Java Libraries and Methods
Chapter Outline

- Using Library Methods
  - Java.Math example
- Writing your own Methods
  - Void Methods
  - Methods with Return Value
- Writing Comments
Using Library Methods
Java Libraries

• The Java Development Kit comes with many libraries which contain classes and methods for you to use
• These are classes and methods that other people have written to solve common tasks
• Some of these include:
  • a Math library (java.lang.Math)
  • String library (java.lang.String)
  • Graphics library (java.awt.* and javax.swing.*)
  • Networking library (java.net)
The Math Library

- Math class
  - Provides many methods that you can use “off the shelf”
    - No need to know how they are implemented
  - Contains constants such as E and PI
- To access:
  - Math.x or Math.m(), where x stands for the name of the constant / m for the name of the method you want to use
Math Library Examples

double positiveNumber = Math.abs(-10);
double twoCubed = Math.pow(2, 3);
double someTrigThingy = Math.sin(Math.PI);

To discover the details of a method (i.e. how to call it), look at the online documentation!

http://download.oracle.com/javase/6/docs/api/java/lang/Math.html

For each method it lists what the method needs as input and what it returns as output.
How to Call a Static Method

Name of Class . Name of Method ( Parameter )

(Values for each Parameter, Separated by Commas)

Parentheses are mandatory, even if there are no Parameters!

• Look at the method headers in the Javadoc to determine the number of parameters and their types!
• http://download.oracle.com/javase/6/docs/api/

COMP-202 - Java Library and Methods, © 2013 Jörg Kienzle and others
Example: Math

- **Class:** `java.lang.Math`
- **Provides many useful mathematical functions**
  - `static double sin(double a)`
  - `static double sqrt(double a)`
  - `static double pow(double a, double b)`
  - `static int abs(int a)`
  - `static int max(int a, int b)`

*Return Type*, i.e. what the Method Produces as Output

*Method Parameters*, i.e. what the Method Requires as Input

{Name of the Method}
Detailed Example: \texttt{sin} (Parameters)

- \texttt{static double sin(double a)}

- The \texttt{sin} method expects as parameter one double expression
- When you call \texttt{sin}, you must put an expression that evaluates to a double between the ( )

- Examples:
  - ... \texttt{Math.sin(3.0)}
  - ... \texttt{Math.sin(2 * Math.PI)}

- Bad Examples:
  - ... \texttt{Math.sin()}
  - ... \texttt{Math.sin(2.3, 5.2)}
  - ... \texttt{Math.sin(“3.0”)}
Detailed Example: \texttt{sin} (Return Type)

- \texttt{static double sin(double a)}

- The \texttt{sin} method produces as \textit{return value} a double
- You must call \texttt{sin} in a place where a double expression is valid
- Examples:
  - \texttt{double angle = Math.sin(3.0);}
  - \texttt{System.out.println(Math.pow(Math.sin(angle),2));}
- Bad Example:
  - \texttt{int angle = Math.sin(3.0);}
Other Libraries

- There are many other libraries that come with the Java SDK
  - See http://download.oracle.com/javase/6/docs/api/
- They are organized in packages
- Anything in the `java.lang` package is available to your program by default
- Anything else needs to be imported
- Example:
  - To use the `Scanner` class in `java.util`, write:
    - `import java.util.Scanner;`
Writing Your Own Methods
Methods

- Often, we can use a library method to solve our problem
- Why reinvent the wheel?
- Sometimes we can’t do this, because:
  - Our need is very specific, and so no one else has bothered to write a library
  - There is a library method that solves our need, but not in a satisfactory way
    - Too slow
    - Uses too much memory
- That’s when you write your own method!
Example: Convert Numbers to Binary

• Convert a decimal number (which is assumed to be smaller than 128) to binary format
import java.util.Scanner;

public class BinaryConversion {
    public static void main(String [] args) {
        Scanner keyboard = new Scanner(System.in);
        System.out.println("Please type a positive number < 128: ");
        byte number = keyboard.nextByte();

