Lecture 2: Introducing Java

CS 170, Section 000, Fall 2009
1 September 2009

Lecture Plan

• Recap from Lecture 1
• Exercise
• Introduction to UNIX
• Introduction to Java
  – Writing and running a first program
Recap: Key Concepts in CS 170

- **Elementary computer architecture** - what is a computer, how does it work.
- **UNIX system** - how to use the UNIX system, commands to list and navigate directory hierarchy, edit and save files, etc.
- **Programming concepts**, such as parameter passing mechanisms, objects, data abstraction, inheritance, etc.
- **Programming methodologies**, such as modular programming and object oriented programming.
- **Intro to data structures** - computer programs manage information, how do you manage information? Arrays
- Some **object-oriented programming concepts**, such as inheritance and interfaces

Key Tools

- **Java** programming language
- **UNIX** operating system
- **Eclipse** development environment
- **If time:** **Javascript and HTML** (Elementary web programming)
Key fact of the day: The Class Webpage

- http://www.mathcs.emory.edu/~cs170000
  - Swine Flu update 😊

Recap: Modern Computer Architecture

- Computer components: CPU, memory, hard disk, floppy disk, monitor, printer, and communication devices.
Program Languages

- Instructions are specified using a computer programming language
- Three types of programming languages:
  - Machine language
  - Assembly language
  - High-level language

Assembly Language

- Developed to make programming easy
- The computer cannot understand assembly language
  - Need to convert assembly code to machine code

```
ADDF3 R1, R2, R3
```
Program Execution Exercise

- Program: add 3 numbers together and print result: \( D = A + B + C \);
  - Input: \( A \rightarrow \text{Memory} \); \( B \rightarrow \text{Memory} \); \( C \rightarrow \text{Memory} \);
  - Add \( A, B \rightarrow \text{TEMP} \) //fetch \( A, B \) values from Memory
  - ADD \( C, \text{TEMP} \rightarrow D \) //fetch \( C, \text{TEMP} \) from Memory
  - Print \( D \);

---

High-Level Language

- English-like and easy to learn and program
- Can be understood by just looking at the code

\[
\text{area} = 5 \times 5 \times 3.1415;
\]

- Required steps to produce byte code?
Compiling Source Code

- Source code
  - A program written in a high-level language
- Compiler
  - Used to translate the source program into a machine language program called an object program
- Linker
  - Used to link your object program with supporting libraries

---

History of Java

- James Gosling and Sun Microsystems
- Java presented on May 20, 1995 during Sun World
- HotJava
  - The first Java-enabled Web browser
- Why the name Java?

Silk   Lyric   Pepper   NetProse
Neon   Java    DNA      WebDancer
WebSpinner    WRL (WebRunner Languages)
Why Java?

- Applications for:
  - Internet services (Java Applets, Java Web Applications)
  - Desktop computers
  - Hand-held devices
- Well-established and popular language
  - Langpop.com

Characteristics of Java

- Java Is Object-Oriented: modularity/flexibility
- Java Is Distributed: several computers working together
- Java Is Robust: Compiler can catch many errors
- Java Is Architecture-Neutral
- Java Is Portable
Java Architecture

• Java is Portable and Architecture Neutral

![Java Architecture Diagram]

Java Standards

• Language Specification:
  – Syntax and semantics (grammar and rhetoric)

• API (Application Programming Interface)
  – Predefined classes for developing programs
Creating, Compiling and Running Programs

Programming in Unix

- Unix operating system
  - Command line interface (CLI) through shell window
  - GNOME desktop environment
- Unix editors
  - vi or Vim
  - gedit
- Java compiler
  - javac included in JDK
- Java Virtual Machine (JVM)
  - java
Files and Folders in Unix

- Java source code are saved as .java files
  - Welcome.java
- Files: a collection of items of information that are kept together
- Files have names; legal names:
  - letters (A-z), numbers (0-9), “.”, “-”, and “_”.
  - Welcome1.java, 3p0.x, cs170-example.1
- Files are stored in folders or directories; these file containers can be nested

Integrated Development Environment (IDE)

- IDE is a software application that provides comprehensive facilities for software development
  - Source code editor
  - Compiler and/or interpreter
  - Debugger
- Popular IDEs
  - Eclipse
  - NetBeans Open Source by Sun
  - Borland JBuilder
Debugging

- Etymology:
  - De (remove) bugs (?)

- A software bug is used to describe an error, flaw, mistake or fault in a computer program or system that produces an incorrect or unexpected result.

