Welcome to CSC 1214: Object-Oriented Programming

Lecture 1: Introduction

CSC 1214: Object-Oriented Programming

Marriette Katarahweire & Dr. Engineer Bainomugisha

Instructors

- Marriette Katarahweire
  Email: kmarriette@cit.mak.ac.ug
  Office: Block A - Room 202
  Student Hours: Mondays 11h:00-13h:00
- Dr. Engineer Bainomugisha
  Email: baino@cit.mak.ac.ug
  Office: Block A - Room 404
  Student Hours: Wednesdays and Fridays 11h:00-16h:00
- Tutorial Assistant: Stephen Mpirirwe

Course Details

- Course Code: CSC 1214
- Course Name: Object-Oriented Programming
  - *Previously (CSC 1207 - Programming Methodology II)*
- 3 hours of lecture
- 2 hours of lab sessions (time and venue to be communicated)
- Student expected to spend 4 hours practicing
- Course Prerequisites:
  - Computer Literacy
  - Structured Programming

Course Objectives

- The course is to give an in depth understanding of Object Oriented programming.
- Move the students programming skills from basic to advanced
- Avail students with skills to handle non-functional program aspects like robustness and security
- Train students to develop complete software applications

Course Outline

- The Object Oriented programming paradigm.
- Classes and Objects
- Inheritance and visibility modifiers
- Interfaces and abstract classes
- Graphical user interfaces and action handlers
- Exception handling
- Working with files and databases
- Sessions and user management

Assessment Style

- Tests and programming assignments (40%)
- Practical Examination (30%)
- Written Examination (30%)

