Computer Programming I - CSC111

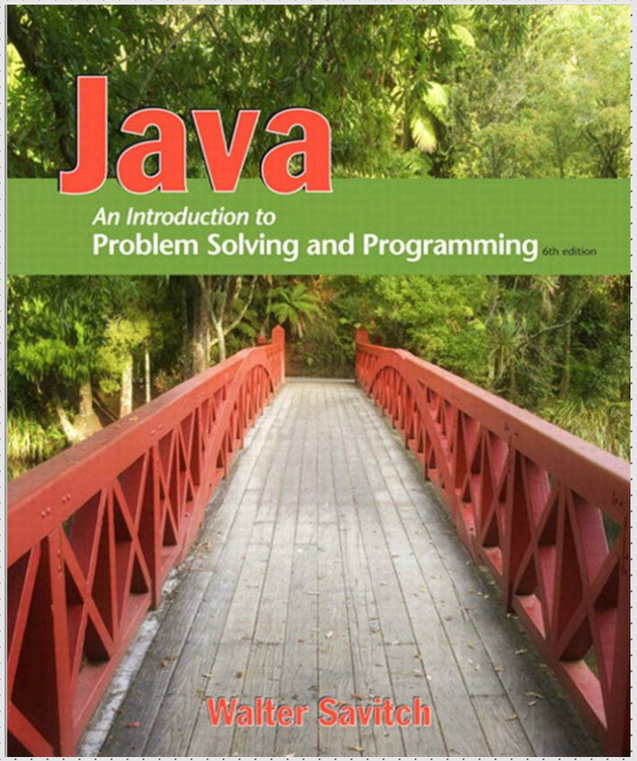
Introduction

Dr. Mejdl Safran

mejdl@ksu.edu.sa

Syllabus

- Textbook
  - *Java: An Introduction to Problem Solving and Programming,7ed, W. Savitch, Pearson International (Textbook)*
  - Java How to Program, 7ed, Deitel and Deitel, Pearson International (Reference)
  - Introduction to Java Programming, Comprehensive Version, 10ed Y. Daniel Liang, Prentice Hall (Reference)



Content

Assessment Methods & Policy

Assignments , Quizzes & Attendance	10%
Class Project	5%
Lab Exams	10% + 15%
Midterms	10% + 10%
Final exam	40%

Homework Assignments & Quizzes

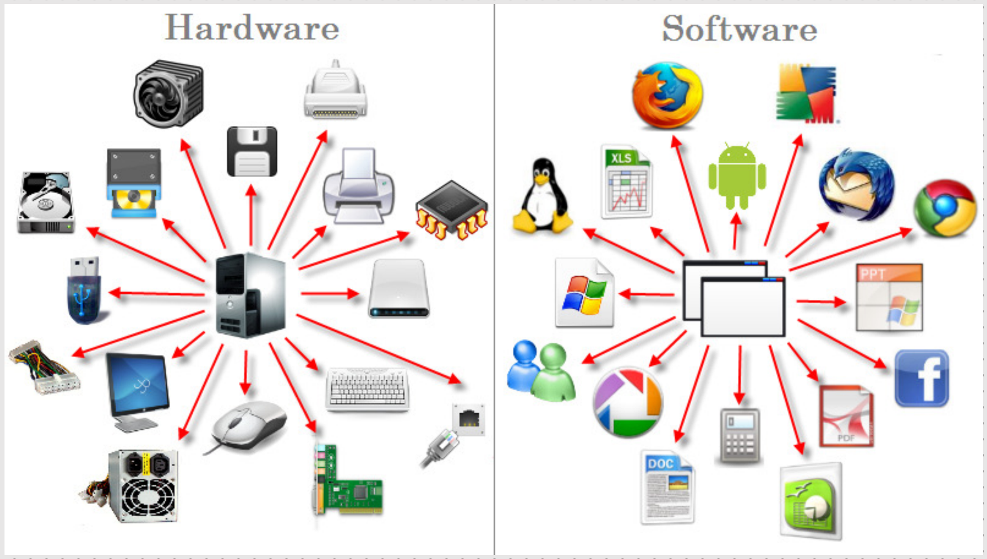
- Homework will be assigned and graded
- All homework assignments will be given with a strict deadline, and students are required to submit assignments on or before the deadline.
- *Cheating will not be tolerated.*
- All homework assignments or project documents should be submitted using MS-Word and/or appropriate computer software.
- *No hand written submission will be accepted*.
- In-class quizzes will be given throughout the semester