        System.out.print(number + " decimal = ");
        // print out one digit after the other
        System.out.print(number/64);
        System.out.print((number%64)/32);
        System.out.print((number%32)/16);
        System.out.print((number%16)/8);
        System.out.print((number%8)/4);
        System.out.print((number%4)/2);
        System.out.print(number%2);
        System.out.println(" binary.");
    }
}
• What if we want to display the number together with the 4 following ones in binary?
• Possible Solution: Copy Paste!

```java
import java.util.Scanner;
public class BinaryConversion {
    public static void main(String[] args) {
        Scanner keyboard = new Scanner(System.in);
        System.out.println("Please type a positive number < 128: ");
        byte number = keyboard.nextByte();
        System.out.print(number + " decimal = ");
        System.out.print(number/64);
        System.out.print((number%64)/32);
        System.out.print((number%32)/16);
        System.out.print((number%16)/8);
        System.out.print((number%8)/4);
        System.out.print((number%4)/2);
        System.out.print(number%2);
        System.out.println(" binary.");

        System.out.print(number + 1 + " decimal = ");
        System.out.print((number+1)/64);
        System.out.print(((number+1)%64)/32);
        System.out.print(((number+1)%32)/16);
        System.out.print(((number+1)%16)/8);
        System.out.print(((number+1)%8)/4);
        System.out.print(((number+1)%4)/2);
        System.out.print((number+1)%2);
        System.out.println(" binary.");

        System.out.print(number + 2 + " decimal = ");
        System.out.print((number+2)/64);
        System.out.print(((number+2)%64)/32);
        System.out.print(((number+2)%32)/16);
        System.out.print(((number+2)%16)/8);
        System.out.print(((number+2)%8)/4);
        System.out.print(((number+2)%4)/2);
        System.out.print((number+2)%2);
        System.out.println(" binary.");

        System.out.print(number + 3 + " decimal = ");
        System.out.print((number+3)/64);
        System.out.print(((number+3)%64)/32);
        System.out.print(((number+3)%32)/16);
        System.out.print(((number+3)%16)/8);
        System.out.print(((number+3)%8)/4);
        System.out.print(((number+3)%4)/2);
        System.out.print((number+3)%2);
        System.out.println(" binary.");

        System.out.print(number + 4 + " decimal = ");
        System.out.print((number+4)/64);
        System.out.print(((number+4)%64)/32);
        System.out.print(((number+4)%32)/16);
        System.out.print(((number+4)%16)/8);
        System.out.print(((number+4)%8)/4);
        System.out.print(((number+4)%4)/2);
        System.out.print((number+4)%2);
        System.out.println(" binary.");
    }
}
```
Problems with Copy / Paste

• Very long repetitive code is not readable
• Error-prone
  • Wrong adjustments after Copy / Paste
• What if we discover that our initial code was wrong?
  • We have to change the code many times

• Better solution: write a method and call it multiple times!
Before Writing a Method

• To write the method, we need to determine:
  • What *name* do we want to give the method?
  • How many *parameters* does it need?
    • (i.e. How many inputs does the method need)
  • What are the types of each parameter?
  • What names should we give the parameters?
  • Should the method be usable in an expression?
    • If yes, determine the *return type* of the method
    • If no, the return type is *void*
Binary Conversion (3)

- **What name** do we want to give the method?
  - `displayInBinary`
- **How many parameters** does it need?
  - 1
- **What are the types of each parameter?**
  - `byte`
- **What names should we give the parameters?**
  - `n`
- **Should the method be usable in an expression?**
  - No ⇒ the return type is `void`
Binary Conversion (4)

```java
public static void displayInBinary(byte n) {
    System.out.print(n + " decimal = ");
    // print out one digit after the other
    System.out.print(n/64);
    System.out.print((n%64)/32);
    System.out.print((n%32)/16);
    System.out.print((n%16)/8);
    System.out.print((n%8)/4);
    System.out.print((n%4)/2);
    System.out.print(n%2);
    System.out.println(" binary.");
}
```
Parameter Passing

- When we call a method, we must provide values of the correct type for each parameter of the method.

- Example: `public static displayInBinary(byte n)`

- The method requires one value of type byte.

- Actual call: `BinaryConversion.displayInBinary(27);`

- When executing the code inside the method, the variable `n` takes the value 27.
• Call the Method as often as desired, passing a different value as a parameter

```java
import java.util.Scanner;
public class BinaryConversion {
    public static void displayInBinary(byte n) {
        // see previous slide
    }
    public static void main(String[] args) {
        Scanner keyboard = new Scanner(System.in);
        System.out.println("Please type a positive number < 128: ");
        byte number = keyboard.nextByte();
        displayInBinary(number);       // or BinaryConversion.displayInBinary
        displayInBinary(number + 1);
        displayInBinary(number + 2);
        displayInBinary(number + 3);
        displayInBinary(number + 4);
    }
}
```
import java.util.Scanner;
public class BinaryConversion {
  public static void displayInBinary(byte n) {
    System.out.print(n + " decimal = ");
    System.out.print(n/64);
    System.out.print((n%64)/32);
    System.out.print((n%32)/16);
    System.out.print((n%16)/8);
    System.out.print((n%8)/4);
    System.out.print((n%4)/2);
    System.out.print(n%2);
    System.out.println(" binary.");
  }
  public static void main(String [] args) {
    Scanner keyboard = new Scanner(System.in);
    System.out.println("Please type a positive number < 128:");
    byte number = keyboard.nextByte();
    displayInBinary(number);
    displayInBinary(number + 1);
    displayInBinary(number + 2);
    displayInBinary(number + 3);
    displayInBinary(number + 4);
  }
}
Void Methods

