Lecture Set 2: Starting Java

This set has Java Concepts and Java Programming Basics
This Course: Intro to Procedural Programming using Java

Why Java?

- Popular modern language
- Used in web, business, telecom applications
- Developed in 1990s, incorporates many features from earlier languages
  - Object-orientation
  - Garbage collection
  - Portability of object code
Portability of Object Code?

- Object code is 2GL (assembly) / 1GL (machine code)
- Last time we said that 2GL / 1GL is architecture-specific
- How can Java have portable object code?
  Answer: *Java Virtual Machine (JVM)*
Java Virtual Machine

- Java includes definition of *Java bytecode* = “fake” machine code for Java
- Java compilers produce Java bytecode
- To run Java bytecode, must have bytecode interpreter (“Java Virtual Machine”) on client machine
Facts about JVMs

- For efficiency, JVMs often compile bytecode into native machine code
- There are also “native” Java compilers (these compile Java directly to machine code)
example1a

/* This is a very basic Java program to get things started. 
* Notice the difference between println and print 
* The things inside the quotation marks are called "String literals" */

public class Example1a {

    public static void main (String args[]) {  //where the program starts
        System.out.println("Hello World!");
        System.out.print("Or maybe I should say: ");
        System.out.println("Goodbye World!");
    }

}
Method Headers

- **main** is a method = “operation”
  - Operations require operands = data to work on
  - Operations return new data (result)
  - Header gives information on form of operands, result for methods
    For main:
    - Operand is collection of Strings
    - Result is “void” (= unimportant)
    - More later on “public”, “static”
- Every program must have exactly one “main” method (where execution begins)
Comments?

- Comments: explanations added by programmer
  - Two styles
    - /* ... */
    - // to end of line...
  - Comments are essential for good programming!
Objects

- Bundles of data ("instance variables") and methods ("functions")
- Created using classes as "templates"
- We’ll learn more later this semester
Java Program Organization

- Class
  - Structure around which all Java programs are based
  - A typical Java program consists of many classes
  - Each class resides in its own file, whose name is based on the class’s name
  - The class is delimited by curly braces { … }.

File name: Example1.java:

```java
public class Example1a {
    ...
    \(\text{(contents of the class go here)}\) ...
}
```

A class consist of data (variables) and operations (methods)
/* Have you ever noticed that your dryer eats your socks?  
* This example illustrates variables and the assignment  
* operator (=)    
* Note that each variable is "declared" just ONCE!   */

public class Example1b {

    public static void main(String args[]) {
        int numberOfSocks;
        numberOfSocks = 27;    // An odd number of socks???  Crazy.
        System.out.print("The number of socks at the beginning is: ");
        System.out.println(numberOfSocks);
        System.out.println("OK, I'm going to put them in the dryer...");
        int socksLostInDryer = numberOfSocks / 3;    // I lose about a third each time
        numberOfSocks = numberOfSocks - socksLostInDryer;
        System.out.print("The number of socks lost was: ");
        System.out.println(socksLostInDryer);
        System.out.print("The number of socks is now: ");
        System.out.println(numberOfSocks);
    }
}

Java Program Organization

- **Methods**
  - Where most computation takes place
  - Each method has a name, a list of arguments enclosed in (…), and body (collection of statements) in {…}

```java
public static void main( String[] args ) {
    ... (contents of the main method go here) ...
}
```

- **Variables**
  - Storage locations that program can operate on
  - Variables can store data of different forms (integers, for example)

```java
int secondsPerMinute = 60;
int minutesPerLecture = 50;
```
Java Program Organization

- Statements: Many different types
  - Declarations – specify variable types (and optionally initialize)
    ```java
    int x, y, z; // three integer variables
    String s = "Howdy"; // a character string variable
    boolean isValid = true; // a boolean (true/false) variable
    ```
  - Assignments – assign variables new values
    ```java
    x = 13;
    ```
  - Method invocation – call other methods
    ```java
    System.out.println("Print this message");
    ```
  - Control flow – determine the order of statement execution.
    (These include if-then-else, while, do-while, for. More later.)
- Built-in Operators: For manipulating values (+, -, *, /, etc.)
/* This example illustrates "concatenation" of strings, and
* shows how Java automatically converts values into strings */

public class Example1c {

    public static void main(String[] args) {
        System.out.println("My " + "name " + "is " + "Fred.");
        int secondsPerMinute = 60;
        int minutesPerLecture = 50;
        int secondsPerLecture = secondsPerMinute * minutesPerLecture;
        System.out.println("There are " + secondsPerLecture + " seconds in a lecture.");
    }
}

# Built-in (Primitive) Types

<table>
<thead>
<tr>
<th></th>
<th>Type name</th>
<th>Size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Integers</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>byte</code></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td><code>short</code></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td><code>int</code></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td><code>long</code></td>
<td>8</td>
</tr>
<tr>
<td><strong>Reals</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>float</code></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td><code>double</code></td>
<td>8</td>
</tr>
<tr>
<td><strong>Other</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>char</code></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td><code>boolean</code></td>
<td>1</td>
</tr>
</tbody>
</table>
String Type

- Elements of String type are sequences of characters
  “abc” “Call me Ishmael” etc.
- String type is not built-in
- We will use it a lot
- Useful operation: concatenation (+)
  “abc” + “def” = “abcdef”
Example 2: Basic Types

/* Demonstration of "primitive types"
 * and also the String type.
 *
 * Note that you can declare many different variables with one statement! */

public class Example2 {
    public static void main(String[] args) {
        int i1, i2, i3;
        double f1 = 7.3, f2 = 9.4;
        boolean b1, b2;
        char c;
        String s;
        i1 = 7;
        i2 = 3;
        i3 = i1 + i2 * 5 - 2;
        f1 = 3.1415927;
        b1 = true;
        b2 = (f2 < f1);
        c = 'X';
        s = "Hello " + "there" + " my friend.";
        System.out.println("i3 = " + i3);
        System.out.println("f1 = " + f1);
        System.out.println("b1 = " + b1);
        System.out.println("b2 = " + b2);
        System.out.println("c = " + c);
        System.out.println("s = " + s);
    }
}
Programming Errors

- Types of Errors
  - Syntax Errors
    - violates languages grammar
    - compiler warns about these
    - Eclipse puts red squiggles under the offending code
  - Semantic/Logic Errors
    - program doesn’t work properly
    - run-time errors = crash or hang
    - can be more subtle (harder to find)

- Debugging
  - process of finding and fixing problems
  - to minimize debugging frustration – use “unit” testing
    - write a small part, thoroughly test it, cycle back