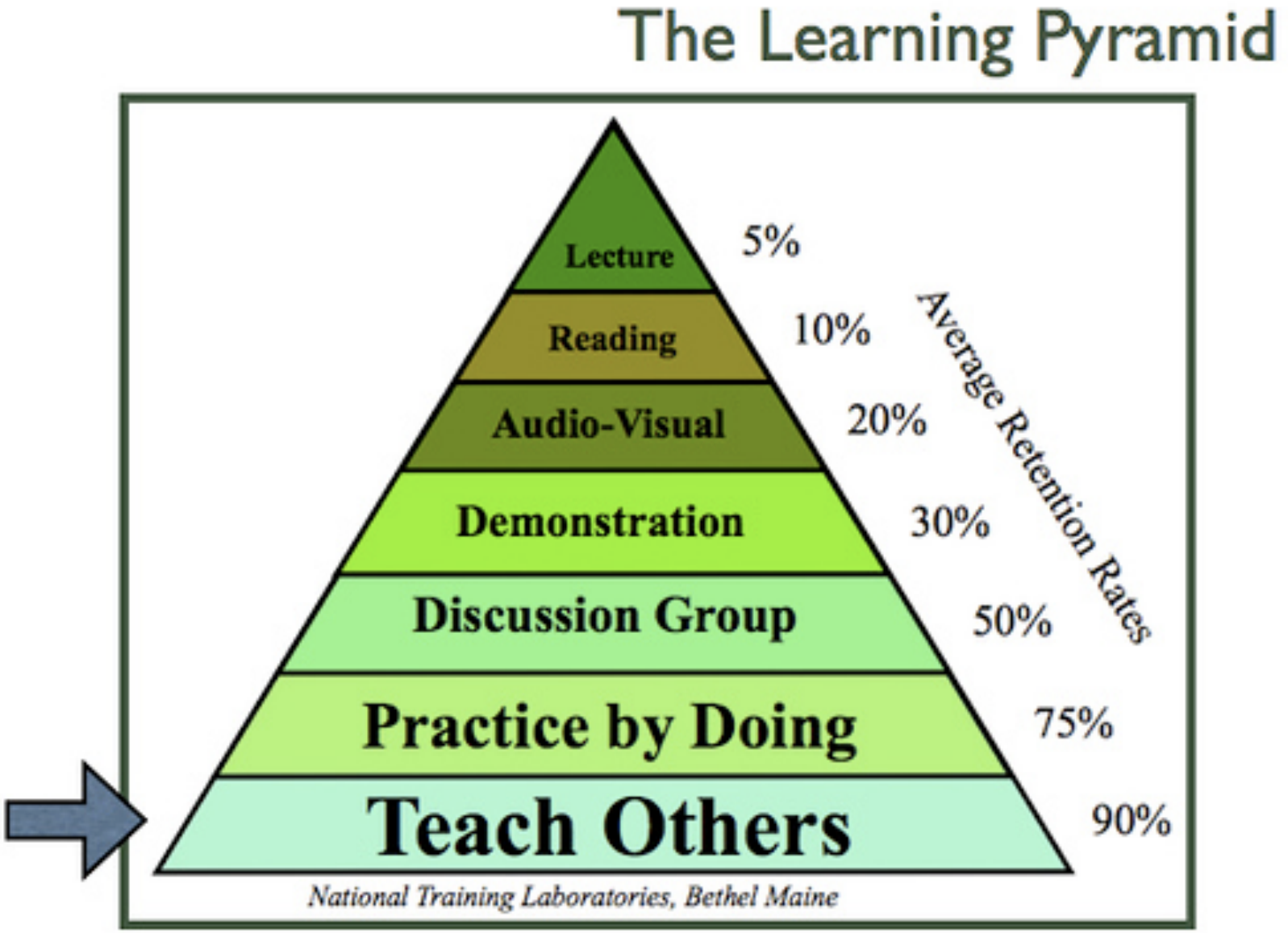
Reading References

- Java Software Solution: Foundations of Program Design by John Lewis and William Loftus
- Introduction to Java Programming by Y. Daniel Liang
- Internet Resources
  - e.g., Java Tutorials by Oracle
    http://docs.oracle.com/javase/tutorial/

Course Rules

- Classes are compulsory (attendance & participation = 10 marks)
- No phones allowed during class time
- Time keeping
- No movements in & out of class
- Work together but write own code - No Xeroxing friends work

Teaching & Learning Pattern



Let the fun begin!

Lecture I: Introduction (2)

CSC 1214: Object-Oriented Programming

Marriette Katarahweire & Dr. Engineer Bainomugisha

Focus of the Course

- Object-Oriented Software Development
  - problem solving
  - program design, implementation, and testing
  - object-oriented concepts
    - classes
    - objects
    - encapsulation
    - inheritance
    - polymorphism
    - graphical user interfaces
  - the Java programming language

Computer Processing

- Hardware: Physical tangible parts of a computer
- Software: Programs & data
  - program: a series of instructions
- A computer requires both software and hardware. Each is useless without the other

Software Categories

- Operating System
  - controls all machine activities
  - provides the user interface to the computer
  - manages resources such as CPU and memory
  - Examples??
- Application program
  generic term for any other kind of software
  examples: word processors, missile control systems, games, software development tools (or Kits)
- Most operating systems and application programs have a graphical user interface (GUI)

Software Development Kits/Tools

- SDKs are specialized application programs that allow programmers to write and test programs
- Experienced programmers generally prefer an Integrated Development Environment (IDE)
- Examples: Sun's Java SDK (JDK), Dr Java IDE, NetBeans, Eclipse

User Interface Styles

- Mainly 2 styles for any type of program:
  - Command Line Interface (CLI)
  - Graphical User Interface (GUI)
- As a computer programmer, you must be able to use and/or write programs for both styles of user interface

Command Line Interface (CLI)

- Computer types a Prompt requesting input
- User types a Command with parameters
- An old style of interaction that does not require a lot of computer power, but still in use today in some O/S and applications
- Not user friendly, but is very efficient when combined with scripting
- Example: DOS prompt, command & parameter

```
C:\> type file.txt
```

(display the contents of the file)

Graphical User Interface (GUI)

- Computer displays a combination of text and graphical symbols offering options to the user
- User manipulates mouse and uses keyboard to select from the offered options (hot keys) or to enter text
- More common now (computer power is cheap)
- Considered by most to be user friendly
- Examples: M/S Windows/Office or MAC O/S