- When a method returns `void`, you can't use it as part of an expression.
- The purpose of the method is to have a side-effect, not to perform a direct computation.
- One possible side-effect is to display something on the screen.
- `System.out.println()` is a void method.
Methods with Return Value

• If you want to use a result from a computation in your method in an expression, you have to do the following:
  • Change the method header to specify the type of data you want your method to produce
  • Add at least one return statement in the method definition
    • The expression after in the return statement must be of the same type as what is specified in the method header

```java
public static double hypotenuse(double a, double b) {
    double h = Math.sqrt(Math.pow(a,2.0) + Math.pow(b,2.0));
    return h;
}
```

• Now we can call the method

```java
double x = 3.0;
double y = 4.0;
double z = hypotenuse(x,y);
```
Example: Method that tests if a Number is Even

• **Method Header**
  ```java
  boolean isEven(int n)
  ```

• **Implementation**
  ```java
  boolean isEven(int n) {
      return (n%2 == 0);
  }
  ```
Pro’s and Con’s of Methods

• Advantages of Methods
  • Code reusability
  • Reduces code duplication
  • Easier debugging
  • Problems are decomposed
  • Hides tricky logic
  • Easier to read and understand

• Disadvantages of Methods
  • It takes initially a little more time to set them up
Method Exercises

- Write a method called “sayGreeting” that displays a greeting message on the screen. The method should take as input two Strings. One String should be the name of the speaker, the other String should be the name of the listener.

- Write a method called “computeAreaCircle”. The method should take as input the radius of the circle and return a double representing the area of the circle.

- Write a main program that asks the user to enter 3 numbers (one at a time) and for each of these output the area of a circle with that radius.
Writing Comments
Syntax for Comments

• Single line comments
  • Write // anywhere in the code, and the rest of the line is ignored by the compiler

• Multi-line comments
  • Write /* anywhere in the code, and everything that follows is ignored by the compiler until it sees */
Purpose of Comments

• Comments are generally used to help a programmer understand what the code is doing.
• This is useful both for when other people read your code or if you go back to your code at a later point.
• A good comment will make it clear what a complicated piece of code will do.
• A bad comment will either mislead the user or provide unnecessary information (i.e. over commenting)
• Generally, it's better to err on the side of too many comments.
• Every method should be preceded by a brief comment saying what it's purpose is.
// This method takes as input a double representing a
// radius of a circle. It calculates the area of a circle
// using the equation PI*r^2.
// The method returns a double representing the area.
public static double computeAreaCircle(double radius) {
    return radius * radius * Math.PI;
}
Bad Comment Example (1)

// This method takes as input a thing that represents the thing that measures how long it takes to go from the center of a round circle to the outer edge of it. I learned in elementary school that the equation for this is to take the number PI and multiply it by that distance and then multiply it by that distance again. The number PI does not really have anything to do with apple pie, although I kind of wish it did because it's really delicious. However, one thing the two have in common is they both are round.

```java
public static double computeAreaCircle(double radius) {
    return radius * radius * Math.PI;
}
```
/ Good Comment Example (2) 

// This method converts the number n, which // must be strictly smaller than 128, to binary format // and displays the result on the screen 
static void displayInBinary(byte n) { 
    // TODO: make sure method also works with negative numbers 
    System.out.print(n + " decimal = "); 
    // print out one digit after the other 
    System.out.print(n/64); 
    System.out.print((n%64)/32); 
    System.out.print((n%32)/16); 
    System.out.print((n%16)/8); 
    System.out.print((n%8)/4); 
    System.out.print((n%4)/2); 
    System.out.print(n%2); 
    System.out.println(" binary."); 
}
Bad Comment Examples (2)

// sometimes I believe the compiler ignores all my comments

/*
* You may think you know what the following code does.
* But you don't. Trust me.
* Fiddle with it, and you'll spend many a sleepless
* night cursing the moment you thought you'd be clever
* enough to "optimize" the code below.
* Now close this file and go play with something else.
*/

// drunk, fix later