Lecture 3: Starting Java

Last time:
1. Programming languages
2. Eclipse and CVS

Today:
1. CVS and project submission
2. Basics of Java Programs
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

```
source code

.java

→

Java compiler

→

.object code

.class

→

JVM

client

→

JVM

client
```
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)
/* This is a very basic Java program to get things started. */

public class Example1 {

    // main is where the program starts
    public static void main(String args[]) {
        int secondsPerMinute = 60;
        int minutesPerLecture = 50;
        int totalSeconds = secondsPerMinute * minutesPerLecture;
        System.out.println("There are "+ totalSeconds
                 + " seconds in a lecture.");
    }
}

Example1.java

A Sample Java Program
Comments? Class?

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

- Classes
  - Classes are key components of Java programs
  - Class: a collection of data and associated operations
    - data: “instance variables”
    - operations: “methods”
      - One method in this program: “main”
/* This is a very basic Java program to get things started. */

public class Example1 {

    // main is where the program starts
    public static void main(String args[]) {
        int secondsPerMinute = 60;
        int minutesPerLecture = 50;
        int totalSeconds = secondsPerMinute * minutesPerLecture;
        System.out.println("There are " + totalSeconds + " seconds in a lecture.");
    }
}

Java Program (cont.)
**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)
Statements?

- Basic commands in a program
- Two kinds of statements in this program
  - Variable declarations
    ```java
    int secondsPerMinute = 60;
    ```
    - Introduce a named storage location (“secondsPerMinute”)
    - Define form of data that can be stored (“int” = integer)
    - (Optional) Give initial value (“60”)
  - Method invocation
    ```java
    System.out.println("There are" ...);
    ```
    - Calls to methods in objects = “class instances”
    - Method is “println” in subobject “out” within object “System”
Objects?

• Bundles of data ("instance variables") and methods
• Created using classes as "templates"
• We’ll learn more later this semester
Executing the Program

```java
/* This is a very basic Java program to get things started. */

public class Example1 {

    // main is where the program starts
    public static void main(String[] args) {

        int secondsPerMinute = 60;
        int minutesPerLecture = 50;
        int totalSeconds = secondsPerMinute * minutesPerLecture;

        System.out.println("There are "+ totalSeconds + " seconds in a lecture.");
    }
}
```

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>secondsPerMinute</td>
<td>60</td>
</tr>
<tr>
<td>minutesPerLecture</td>
<td>50</td>
</tr>
<tr>
<td>totalSeconds</td>
<td>3000</td>
</tr>
</tbody>
</table>

There are 3000 seconds in a lecture.
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 Example1 {
    ... (contents of the class go here) ...
}
```

A class consist of data (**variables**) and operations (**methods**)
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.)
Types

- Two references to the “form” of values
  - Method headers
  - Variables
- *Type*: the form of a data value in Java
## Built-in (Primitive) Types

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

public class Example2 {

    public static void main(String[] args) {
        int i1, i2, i3;
        double f1, f2;
        boolean b1, b2;
        char c;
        String s;

        i1 = 17;
        i2 = 23;
        i3 = i1 + i2 * 15 - 24;
        f1 = 3.1415927;
        f2 = f1 / 337.2734;
        b1 = false;
        b2 = (f2 < f1);
        c = 'X';
        s = "Hello " + "there" + " my friend.";
        System.out.println("i3 = " + i3);
        System.out.println("f2 = " + f2);
        System.out.println("b1 = " + b1);
        System.out.println("b2 = " + b2);
        System.out.println("c = " + c);
        System.out.println("s = " + s);
    }
}

CMSC 131 Spring 2007
Bonnie Dorr (adapted from Rance Cleaveland)