sum and Average Value
2: Counting Matches
3: Finding the First Match
4: Prompting until a match is found
5: Maximum and Minimum
6: Comparing Adjacent Values

```
int upperCaseLetters = 0;
for (int i = 0; i < str.length(); i++) {
   char ch = str.charAt(i);
   if (Character.isUpperCase(ch)) {
      upperCaseLetters++;
   }
}
```

Length = 10
Last position: Length – 1 (that is, 9)

0	1	2	3	4	5	6	7	8	9
P	r	o	f	.		P	l	u	m

Character
+isDigit(c: char): boolean +isLetter(c: char): boolean +isUpperCase(c: char): boolean +isLowerCase(c: char): boolean +isWhiteSpace(c: char): boolean

- <u>Max/Min Strategy:</u>
  - Get **first** input value, this is your **best** answer so far.
  - Loop through the rest of the values, **looking to beat** your current winner.

- <u>Get a Random Number:</u>
  - Get a double from 0.0 → 1.00 `double r = Math.random();`
  - Pick a number from 1 to 6 `int die1 = (int) (Math.random() * 6) + 1;`

Flow of Control Scorecard

- The order in which a program performs actions.
  - Sequence: a list of ordered statements
    - compound statement { } ✔
  - Branching: optional execution based on condition
    - if/then/else statement ✔
    - Switch/case statement ✔
  - Looping: continual execution while a condition is true
    - while loop ✔
    - for loop ✔
    - do loop ✔
  - Subroutines: Transfer control to a different function, then return
    - Method call
  - Program Halt
    - System.exit(0); ✔

YOU ARE HERE

Method Introduction

- A method is a set of instructions with a name.
  - A method is declared *inside* a class.

```
import java.util.Scanner;

public class Example {
   public static void main(String[] args) {
      int y = f(10);
   }

   public static int f (int x) {
      int val = x * x + 1;
      return val;
   }
}
```

- You <u>declare</u> a method by defining a named block of code.
- You <u>call</u> a method in order to execute its instructions.

- It is like a function in math:
  - Define: $f(x) = x^2 + 1$
  - What is the value of y? $y = f(10)$

Method Introduction

- You have already used some methods:

Scanner
+nextInt(): int +nextDouble(): double +next(): String +close(): void +hasNextInt(): boolean +hasNextDouble(): boolean

Character
+isDigit(c: char): boolean +isLetter(c: char): boolean +isUpperCase(c: char): boolean +isLowerCase(c: char): boolean +isWhiteSpace(c: char): boolean +toUpperCase(c: char): char +toLowerCase(c: char): char

String
+length(): int +charAt(int): char +substring(int, int): String +substring(int): String +compareTo(String): int +equals(String): boolean

Math
+PI: double
+sqrt(double): double +pow(double, double): double +random(): double

- Examples:

```
String str = "Prof. Plum";
int size = str.length();
char ch = str.charAt(0);
if (Character.isUpperCase(ch)) {
double y = Math.pow(2, 3);
```

- A Method call is made up of:
  - A Class name (Character, Math) or an object name (str),
  - A dot (.) separating the class/object from the method name,
  - A method name (starting with lowercase),
  - ( ) a set of parenthesis to provide input information.
  - A method *may* return a value.
  - Scanner.close() does not!

SampleMethod.java

Method Example

- Define: $f(x) = x^2 + 1$
- What is the value of: $f(4) + 2 \times f(3)$

Class named: **SampleMethod**

```
public class SampleMethod {
   public static void main(String[] args) {
      System.out.println("Testing a method call");
      System.out.println("The answer is: " + (f(4) + 2 * f(3)));
   }

   private static int f(int x) {
      return x * x + 1;
   }
}
```

