ISE 582: Information Technology for Industrial Engineers
University of Southern California
Department of Industrial and Systems Engineering

Lecture 2-2
First cup of JAVA

Handouts
• Lecture 2-1: HTML Forms
• Lecture 2-2: First Cup of Java
• Homework 1
• READ:
  – Winston & Narasimhan: Chapt 1-5 (pp 1-33)

Announcements
• Homework 1 is due a week from today
• Next week, Luca will be teaching class
The Agenda for Today

- HTML Forms
- Introduction to Objects
- Programming in General
- Getting Started in JAVA

Forms Help

- At USC,  
  - http://www.usc.edu/uscweb/authoring/scripting.html/
- Public domain scripts, e.g.:  
  - http://www.jsworld.com/scripts/forms/  
- Example: Feedback Form, Opinion Poll

Introduction to Objects

- Object Instance  
  - Attributes (properties): associated data  
  - Operations: actions that change attributes  
    - Methods in JAVA  
    - Member functions in C++
- Object Class  
  - Category or set of objects  
  - Share same attributes, operations and relationships to other objects
Example: Object Class

<table>
<thead>
<tr>
<th>Class</th>
<th>Attributes</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>WIZARD</td>
<td>name</td>
<td>changeName(...)</td>
</tr>
<tr>
<td></td>
<td>age</td>
<td>incrementAge()</td>
</tr>
<tr>
<td></td>
<td>house</td>
<td>calculateDOB()</td>
</tr>
<tr>
<td></td>
<td>claimToFame</td>
<td></td>
</tr>
</tbody>
</table>

Example: Object Instances

- **w1: Wizard**
  - name = “Albus Dumbledore”
  - age = 161
  - house = “Gryffindor”
  - claimToFame = “Defeated dark wizard Grindelwald”

- **w2: Wizard**
  - name = “Harry Potter”
  - age = 21
  - house = “Gryffindor”
  - claimToFame = “Survived deadly curse by Voldemort”

Objects and older constructs

- Operation/ Method = function/procedure
- Object Instance = FAT variable containing multiple pieces of data and its own functions
Naming Conventions

- Convention:
  - Attributes/operation names begin with lower-case characters
  - Class and type names are capitalized
- Common Operations:
  - get*, set*, is*

Class name
attributes:Type
operations

Programming in General

- Algorithms
- Pseudocode
- Reusable Components
- Testing and Debugging

Algorithms and Pseudocode

- Hardest part in designing methods: coming up with a plan or strategy for carrying out the action
- Algorithm: set of instructions for solving a problem. So precise that even a dumb computer could follow the instructions and get the same results
- Pseudocode: an algorithm written in a mixture of English and some programming language
Reusable Components
- Most programs combine already existing components
- Be sure to design classes so that they are reusable

Testing and Debugging
- Syntax Error: a grammatical mistake
- Run-time Error: detected when program is run
- Logic Error: incorrect but legal

Development Cycle
- Write or edit source code
- Compile source code
- Execute program
- Fix bugs that emerge during compilation
- Fix bugs that emerge during execution
Getting Started in JAVA

• A Little JAVA History
• Compile and Execute a Simple Program
• Declare Variables
• Arithmetic Expressions, the Math Class
• Define a Simple Method

JAVA: Definition

• 1. An island of Indonesia separated from Borneo by the Java Sea, an arm of the western Pacific Ocean. Center of an early Hindu Javanese civilization, Java was converted to Islam before the arrival of the Europeans (mainly the Dutch) in the late 16th century.

JAVA: Definition

• 2. A trademark used for a programming language designed to develop applications, especially ones for the Internet, that can operate on different platforms.
• 3. n. Informal: Brewed coffee
A Little JAVA History

- 1991: Programming Language for Home Appliances (Gosling)
- 1994: Coupled with HotJava web-browser (Naughton and Payne)
- 1995: Netscape made browsers capable of running Java programs

Source to Byte Code

- The Java compiler translates the class definitions and programs into byte code
- Byte code is executed by an interpreter, called a Java Virtual Machine

Example: Compute Movie Rating

- Suppose script(6), acting(9), directing(8)
- To arrange for these scores to be totaled, define main inside class definition:

```java
public static void main (String argv[]) {
    System.out.print("The rating of the movie is ");
    System.out.println(6 + 9 + 8);
}
```
Example continued

- Embed method in a class definition
  ```java
  public class Demonstrate {
      ...
  }
  ```
- Explicit values are said to be *literal*
- Java is *blank insensitive, but case sensitive*

Example: Compile and Execute

- At the command prompt:
  - Type "javac Demonstrate.java"
  - (this generates a file Demonstrate.class)
  - Type "java Demonstrate"
- What will the output be?
- Data in this program is said to be
  - Wired in or hard coded

Declaring Variables in JAVA

- Identifiers are names of variables
  - E.g. age, claimToFame, many$$$, no_way
- Variables are reserved chunks of memory that contain values.
- Type of variable determines how much memory is allocated to it.
Some Variable Types

- char: 2 bytes: characters
- byte, short, int, long: increasing: integers
- float, double: 4, 8: floating point number
- OTHERS: boolean, String, arrays etc

```java
public class Demonstrate{
    public static void main (String argv[])
        int script, acting, direction
    }
}
```

COMMENTING YOUR CODE

- // This is a one-line comment
- /* This comment
   just goes on
   and on and on… */
- NOTE: no nesting of comments.

ARITHMETIC EXPRESSIONS

- The usual operators: +, -, *, /
- Modulus: % (remainder in division)
- The usual precedence rules apply
- Mixing data types in expressions
- Casting an expression
  – Original data type remains unchanged
- Assignment operates from R to L
### The Math Class

- `public class Demonstrate {
  
  public static void main(String argv[]) {
    System.out.println("natural log of 10:" + Math.log(10));
    System.out.println("abs value of -10:" + Math.abs(-10));
    System.out.println("max of 2 and 3:" + Math.max(2, 3));
    System.out.println("5th power of 6:" + Math.pow(6, 5));
    System.out.println("sqrt of 7:" + Math.sqrt(7));
    System.out.println("sine of 8 rad:" + Math.sin(8));
    System.out.println("Random no.(0,1):" + Math.random());
  }
}

### Defining a Simple Method

```java
public class Demonstrate {
  public static void main(String argv[]) {
    int script = 6, acting = 9, direction = 8;
    System.out.print("The rating of the movie is ");
    System.out.println(movieRating(script, acting, direction));
  }

  public static int movieRating(int s, int a, int d) {
    return s + a + d;
  }
}
```

### Defining Methods in Multiple Files

```java
public class Movie {
  public static int movieRating(int s, int a, int d) {
    return s + a + d;
  }
}

public class Demonstrate {
  public static void main(String argv[]) {
    int x = 6, a = 9, d = 8;
    System.out.print("The rating of the movie is ");
    System.out.println(Movie.movieRating(x, a, d));
  }
}
```
Other Things to Know

• You can define more than one class in the same file
• To call a method in a different class, preface the method name with the class name and join with a dot.
• You can define multiple methods with the same name in the same class as long as they have different signatures (i.e. arrangement of parameter data types)