First Java Program

// This program prints Welcome to Java!
public class Welcome {
    public static void main (String[] args) {
        System.out.println ("Welcome to Java!");
    }
}

Demonstration

• Create a **source** program with editor
  – `gedit Welcome.java`

• Save the program *(which component?)*

• **Compile** the program with a compiler
  – `javac Welcome.java`

• **Execute** the program with java JVM
  – `java Welcome`

Creating and Editing Programs with **gedit**

To use `gedit`, type

```
  gedit Welcome.java
```

from the shell
Trace Program Execution

```java
// This program prints Welcome to Java!
public class Welcome {
    public static void main(String[] args) {
        System.out.println("Welcome to Java!");
    }
}
```

Enter main method

Trace Program Execution (cont’d)

```java
// This program prints Welcome to Java!
public class Welcome {
    public static void main(String[] args) {
        System.out.println("Welcome to Java!");
    }
}
```

Execute statement
Trace Program Execution (cont’d)

```java
// This program prints Welcome to Java!
public class Welcome {
    public static void main(String[] args) {
        System.out.println("Welcome to Java!");
    }
}
```

print a message to the console

Structure of a Java Program

- White spaces
- Comments
- Reserved words
- Modifiers
- Statements
- Blocks
- Classes
- Methods
- The main method
Comments (ignored text)

- Programming is as much about **communicating with people** as with the computer

- Three types of comments:
  - Line comment
    ```
    // This is an example of line comment
    ```
  - Paragraph comment
    ```
    /* This is an example of paragraph comment.
    It can have few lines...*/
    ```
  - Javadoc comment
    ```
    /**
    This is a Javadoc comment.
    */
    ```

Reserved Words (a.k.a. keywords)

- Words that have a specific meaning to the compiler
- Cannot be used for other purposes in the program

```java
import java.io.*;
/**
 * This class prints string Welcome to Java!
 */
public class Welcome {
    public static void main(String[] args) {
        //The statement below prints the string
        System.out.println("Welcome to Java!");
    }
}
```
Modifiers

• Certain reserved words are called modifiers
• They specify the properties of the data, methods, and classes
  – Public, static, private, final, abstract, and protected

```java
import java.io.*;

/**
 * This class prints string Welcome to Java!
 */
public class Welcome {
    public static void main(String[] args) {
        // The statement below prints the string
        System.out.println("Welcome to Java!");
    }
}
```

Statements

• A statement represents an action or a sequence of actions;

```java
import java.io.*;

/*
 * This class prints string Welcome to Java!
 */
public class Welcome {
    public static void main(String[] args) {
        // The statement below prints the string
        System.out.println("Welcome to Java!");
    }
}
```

• Another example: `Area = r*r*3.14;`
Blocks

- A pair of braces `{ ... }` in a program forms a block that groups components of a program

```java
public class Test {
    public static void main(String[] args) {
        System.out.println("Welcome to Java!");
    }
}
```

Classes

- The `class` is the essential Java construct
- A class is a template or blueprint for objects
- We will learn a lot about classes later in the course

```java
import java.io.*;

/**
 * This class prints string Welcome to Java!
 */
public class Welcome {
    public static void main(String[] args) {
        //The statement below prints the string
        System.out.println("Welcome to Java!");
    }
}
```
Methods

- **Method** is a collection of statements
  - perform a sequences of operations
  - Used by invoking a statement with a string argument
- Method **arguments** are enclosed within parentheses and separated by commas

```java
public static void main(String[] args) {
    // The statement below prints the string
    System.out.println("Welcome to Java!");
}
```

The **main** Method

- The main method provides the control of program flow
- The Java interpreter executes the application by invoking the main method

```java
public class Welcome {
    public static void main(String[] args) {
        // The statement below prints the string
        System.out.println("Welcome to Java!");
    }
}
```
Another Example

```java
public class Welcome1 {
    public static void main(String[] args) {
        System.out.println("Programming is fun!");
        System.out.println("Fundamentals first");
        System.out.println("Problem driven");
    }
}
```

Class Exercise

- Let’s write a Java program that displays a smiley
  – a.k.a. ASCII Art
Displaying Text in Dialog Box

• Java can be used to rapid application development
• For instance, you can create message dialog boxes very easily
• Simply use the `showMessageDialog` method in the `JOptionPane` class
  – `JOptionPane` is one of the many predefined classes in the Java API

How to make use of `JOptionPane`?

```java
import javax.swing.JOptionPane;

JOptionPane.showMessageDialog(null,
    "Welcome to Java!",
    "Display Message",
    JOptionPane.INFORMATION_MESSAGE);
```
Summary

- Computer Architecture (Recap, exercise)
- Programs and programming languages
- Java language
- First Java program

ToDo

- Chapter 1 (skim Section 1.5)
- Lab today at 4pm!