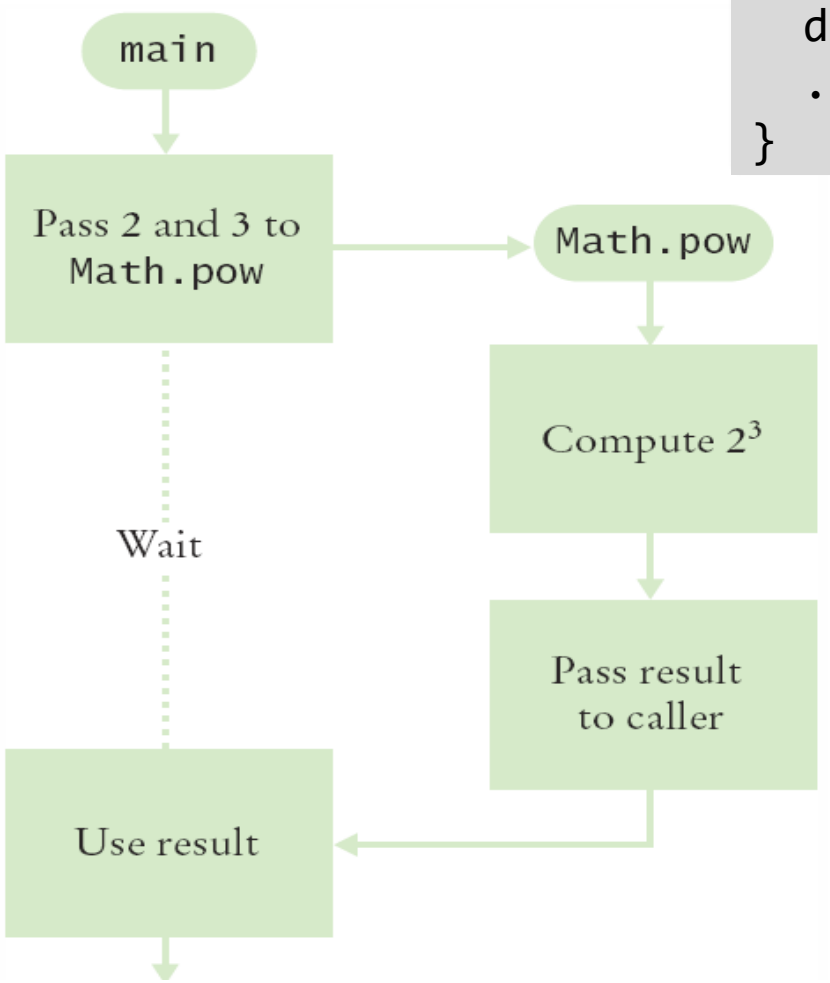
Method named: **main**

Method named: **f**

```
Testing a method call
The answer is: 37
```

Flowchart of Calling a Method

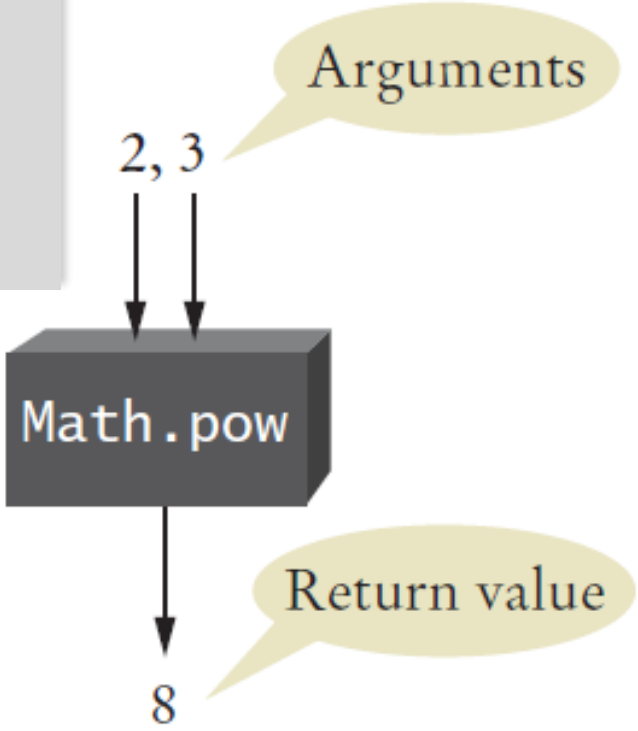
```
public static void main(String[] args)
{
  double result = Math.pow(2, 3);
  . . .
}
```



