Chapter 1:
The Way of the Program

Think Java:
How to Think Like a Computer Scientist

5.1.2

by Allen B. Downey

Agenda

- Introductions
- Course Administration
- Programming Concepts
- Java “Hello World” Demo
- Debugging strategies

Welcome to CSIS10A (5 mins)

- Typical format for class meetings
  - Short lectures on Tuesday and Thursday (monitors at 90^{o})
  - Lab Part A on Tuesday, part B on Thursday
  - Listen, code, Listen, code
- Syllabus
- Aim of this course (learning to think like a CS)
- What is a program? A set of step-by-step instructions that directs a computer to solve some problem.
- Pretty much anything you can do with a computer, you can do by programming in Java

Do introductions (15 mins)

- class website: mpconline.mpc.edu
- Fill out the CSIS10A Guestbook survey
  - Name
  - Major and what year
  - Why taking the course
  - Something unique about yourself

Do administrative stuff (10 mins)

- Few if any handouts will be printed. They will be posted on the web instead.
- No late work (model solutions, quick turnaround)
- For conflicts with class meetings: let me know now!
- We will use the FREE ONLINE textbook Think Java
- Also recommended: Starting Out with Java by Gaddis (4^{th} or 5^{th} edition OK) for deeper coverage
- How to get help
  - Computer science tutoring is available, contact Tom Rebold, trebold@mpc.edu for more information

Warning – there is MATH in this class!!

- If you didn't like math (algebra), you probably will have a hard time in this class

1. Solve for x: $5x + 3 = 13$
2. What is 8 squared?
3. Square root of 36
4. Area of a circle of radius 2:
5. Average of 10, 14, 8, 4
6. Convert 3 kg into lbs
7. Convert 43 ounces into pounds and ounces
8. What is 15% of $30.00 ?

The Wise Approach

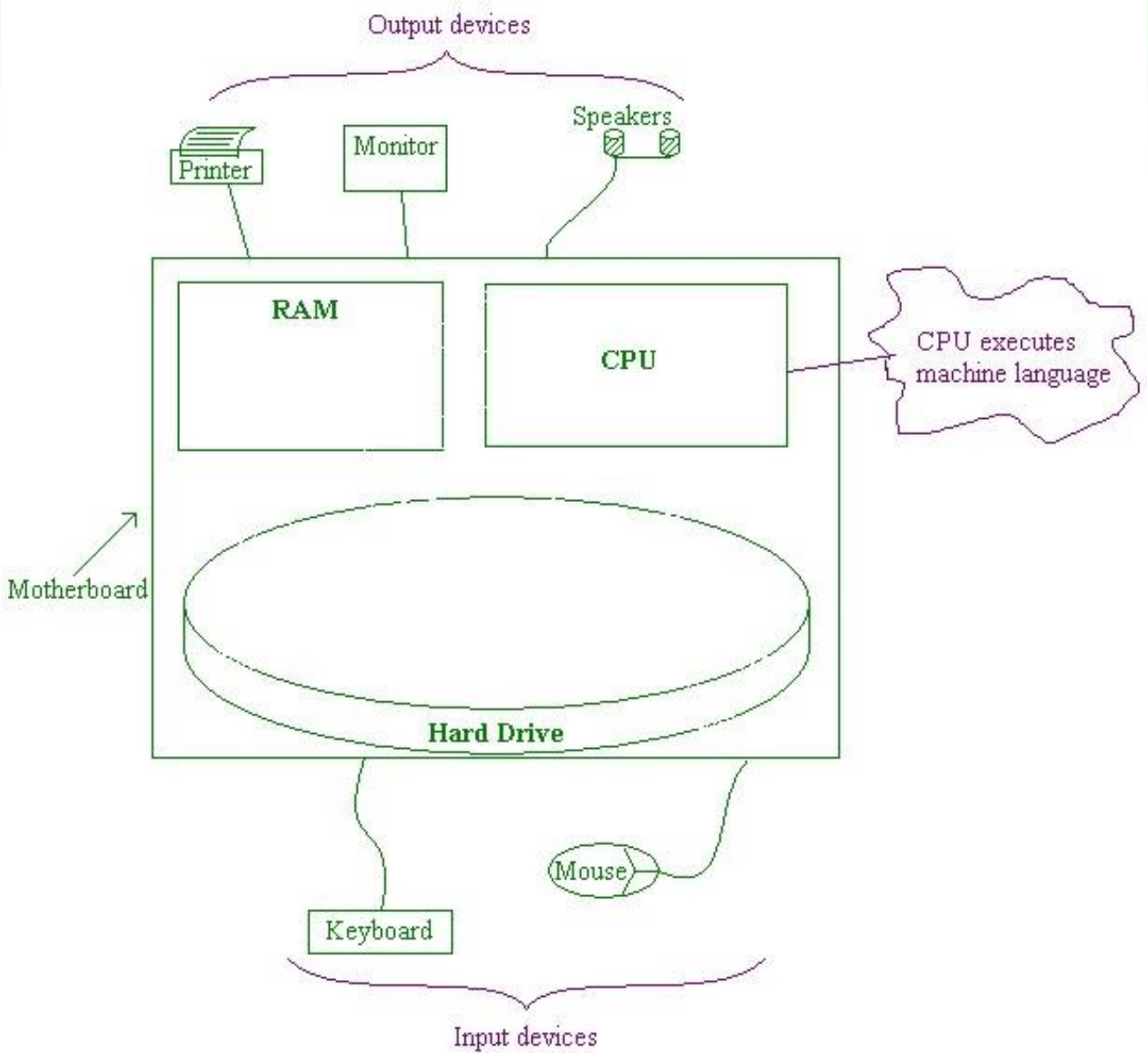
- If the class seems too hard for you
  - enroll in the advisory courses first
    - like MATH263, ENGL111/112, CSIS9
  - then come back to CSIS10A
- A solid foundation in English and Math
  - will prepare you for much of what you will discover at MPC and beyond!
- Java has a number of advanced concepts
  - CSIS9 provides a gentler programming intro