Chapter Outline

- What a computer is
- What a computer program is
- The Programmer’s Algorithm
- How a program that you write in Java is changed into a form that your computer can understand
- Characteristics of Java

Hardware & Software

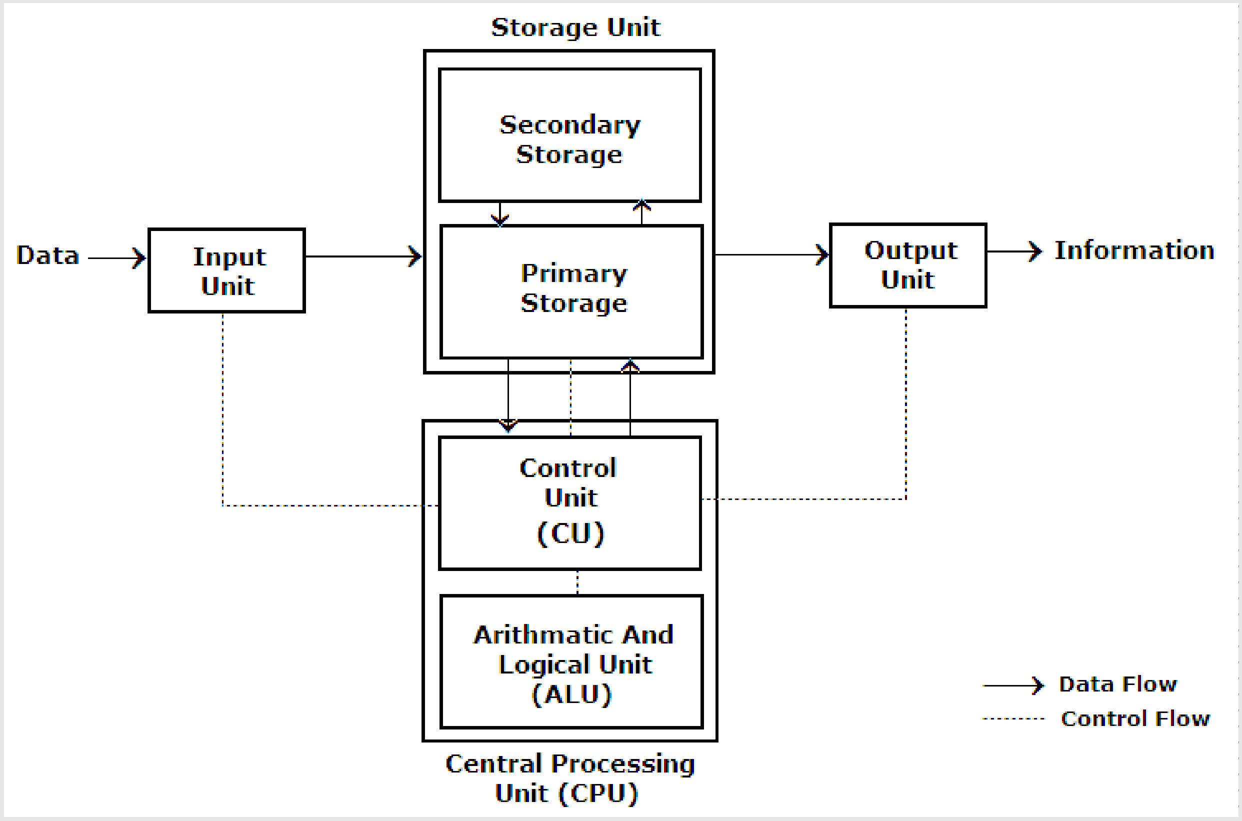
- Computer systems consist of *hardware* and *software*.
- Familiarity with hardware basics helps us understand software.