- One method ‘calls’ another
  - main calls `Math.pow`()
  - Passes two arguments (input information)
    - 2 and 3
  - `Math.pow` starts
    - Uses variables (2, 3)
    - Does its job
    - Returns the answer
  - main uses result

Arguments and Return Values

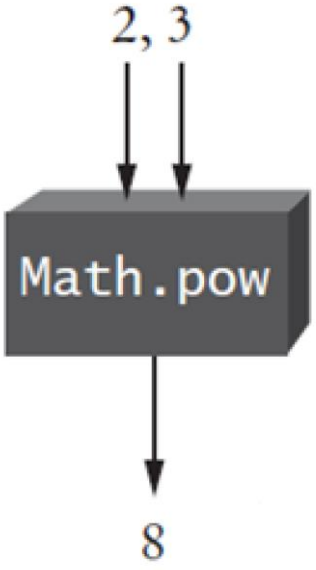
```
public static void main(String[] args)
{
  double result = Math.pow(2,3);
  . . .
}
```



- `main` 'passes' two arguments (2 and 3) to `Math.pow`
- `Math.pow` calculates and returns a value of 8 to `main`
- `main` stores the return value to variable 'result'

Black Box Analogy

- A thermostat is a ‘black box’.
  - Set a desired temperature.
  - Turns on heater/AC as required.
  - You don’t have to know how it really works!
    - How does it know the current temp?
    - What signals/commands does it send to the heater or A/C?



- Use methods like ‘black boxes’.
  - Pass the method what it needs to do its job.
  - Receive the answer.

- How does Math.pow() work?
  - *I don't care.*
  - I provide the input, it returns an answer.

Implementing Methods

- Design a method to calculate the volume of a cube.
  - What information does it need to do its job?
    - What is the type of the information?
  - What type of value does it answer with?

- When writing this method:
  - Pick a name for the method:
  - Declare a variable for each incoming argument (parameter variables):
  - Specify the type of the return value:
  - Add modifiers such as:

```
public static double cubeVolume(double sideLength)
```

  - Notice that the parameter variable goes *inside* the ( ).

Inside the Box

- Then write the body of the method
  - The body is surrounded by curly braces `{ }`
  - The body contains any additional variable declarations & statements that are executed when the method is called.
  - It will also return the calculated answer.

in

```
public static double cubeVolume(double sideLength)
{
  double volume = sideLength * sideLength * sideLength;
  return volume;
}
```

out

New keyword: `return`

The type of the return value (`volume`) better match the type required by the method (`double`)!

Back from the Box

- The values returned from `cubeVolume` are stored in local variables inside `main`.
- The results are then printed out.

```
public static void main(String[] args)
{
   double result1 = cubeVolume(2);
   double result2 = cubeVolume(10);

   System.out.println("A cube of side length 2 has volume "  + result1);
   System.out.println("A cube of side length 10 has volume " + result2);
}
```

Cubes.java

Cubes.java

```
public class Cubes {

    public static void main(String[] args) {
        double result1 = cubeVolume(2);
        double result2 = cubeVolume(10);

        System.out.println("A cube with side length 2 has volume " + result1);
        System.out.println("A cube with side length 10 has volume " + result2);
    }

    private static double cubeVolume(double sideLength) {
        double volume = sideLength * sideLength * sideLength;
        return volume;
    }
}
```

```
A cube with side length 2 has volume 8.0
A cube with side length 10 has volume 1000.0
```