Overall Goal for Class

- The goal is for you to think like a computer scientist.
  - Most important skill is problem-solving.
  - formulate problems
  - think creatively about solutions
  - express solutions neatly and accurately

- The process of learning to program is an excellent opportunity to practice problem-solving skills.

Basic computer anatomy



What the parts do

- CPU
  - ``The brain''; performs relatively basic operations
  - It only executes *machine language*
  - The machine language varies from CPU to CPU
- Storage
  - Primary storage/random-access memory/RAM/``memory''
    - Fast, but volatile and expensive
  - Secondary storage/hard drive/hard disk
    - Cheap and non-volatile, but slow
- Input devices (the ``I'' in ``I/O'')
  - Mouse, keyboard
- Output devices (the ``O'' in ``I/O'')
  - Monitor, speakers, printer

What is a program?

- A program is a sequence of instructions that explain how to perform a computation
- Types of computations:
  - mathematical:
    - Example: solving system of equations
    - computing cost of a grocery purchase
  - symbolic:
    - Example: search/replace text in a doc
    - compiling a program

Example: A First Program

```
class Hello
{
  // main: generate some simple output

  public static void main(String[] args) {

    System.out.println("Hello, world.");
  }
}
```

- Can you guess what the task is?

- Watch what happens when it runs

Program Statements

- The instructions in a program are called statements – only 5 different types
  1. input: Get data from the keyboard or some other device.
  2. output: Display data on the screen or other device.
  3. math: Perform basic operation like addition and multiplication.
  4. testing: Check for certain conditions and run the appropriate sequence of statements.
  5. repetition: Perform some action repeatedly, usually with some variation.

Larger Tasks are Broken Down into Smaller

- Every program is made from the 5 operations.
- Programming is the process of breaking a large, complex task into smaller and smaller subtasks
  - until the subtasks are performed with one of these basic operations.
  - which operation does the Hello World program use?

High Level Languages

- **High-level language** (readable by humans)
  - Java, Python, C or C++, and Perl.
  - almost all programs are written in high level languages
  - shorter, easier to program, portable so they run on different computers
  - easier to debug

Low Level Languages

- **Low-level language** (readable by computers)
  - sometimes called **machine language**
  - computers can only run programs written in low-level languages
  - each computer processor (Intel vs Motorola vs AMD) runs different machine language
  - Programs have to be **translated** into machine language before they can run.