Components of computer system

- Most modern computers have similar components including
  - Input devices (keyboard, mouse, etc.)
  - Output devices (display screen, printer, etc.)
  - A processor (CPU): processes a program’s instructions
  - Two kinds of memory:
    1) Main memory (RAM): temporary storage
    2) Secondary Memory: persistent storage

Components of computer system



Programs

- A *program* is a set of instructions for a computer to follow.
- We use programs almost daily (email, word processors, video games, bank ATMs, etc.).
- Programs are stored in files.
- Programs files are copied from secondary memory to main memory in order to be run.

How to write a program

Levels of Abstraction

- Human thought
- Pseudo-Natural Language (Arabic, English)
- High-level Programming Language (C, C++, Java, …)
- Machine Code

The Programmer’s Algorithm

- An **algorithm** is a finite sequence of instructions that produces a solution to a problem.

- The programmer’s algorithm:
  - Define the problem
  - Plan the problem solution
  - Code the program
  - Compile the program
  - Run the program
  - Test and debug the program

Defining the problem

- The problem must be defined in terms of:
  - *Input*: data to be processed
  - *Output*: the expected results
    - Look for nouns in the problem statement that suggest output and input
  - and *processing*: the statements to achieve
    - Look for verbs to suggest processing steps

Example: sum and average of 5 numbers

- Input:
  - Five numbers: x1, x2, x3, x4, x5
- Processing:
  - Sum = x1 + x2 + x3 + x4 + x5
  - Average = Sum/5
- Output:
  - Sum
  - Average

Planning the solution

- When planning, algorithms are used to outline the solution steps using Englishlike statements, called *pseudocode*
- Simple pseudocode:

```
1.    Start program
2.    Get five numbers (x1, x2, x3, x4, x5)
3.    Add them (sum = x1 + x2 + x3 + x4 + x5)
4.    Compute average (avg = sum / 5)
5.    Print sum & avg
6.    End program
```

Coding the program

- Coding is writing the program in a formal language called *programming language*.
- The program is written by translating the algorithm steps into a programming language statements
- The written program is called *source code* and it is save in a file with ".java" extension.

Why coding in programming languages?

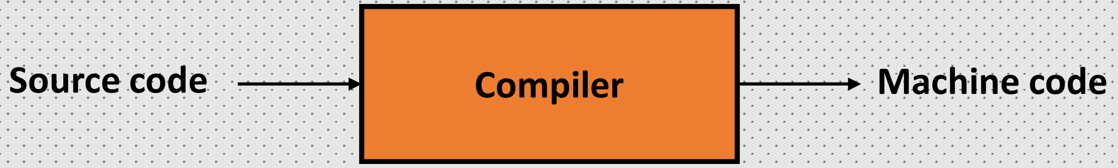
Why coding in programming languages

- *High-level languages* are relatively easy to use
  - Java, C#, C++, Visual Basic, Python, Ruby.
- Unfortunately, computer hardware does not understand high-level languages.
  - Therefore, a high-level language program must be translated into a *low-level language (machine code).*

How to translate source code into machine code?

Compiling computer programs

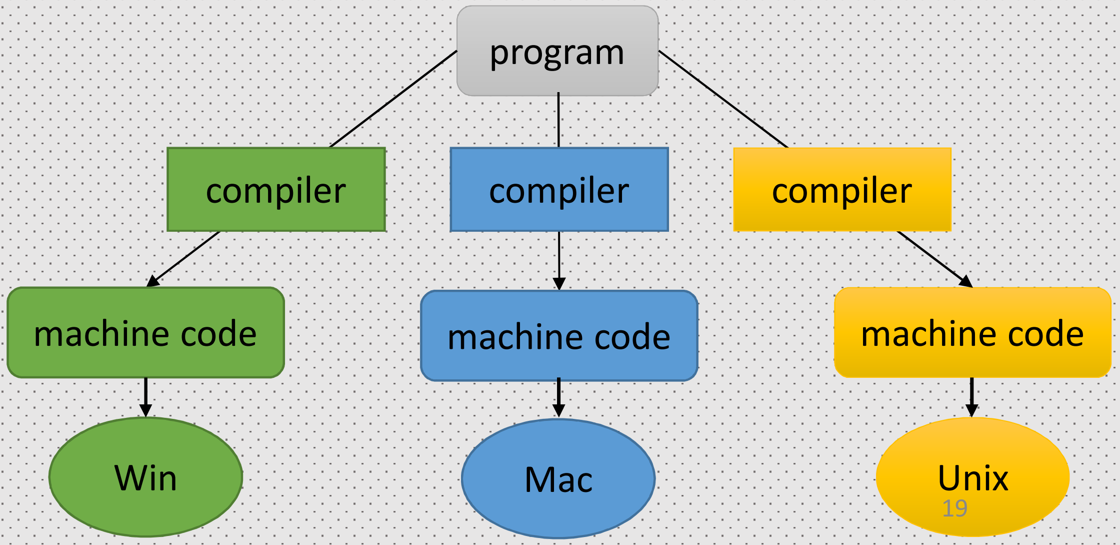
- A *compiler* translates a program from a high-level language to a low-level language the computer can run.