Number Talker

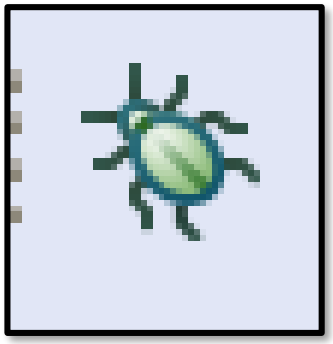
NumberTalker.java

```java
import java.util.Scanner;

public class NumberTalker {

   public static void main(String[] args) {
      Scanner in = new Scanner(System.in);
      int hundreds, tens, ones;

      System.out.print("Enter a three-digit number: ");
      int num = in.nextInt();

      hundreds = num / 100;
      num = num - (100 * hundreds);
      tens = num / 10;
      num = num - (10 * tens);
      ones = num;

      printDigit(hundreds);
      printDigit(tens);
      printDigit(ones);

      in.close();
   }

   public static void printDigit(int digit) {
      String digitName;

      switch (digit) {
         case 0: digitName = "zero";  break;
         case 1: digitName = "one";   break;
         case 2: digitName = "two";   break;
         case 3: digitName = "three"; break;
         case 4: digitName = "four";  break;
         case 5: digitName = "five";  break;
         case 6: digitName = "six";   break;
         case 7: digitName = "seven"; break;
         case 8: digitName = "eight"; break;
         case 9: digitName = "nine";  break;
         default:digitName = "";      break;
      }
      System.out.print(digitName + " - ");
   }

}
```

```
Enter a three-digit number: 369
three - six - nine -
```

NameFormatter.java

```java
import java.util.Scanner;

public class NameFormatter {

   public static void main(String[] args) {
      Scanner in = new Scanner(System.in);

      System.out.print("Your first name? ");
      String first = in.next();
      System.out.print("Your last name? ");
      String last = in.next();

      String finalName = prettyName(first, last);
      System.out.print("Official Name: " + finalName);

      in.close();
   }

   public static String prettyName(String f, String l) {
      String consName ="";

      String fLower = f.toLowerCase();
      String lLower = l.toLowerCase();

      consName = consName + Character.toUpperCase(lLower.charAt(0));
      consName = consName + lLower.substring(1);
      consName = consName + ", ";
      consName = consName + Character.toUpperCase(fLower.charAt(0));
      consName = consName + fLower.substring(1);

      return consName;
   }
}
```



Let's use the debugger!

```
Your first name? JOE
Your last name? sMiTh
Official Name: Smith, Joe
```

Character
+isDigit(c: char): boolean +isLetter(c: char): boolean +isUpperCase(c: char): boolean +isLowerCase(c: char): boolean +isWhiteSpace(c: char): boolean +toUpperCase(c: char): char +toLowerCase(c: char): char

String
+length(): int +charAt(int): char +substring(int, int): String +substring(int): String +compareTo(String): int

DiceWar.java

```java
public class DiceWar {

   public static void main(String[] args) {
      Scanner in = new Scanner(System.in);
      String choice;

      do {
         int yourRoll1 = rollDice(6);
         int yourRoll2 = rollDice(6);
         int yourTotal = yourRoll1 + yourRoll2;
         System.out.println("You rolled a " + yourRoll1 + " and a " + yourRoll2);

         int myRoll1 = rollDice(6);
         int myRoll2 = rollDice(6);
         int myTotal = myRoll1 + myRoll2;
         System.out.println("I rolled a " + myRoll1 + " and a " + myRoll2);

         if (yourTotal > myTotal)
            System.out.println("You won!");
         else
            System.out.println("I won.");

         System.out.print("\nDo you want to play again (y or n)? ");
         choice = in.next();
      } while (choice.toUpperCase().equals("Y"));

      in.close();
   }

   public static int rollDice(int numSides) {
      int roll = (int) (Math.random() * numSides) + 1;
      return roll;
   }
}
```

CSE110 – Principles of Programming

Eric Eckert

eric.eckert@asu.edu