CS 99: Fundamentals of Programming

Summer 2000

Michael Clarkson

Lecture 1: Programming Basics

Agenda

- Course introduction
- Programming basics
- Java programs

Course Introduction

- Syllabus
- Questionnaire
- To do by tomorrow's lab:
  - Pick up NetIDs (or by Thursday at the latest!)
  - Acquire at least 4 floppy disks, or one ZIP-100 disk
  - Read the assigned sections

Programming Basics

- Writing programs
- Running programs

Writing Programs

- What are programs?
- Algorithms
- Pseudocode
- Functions
- Style

Programs

- Code executed by a computer
- Code: instructions in a programming language (e.g., Java)
- Examples: Microsoft Word, Eudora
- Writing programs is problem solving
  - Primary task: break into simpler problems

Example Java Program

```
class Example {
   public static void main(String[] args) {
       System.out.println("This is a simple example.");
   }
}
```


Algorithms

- Algorithm: "A finite set of rules that gives a sequence of operations for solving a specific type of problem." – Donald Knuth
- Standard simile: like a recipe
  - Has inputs (ingredients), outputs (prepared food)
  - Tells you what to do, what order to do it in
- Programs are composed of algorithms

Pseudocode

- Language-independent method of expressing algorithms
- Great for talking about how to do something without getting caught up in the details of how to program it
- Example:

```
SortCards
    Pick lowest card
    Put in hand
    Repeat
```


Functions

- In almost any language, programs are divided into functions
- Function: code written to perform one (small) well-defined task
- Building blocks of programs
- Libraries of functions exist so that programmers don't have to keep "reinventing the wheel"
- Synonyms: procedure, method
- Java programs are divided first into *classes*, then into methods

Style

- Writing in a programming language requires good style
- Just as writing in a natural language (e.g., English) requires good style
- *Handout*

Running Programs

- Computer architecture (what)
- Compiling/interpreting (how)

Computer Architecture

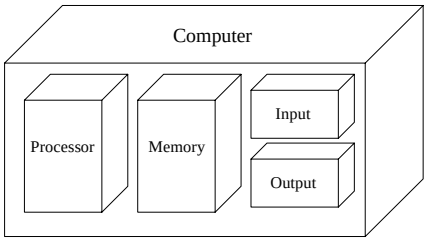
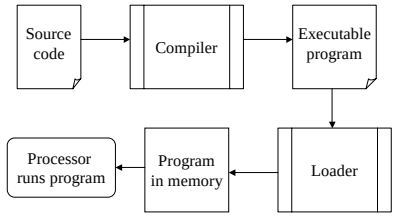


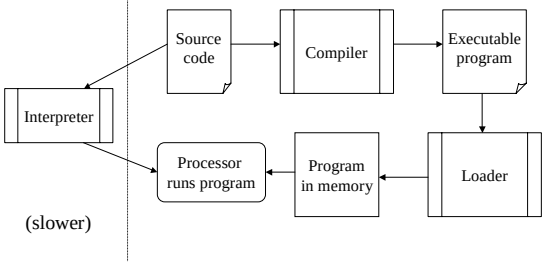
Diagram from *Computer Organization and Design* by Patterson & Hennessy

Compiling



Compiling

Interpreting



(slower)

Java Programs

- Anatomy
- Compiling and executing

Anatomy of a Java Program

In a file called Example.java:

```
class Example {
  public static void main(String[] args) {
      System.out.println("This is a simple example.");
  }
}
```

Things to notice: a class called Example, a method called main, the name of the file

Java Punctuation

- Every class and method has a body that is enclosed in braces: { … }
- Every method is a series of statements, each of which is terminated by a semicolon: …;
- Methods always involve parentheses: (…)

Compiling Java

- A Java source code file always ends with the extension .java (e.g., Example.java)
- The Java compiler translates Java source code into Java bytecode
- Bytecode files always have an extension of .class (e.g., Example.class)

Executing Java

- Java bytecode can be executed in two ways:
  - Interpreted by a Java Virtual Machine (JVM)
  - Compiled by a Just-In Time compiler (JIT)
- Why be so complicated?
  - Bytecode (.class files) is independent of real machines
  - Portability: code can be written on one platform (e.g., Windows) and run on another (e.g., Mac) without any changes

Executing Java [2]

- Execution always starts with the method called *main*
- The main method must look exactly like:

```
public static void main(String[] args) {
      …
}
```


Executing Java [3]