Syllabus and Programming Overview

CSCI 161 – Introduction to Programming I

Professor Thomas Rogers

Overview

- Related reading: Chapter 1
- Topics:
  - Class Essentials
  - What is Programming?
  - Why is Programming Important?
  - Introduction to Java
  - Programming in the Lab and on Your Own Computer

Class Essentials

- Introduction to instructor, student roll call taken.
- Review of instructor's page at cs.millersville.edu/~trogers
- Review of course page at cs.millersville.edu/~trogers/csci161
- Review of class, lab, exam schedule here

What is Programming?

- To understand what a program is you must first understand hardware and software.
- **Hardware** is what the software runs on. Your iPhone is hardware. Hardware in this context is simply a computer.
- A computer is a machine that manipulates data and executes list of instructions known as a program.
- For a machine to be a computer it must be "Turing Complete". Turing completeness is the ability for a system of instructions to simulate a Turing machine. A programming language that is Turing complete is theoretically capable of expressing all tasks accomplishable by computers; nearly all programming languages are Turing complete if the limitations of finite memory are ignored.

What is Programming? (continued)

- **Software** is programming running on a computer - the programs; a program - list of instructions to be carried out by a computer.
- **Programs** are written in languages, called "Programming Languages."
- The first program written by Ada Lovelace in 1842. Was written on punch cards used to control a yet-to-be-built "Difference Engine" (a mechanical computer) by Charles Babbage who is coined "The Father of Computers."

Why is Programming Important?

- Computer Science is a vast discipline with many different areas of study. One can be a computer scientist without ever having to program.
- Yet all colleges and universities that offer Computer Science degrees require their students to take various courses in programming and programming languages. Why?
  Answer: Because the discipline of programming requires "algorithmic thinking." *We study programming not because it is the most important aspect of Computer Science, but because it is the best way to explain the approach that computer scientists take to solving problems.*
- **Algorithm** - A step-by-step instruction of how to accomplish a task.

Why is Programming Important?
(continued)

- Do Androids Dream of Electric Sheep? - Do computers dream? They certainly speak and can be spoken to and their native language is binary.
- We humans think of numbers in terms of the decimal system. Computers think in terms of the binary system and binary is the root of all that is logical in computers.
- Computer programs can be written in many languages, some higher level than others. The lowest level language is the "machine language" (often referred to as machine code), it is binary, and it is tied to the physical computer on which it runs.
- **Code** is the fragment of a program.
- A program must be written, it must follow the **syntax** (rules) of the language it is written in and in order to run it must be **compiled** (put into a form to be executed).

Introduction to Java

- Java is a little different than many languages. It doesn't compile into machine language, it compiles into machine-independent *bytecodes*.
- Java requires a **Java Virtual Machine** (JVM) in order to run a compiled Java program.
- To run the bytecodes of a Java program you need a **Java Runtime**
- **Java:** *A simple, object-oriented, network-savvy, interpreted, robust, secure, architect neutral, portable, high-performance, multithreaded, dynamic language.*
- **Java Class Libraries:** The collection of preexisting Java code that provides solutions to common programming problems (lang, io, math, ...)

Introduction to Java (continued)

- Java is a little different than many languages. It doesn't compile into machine language, it compiles into machine-independent *bytecodes*.
- Java requires a **Java Virtual Machine** (JVM) in order to run a compiled Java program.
- To run the bytecodes of a Java program you need a **Java Runtime**
- **Java:** *A simple, object-oriented, network-savvy, interpreted, robust, secure, architect neutral, portable, high-performance, multithreaded, dynamic language.*
- **Java Class Libraries:** The collection of preexisting Java code that provides solutions to common programming problems (lang, io, math, ...)

Introduction to Java (continued)

- Programming in Java:
  - Type in a program as a Java class.
  - Compile the program.
  - Run the compiled version of the program.
- Hello World:

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello world!");
    }
}
```

Introduction to Java (continued)

- Edit, Compile, Run:
  - Edit file and give it same name as class name and extension of .java: **HelloWorld.java**
  - From terminal window compile as:

    ```
    javac HelloWorld.java
    ```

    That should generate **HelloWorld.class**, which is contains the compiled bytecodes, ready to run.
  - To execute: `java HelloWorld`

Introduction to Java (continued)

- Classes, Methods, Statements:
  - **Class** - HelloWorld is a class. Every .java file require a single class of the same class name as file name.
  - **Method** - A program unit (code) that represents a particular action or computation. **main** is a method with a special purpose.
  - **Statement** - An executable snipped of code that represents a complete command. Each statement ends in a semi-colon (;).
  - **Curly braces** - Multiple statements executed as a unit because they are contained within a method, or multiple methods that are contained within a class are enclosed between matching curly braces.

Introduction to Java (continued)

- So, to review, every Java program:
  - Is stored in a class with the same class name as the file name,
  - Within the class there are methods. At a minimum, a complete program requires a special method called **main**,
  - Inside a method like main there is a series of statements, each of which represents a single command for the computer to execute.

Programming in the Lab and on Your Own Computer

- Overview:
  - This class uses the Eclipse Interactive Development Environment (IDE) development tool which is installed in the Linux Lab (STB 130). If you choose, you can do all your programming assignments in the lab

    However, it is highly recommended that you install Eclipse and other required developer tools on your laptop or computer so that you can easily work on the programming assignments outside of the lab.

Programming in the Lab and on Your Own Computer (continued)

- The programming tools you will need are:
  - For Windows:
    - Before installing Eclipse you *may* need to install the latest Java Development Kit (JDK). To install **Oracle Java SE Development Kit 17 Downloads** click here.
    - For developing your programs use the **Eclipse IDE for Java Developers** program. Install the Eclipse IDE 2021-12 version. Click the Windows 64-bit download link to begin the installation.
    - For uploading your programs to the Linux system use **CyberDuck**. Install the latest Windows version by clicking here.
    - For accessing the Linux systems remotely use **PuTTY**. Install latest version Click on *MSI (Windows Installer)* 32-bit or 64-bit installer to begin the installation.

Programming in the Lab and on Your Own Computer (continued)

- The programming tools you will need are:
  - For Mac OS:
    - Before installing Eclipse you *may* need to install the latest Java Development Kit (JDK). To install the JDK go **Oracle Java SE Development Kit 17 Downloads** click here.
    - For developing your programs use the **Eclipse IDE for Java Developers** program. Install the Eclipse IDE 20210-12 version. Click the download link to begin the installation.
    - For uploading your programs to the Linux system use **CyberDuck**. Install the latest Mac version by clicking here. A ZIP file is downloaded to your machine. CyberDuck can be run from there, saved to your Applications folder, etc.
    - For accessing the Linux systems remotely use the *ssh* command from a Mac OS Terminal window.

Programming in the Lab and on Your Own Computer (continued)

- Accessing the Linux lab remotely:
  - To access the Linux system when uploading or downloading files via CyberDuck using **SFTP**, or logging in via PuTTY (Windows) or ssh (Terminal window on Mac) make sure to use your Linux system userid and password.
  - The remote hostname for use in ssh and PuTTY includes your userid and hostname **csciarch-ssh.millersville.edu**, as in: ***rogers161@csciarch-ssh.millersville.edu***
  - For more info on connecting to the remote Linux machines click here.
  - For additional information on remote access see the remote access document.