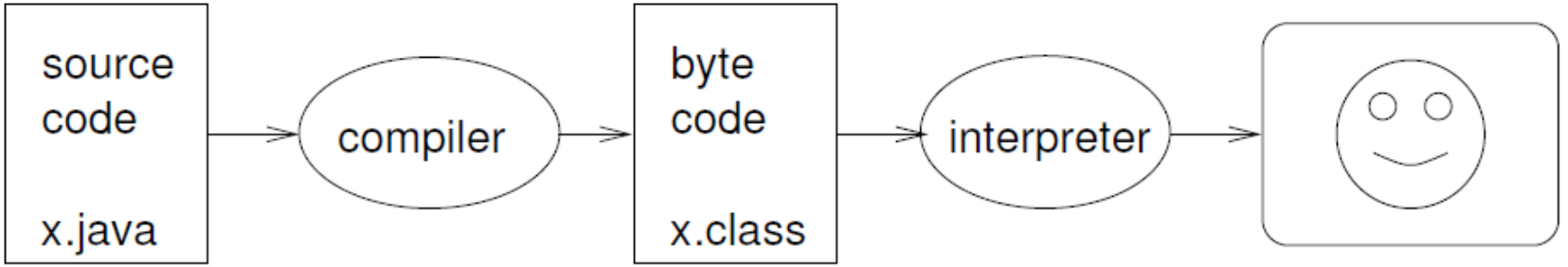
Compilers vs Interpreters

- Two ways to translate a program:
- **Interpreter**: translates line-by-line
  - alternately read line and carrying out command
  - translation takes place every time program is run
- **Compiler**: translates all at once
  - before running any of the commands
  - high-level program is the source code
  - translated program is object code or “executable”
  - compile once, run the compiled code later.

Java is both compiled and interpreted

- **Java Compiler** generates byte code
- **Java Virtual Machine (JVM)** interprets byte code
  - Byte code is like machine language
  - Byte code is portable, like a high-level language.
  - Can compile a program on one machine, interpret the byte code on another machine
  - This ability is an advantage of Java over many other high-level languages.

Running a program in Java



The compiler reads the source code...

... and generates Java byte code.

A Java interpreter reads the byte code...

... and the result appears on the screen.

Source File

- A source file contains source code and is really just a simple text file. It contains (among other things) instructions for the computer to execute.
  - A ``.java'' ending is used to distinguish it as a Java source file
  - Java files have special structure so that the computer can translate it into machine code.
  - Use a text editor or an IDE (in this class we use BlueJ but you can also use Eclipse) to create source and make changes to it.
  - We'll make a simple "Hello World" program now!

Java source file Hello.java

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

Java Bytecode for previous slide (Hello.class)

202 254 186 190 0 0 0 49 0 34 1 0 10 72 101 108 108 111 87 111 114 108 100 7 0 1
1 0 16 106 97 118 97 47 108 97 110 103 47 79 98 106 101 99 116 7 0 3 1 0 6 60 105
110 105 116 62 1 0 3 40 41 86 1 0 4 67 111 100 101 12 0 5 0 6 10 0 4 0 8 1 0 15 76
105 110 101 78 117 109 98 101 114 84 97 98 108 101 1 0 18 76 111 99 97 108 86 97
114 105 97 98 108 101 84 97 98 108 101 1 0 4 116 104 105 115 1 0 12 76 72 101
108 108 111 87 111 114 108 100 59 1 0 4 109 97 105 110 1 0 22 40 91 76 106 97
118 97 47 108 97 110 103 47 83 116 114 105 110 103 59 41 86 1 0 16 106 97 118 97
47 108 97 110 103 47 83 121 115 116 101 109 7 0 16 1 0 3 111 117 116 1 0 21 76
106 97 118 97 47 105 111 47 80 114 105 110 116 83 116 114 101 97 109 59 12 0 18
0 19 9 0 17 0 20 1 0 13 72 101 108 108 111 44 32 87 111 114 108 100 33 8 0 22 1 0
19 106 97 118 97 47 105 111 47 80 114 105 110 116 83 116 114 101 97 109 7 0 24 1
0 7 112 114 105 110 116 108 110 1 0 21 40 76 106 97 118 97 47 108 97 110 103 47
83 116 114 105 110 103 59 41 86 12 0 26 0 27 10 0 25 0 28 1 0 4 97 114 103 115 1 0
19 91 76 106 97 118 97 47 108 97 110 103 47 83 116 114 105 110 103 59 1 0 10 83
111 117 114 99 101 70 105 108 101 1 0 15 72 101 108 108 111 87 111 114 108 100
46 106 97 118 97 0 33 0 2 0 4 0 0 0 0 0 2 0 1 0 5 0 6 0 1 0 7 0 0 0 47 0 1 0 1 0 0 0 5 42
183 0 9 177 0 0 0 2 0 10 0 0 0 6 0 1 0 0 0 14 0 11 0 0 0 12 0 1 0 0 0 5 0 12 0 13 0 0 0 9
0 14 0 15 0 1 0 7 0 0 0 55 0 2 0 1 0 0 0 9 178 0 21 18 23 182 0 29 177 0 0 0 2 0 10 0 0
0 10 0 2 0 0 0 16 0 8 0 17 0 11 0 0 0 12 0 1 0 0 0 9 0 30 0 31 0 0 0 1 0 32 0 0 0 2 0 33