Programming Languages

- A programming language specifies the words and symbols that we can use to write a program
- Has a set of rules that dictate how the words and symbols can be put together to form valid program statements
- Some languages are better for one type of program or one style of user interface than for others
- Examples??
- A programming language has both syntax and semantics

Syntax & Semantics

- The syntax rules of a language define how we can put together symbols, reserved words, and identifiers to make a valid program
- The semantics of a program statement define what that statement means (its purpose or role in a program)
- A program that is syntactically correct is not necessarily logically (semantically) correct
- A program will always do what we tell it to do, not what we meant to tell it to do
- Example: $5 + 3$

Program Translation

- A program must be translated into machine language before it can be executed
- A compiler is a software tool which translates source code into a specific target language. Target language is the machine language for a particular CPU type
- The Java approach is somewhat different

Java History

- The Java programming language was created by James Gosling at Sun Microsystems, Inc.
- Was introduced in 1995 and it's popularity has grown quickly since
- Is a high-level language

Java Properties

- Object-oriented
- Simple
- Automatic garbage collection
- Portable
- Multi-threaded programming
- Secure
- Internet aware
- Distributed
- Architecture neutral
- Dynamic

Java Translation

- The Java compiler translates Java source code into a special representation called bytecode
- Java bytecode is not the machine language for any traditional CPU
- Another software tool, called an interpreter, translates bytecode into machine language and executes it. Implies the Java compiler is not tied to any particular machine
- Java is considered to be architecture-neutral