- The compiler
  - *checks correctness* of the source code (*syntax errors*)
  - *translates* the source code into a machine code if no errors were found

Platform dependent compiling

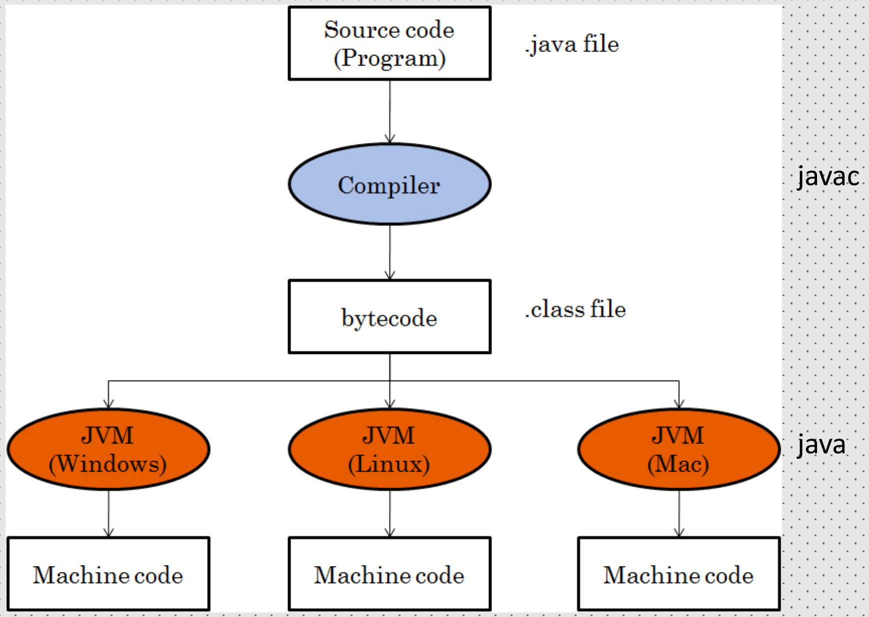
- Most high-level languages need a *different compiler* for each type of computer and for each operating system.
- Most compilers are very large programs that are *expensive* to produce.



How to run a Java program on each computer, with no need to recompile

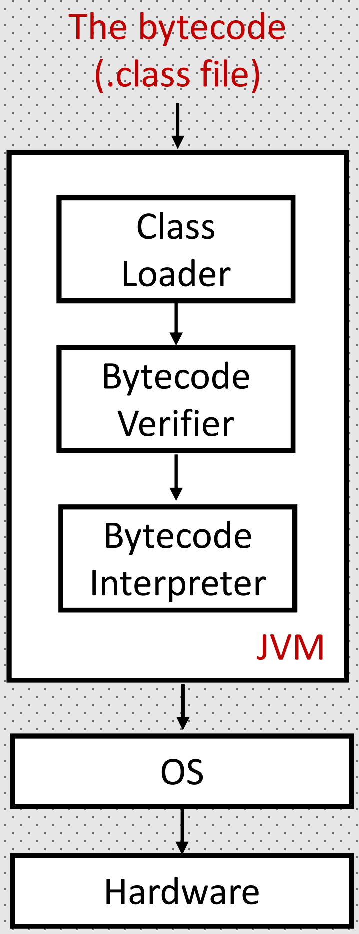
Compiling Java Programs

- The Java *compiler* does not translate a Java program into *machine code* for a particular computer.
- Instead, it translates a Java program into *bytecode*.
- Bytecode is converted into machine code using *Java Interpreter*
- *Platform independent!*