10 min Break
then Demo Pair Programming (Show video)

- Pair up with the person sitting next to you

- One person types
  - the other “steers” (checks for errors)
  - Change Roles!!! Every 10 minutes

- Use BlueJ to write, compile, and run
  - The “Hello World” program

Debugging

- Programming errors are called **bugs**
- the process of tracking them down and correcting them is called **debugging**
- three kinds of errors that can occur in a program
  - **Syntax Errors**: mistakes in grammar
  - **Run-time Errors**: something unexpected happens at runtime
  - **Logic or Semantic Errors**: program computes wrong result

Syntax Errors – found by Compiler

- **Syntax** refers to the structure of your program and the rules about that structure.
  - in English, a sentence must begin with a capital letter and end with a period.
    - this sentence contains a syntax error.
    - So does this one
- Compilers are not forgiving
  - a single syntax error stops the compiler from translating your program to machine language
  - prints out error message
  - demonstrate with Hello World program

Run-time errors -- detected by the interpreter (Java Virtual Machine)

- Run-time error – happens when you run program
  - In Java, when the JVM notices something goes wrong.
  - In Java, run-time errors are called exceptions
  - appear as window or dialog box that contain information about what happened
    - useful for debugging

- Demo run-time error in Hello World program: Divide by Zero

Logic or Semantic Errors – detected by programmer

- logic or semantic error -- program compiles and run but does not do the right thing.
  - does something else.
  - what you told it to do!
- The problem is the program you wrote is not the program you *wanted* to write.
- The semantics, or meaning of the program, are wrong.

Identifying Logic Errors

- work backwards
- look at the output of the program
- try to figure out what it is doing
- Demo: add screen buffer flush to be able to reset error messages.
  - Modify println to have empty parens with text in comment (no output)

Experimental Debugging

- like detective work:
  - given various clues, can you determine why you get the results you see?
- also like an experimental science
  - guess what is wrong
  - modify your program and try again.
  - If your hypothesis was correct, you are closer to a working program
  - If your hypothesis was wrong, you have to come up with a new one.

TIP: Use **Incremental Development**

- Programming is the process of gradually debugging a program until it does what you want.
  - **Always** start with a working program that does *something*
  - make *small* modifications, debugging them as you go, so that you always have a working program.
  - Baby steps take you to the goal
  - **Never** write the whole program then debug
    - You'll likely have to throw it out and start over!
    - Giant steps lead to disaster!

Breaking down the Hello World program

```
class Hello
{
  // main: generate some simple output

  public static void main(String[] args) {

    System.out.println("Hello, world.");
  }
}
```

- This program includes features that are hard to explain to beginners, but it provides a preview of topics we will see in detail later.

Class Definition

- Java programs are made up of class definitions, which have the form:

```
class CLASSNAME
{



}
```

- Here CLASSNAME indicates a name chosen by the programmer.
- The class name in the example is Hello.

Main Method

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

  System.out.println("Hello, world.");
}
```

- main is a **method**, which is a named collection of statements.
  - When the program runs, it starts at the first statement in **main** and ends at the last statement.
  - **main** can have any number of statements, but the example has only one: a print statement
  - The print statement ends with a semi-colon (;)
  - **System.out.println** is a method provided by one of Java's libraries.
  - A library is a collection of class and method definitions.

Curly Braces and Comments

- Java uses curly-braces { and } to group things
  - outermost squiggly-braces (lines 2 and 9) contain the class definition
  - inner braces contain the definition of main
- Line 3 begins with //. That means it's a **comment**
  - English text that you can put in a program, to explain what it does.
  - When compiler sees //, it ignores everything from there until the end of the line.

Start Lab 1, Part A

- We will continue with Part B on Thursday