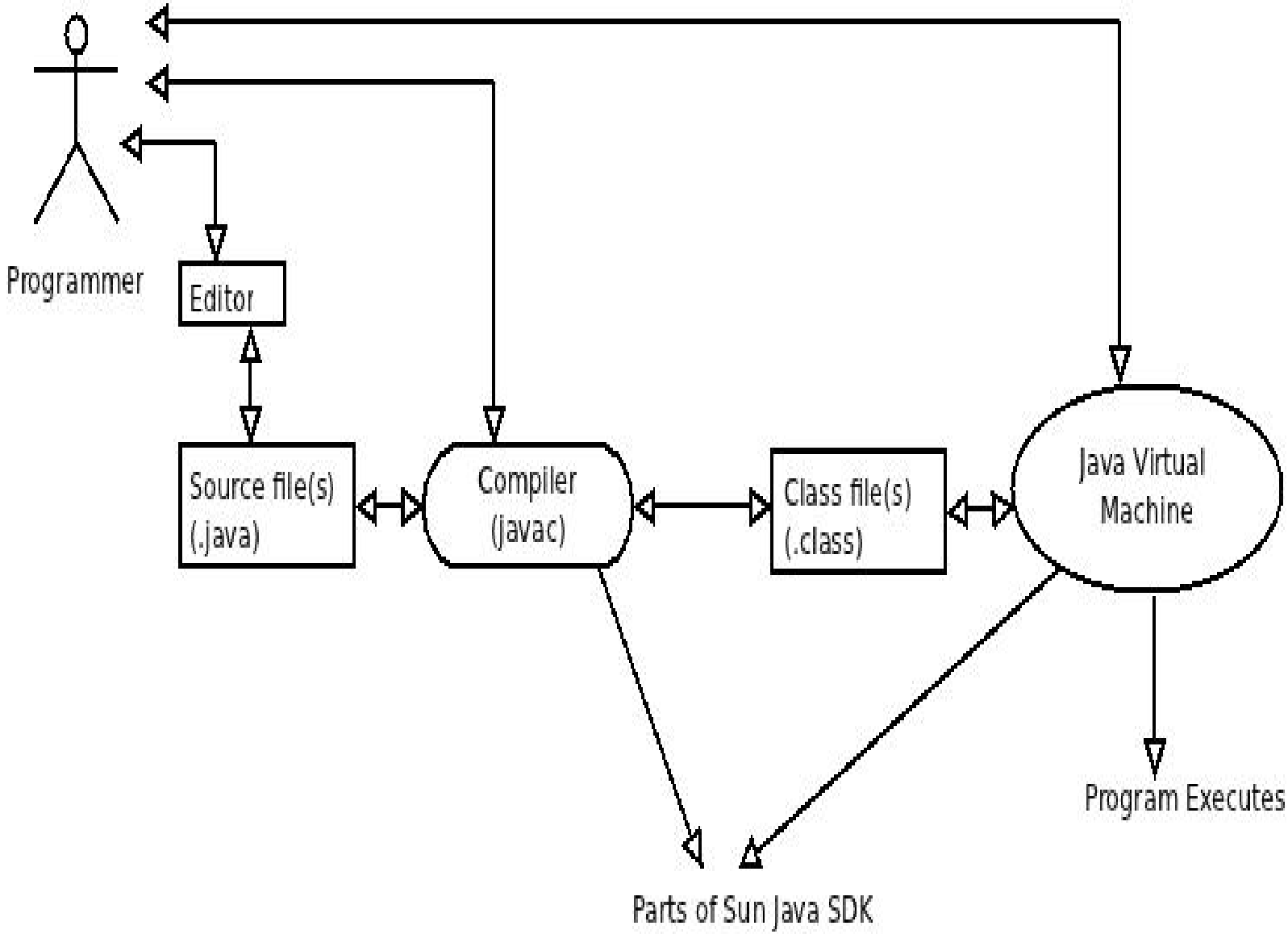
JDK Editions

- Java Standard Edition (J2SE): used to develop client-side standalone applications or applets.
- Java Enterprise Edition (J2EE): used to develop server-side applications such as Java servlets and Java ServerPages.
- Java Micro Edition (J2ME): used to develop applications for mobile devices such as cell phones.

Java Development Environments

- There are many programs that support the development of Java software, including:
  Sun Java Development Kit (JDK)
  Sun NetBeans
  IBM Eclipse
  Borland JBuilder
  DrJava
- Though the details of these environments differ, the basic compilation and execution process is essentially the same

Sun Java's JDK



Sun Java's SDK

- programmer writes source code with files end in ".java" extension
- java compiler (javac) converts (compiles) source code into "bytecode" (files ending in ".class"). Bytecode is "machine code" for Java Virtual Machine (JVM)
- Example:

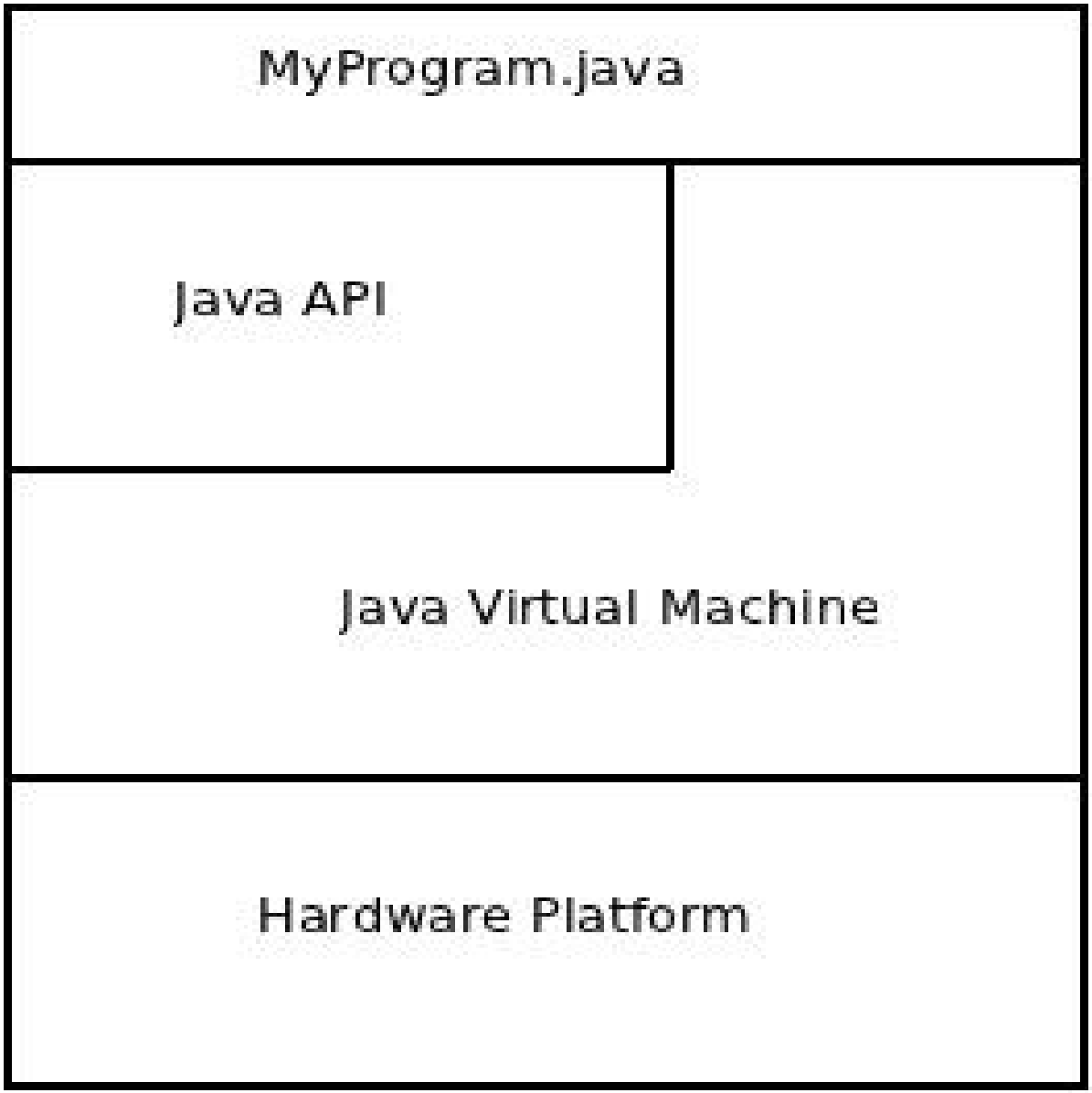
```
C:\> javac HelloWorld.java
 >> HelloWorld.class
C:\> java Hello
```

The Java Platform

- A platform: software or hardware environment in which a program runs.
- Java platform components:
  Java Virtual Machine (JVM)
  Java Application Programming Interface (API)
- Java API
  a collection of ready-made software components that provide many useful capabilities including:
  - graphics
  - networking
  - database
  - input/output

  Grouped into libraries of related classes and interfaces. These libraries are called packages.

Java Platform



Java Program Structure

In the Java programming language:

- A program is made up of one or more classes
- A class contains one or more methods
- A method contains program statements. A Java application always contains a method called main

These terms will be explored in detail throughout the course

Java Program Structure

```
class Welcome {
  private static String my_name = "Arinda";

  public static void main (String [] args){
     System.out.print("Welcome " + my_name);
  }
}
```

The main Method

- Every progam must contain a main method
- Is similar to the main function in C
- Its the entry point for your application and will subsequently invoke all the other methods required by your program.
- Accepts a single argument: an array of elements of type String `public static void main(String [] args)`
- This array is the mechanism through which the runtime system passes information to your application. `java MyProgram arg1 arg2`
- Each string in the array is called a *command-line argument*. Command-line arguments let users affect the operation of the application without recompiling it. For example, a sorting program might allow the user to specify that the data be sorted in descending order with this command-line argument: `-descending`

Command Line Example

```
public class Echo {
  public static void main (String[] args) {
      for (String s: args) {
          System.out.println(s);
      }
  }
}
```

Language Basics

- Variables
- Operators
- Expressions, Statements, and Blocks
- Control Flow Statements