Running Java Programs

- The Java Virtual Machine (JVM):
  - Class Loader
    - stores bytecodes in memory
  - Bytecode Verifier
    - ensures bytecodes don’t violate security requirements
  - Bytecode Interpreter:
    - translates bytecodes into machine codes



Testing and Debugging the program

- Testing
  - Be sure that the output of the program conforms with the input
  - Two types of errors:
    - *Logical errors*: the program runs but provides wrong output
    - *Runtime errors*: the program stops running suddenly when asking the OS executing a non accepted statement (divide by zero, etc.)
- Debugging
  - Find, understand and correct the errors

Some characteristics of Java

- *Object-oriented programming* (OOP)
  - Treats program as a collection of **objects** that interact by means and actions
  - *Encapsulation* (information hiding)
  - Data *Abstraction* (implementation hiding)
  - *Inheritance* (Undergraduate is a subclass of Student)
  - *Polymorphism* (single action in different ways)
- Platform independent
  - Portable
  - Architecture neutral
  - “Write-once, run anywhere”
- Secure
  - Bytecode verifier of the VM

Setting up your Development Environment

- Download latest JDK from

https://www.oracle.com/technetwork/java/javase/downloads/jdk10-downloads-4416644.html

- Java Development Kit (JDK) is required to develop and compile programs
- Java Runtime Environment (JRE) is required to run programs
- Users must have JRE installed
- Developers must have the JDK installed
- JDK includes the JRE

What is an IDE?

- IDE = Integrated Development Environment
- Makes you more productive
- Includes text editor, compiler, debugger, syntax highlighting and code completion, etc.
- Eclipse is the most widely used IDE
- Alternatives:
  - NetBeans (Oracle)
  - IntelliJ IDEA (JetBrains)

Installing Eclipse

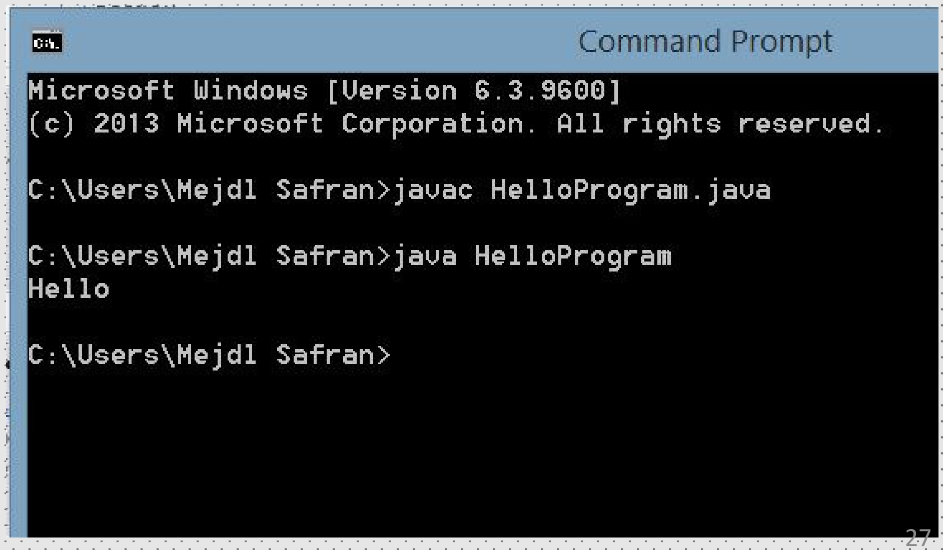
- Download and install the latest Eclipse for Java from:

http://www.eclipse.org/downloads/packages/

- Unzip the content of the archive file you downloaded
- To start Eclipse
  - On PC, double-click on Eclipse.exe
  - On Mac, double-click on Eclipse.app in Application folder

Run Java program – command line

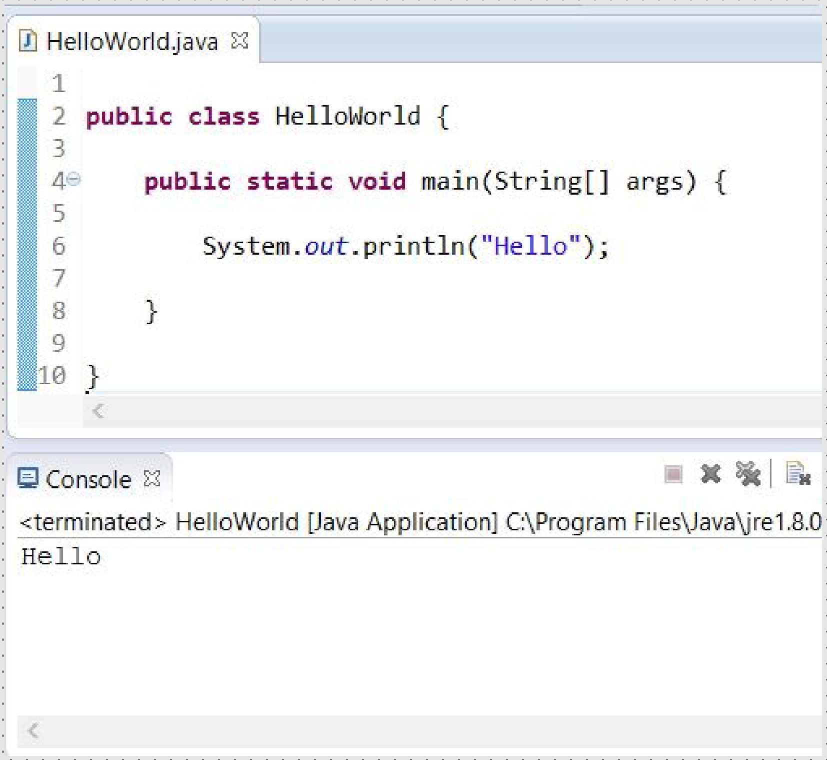
```java
public class HelloWorld {

      public static void main(String[] args) {

      System.out.println("Hello");

      }

}
```



```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\Mejdl Safran>javac HelloProgram.java

C:\Users\Mejdl Safran>java HelloProgram
Hello

C:\Users\Mejdl Safran>
```

Run Java program – Eclipse

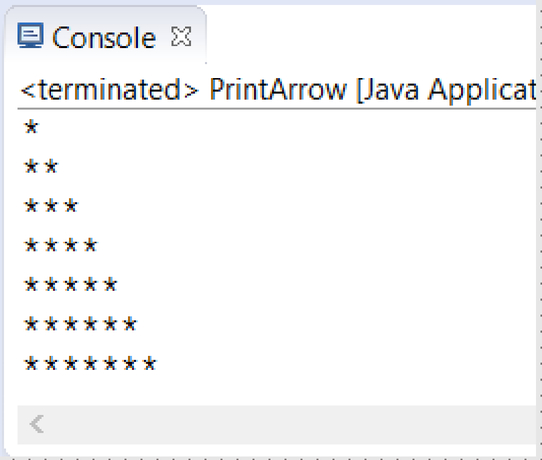
```
public class HelloWorld {

	public static void main(String[] args) {

	System.out.println("Hello");

	}

}
```



```
public class HelloWorld {

    public static void main(String[] args) {

        System.out.println("Hello");

    }

}
```

```
<terminated> HelloWorld [Java Application] C:\Program Files\Java\jre1.8.0
Hello
```

Print triangle of symbols



```
<terminated> PrintArrow [Java Applicat
*
**
***
****
*****
******
*******
```

```java
public class PrintTriangle {

public static void main(String[] args) {

        System.out.println("*");
        System.out.println("**");
        System.out.println("***");
        System.out.println("****");
        System.out.println("*****");
        System.out.println("******");
        System.out.println("*******");
    }
}
```

println() vs. print()

```
System.out.print("My name is ");
System.out.print("Mejdl");
```

```
My name is Mejdl
```

```
System.out.println("My name is ");
System.out.print("Mejdl");
```

```
My name is
Mejdl
```

```
System.out.print("My name is \n");
System.out.print("Mejdl");
```

```
My name is
Mejdl
```

```
System.out.print("Student name:\t");
System.out.print("Saad");
```

```
Student name:	Saad
```

```
System.Out.print("Student name:");
System.out.print("Saad");
```

```
Syntax error
```

```
System.out.println("Student name:");
system.out.println("Saad");
```

```
Syntax error
```