CS-121
Programming I

Dr. Nagia M. Ghanem
nagia.mghanem@gmail.com

Programming I

- **Lectures:**
  – Sunday & Monday 3rd Lecture

- **Section & Lab:**
  – TA : Eng. Mohammad Abdelaziz mohammad.abdelaziz.2011@gmail.com

- **Grading:**
  - Total = 150
  – Final Exam = 105
  – Year's Work = 45
  ◦ Midterm ~ 20
  ◦ Assignments & Project ~ 25

References

- Textbook:

Introduction to Programming Using Java, 6th edition, David J. Eck, 2011, ISBN: 1441419764

- Other books:
  - Java 2, the Complete Reference, 7th edition, Herbert Schildt, 2007, ISBN: 0-07-222420-7
  - Thinking in Java - 4th editon, Bruce Eckel, 2006, ISBN: 0131872486
  - Core Java™2: Volume I – Fundamentals, Fifth Edition, ISBN: 0132354764

- Java Web Site:

java.sun.com

Course Contents

- Introduction to programming and Java
- Types, variables, operators
- Selection statements (if, switch)
- Repetition statements (for, while, do-while)
- More on OOP concepts (objects, classes, instantiation, ..)
- Methods and parameter passing (call by value / call by reference)
- Arrays
- Strings
- Recursion
- Searching & Sorting

Course Objectives

- Understand what is computer programming.
- Learn the steps for problem solving (algorithm development and program design).
- Understand programming language concepts.
- Learn the Java programming language: its syntax, expressions, patterns, and styles.

It is not enough just to write code that works. It is as important--perhaps more important--to write code well; not merely code that works, but code that is legible, maintainable, reusable, fast, and efficient.

Introduction

- **A Computer Program** is a sequence of instructions written to perform a specified task with a computer.
- A computer's Central Processing Unit (CPU) can only carry out instructions that are written in a simple language called machine language.
- If the program is written in another language, it must first translated into machine language.

Introduction

If we have a problem:

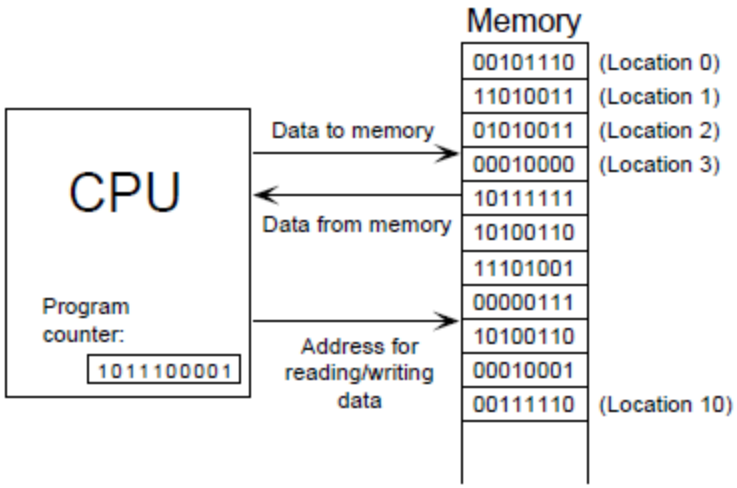
Q1) How do computers execute the instructions of the program?

Q2) How can we design and develop a program to solve the problem? What languages can we use?

Q3) If we are using another language (not machine language), how can we translate into machine language?

How Computers Execute Instructions

- Central Processing Unit (CPU) is responsible for executing instructions.
- Programs and data are stored in the Main Memory (RAM).
- Fetch-Decode-Execute Cycle
- Address of instruction to be executed is stored in program counter register (PC).



Program Execution

- Steps taken by the CPU to run a program (instructions are in machine language):

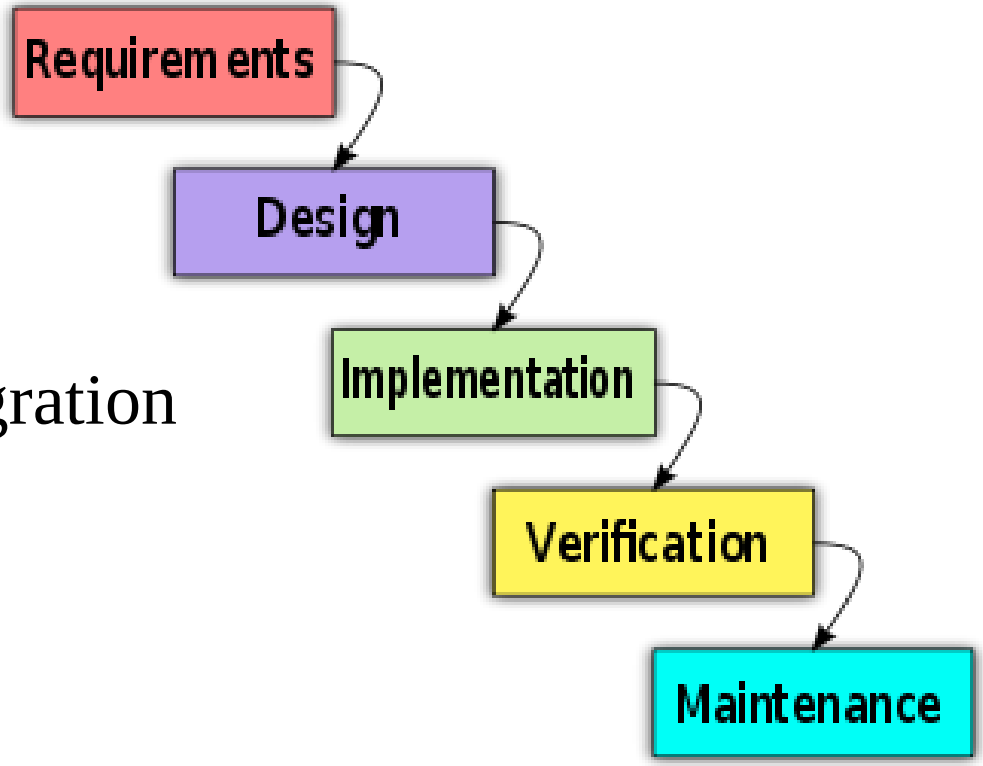
1. Fetch an instruction
2. Decode (interpret) the instruction
3. Retrieve data, if needed
4. Execute (perform) actual processing
5. Store the results, if needed

How to Develop Computer Programs

- Software development is the process of writing and maintaining the software, ideally in a planned and structured process.
- Software Development Process
- Programming Languages Hierarchy
- Programming Paradigms

Software Development Process

- Requirements analysis
- Software design
- Implementation and Integration
- Verification
- Maintenance



Unmodified Waterfall Model

Programming Language Hierarchy

- Machine Language
- Assembly Language
- High-Level Language (HLL)

Machine Language

- The representation of a computer program which is actually read and understood by the computer.
  - A program in machine code consists of a sequence of machine instructions.

- Instructions:
  - Machine instructions are in binary code
  - Instructions specify operations and memory cells involved in the operation

Example:

Operation	Address
0010	0000 0000 0100
0100	0000 0000 0101
0011	0000 0000 0110

Assembly Language

- A symbolic representation of the machine language of a specific processor.
- Is converted to machine code by an assembler.
- Usually, each line of assembly code produces one machine instruction (One-to-one correspondence).
- Programming in assembly language is slow and error-prone but is more efficient in terms of hardware performance.
- Mnemonic representation of the instructions and data
- **Example:**

```
Load    Price
Add     Tax
Store   Cost
```

High-level language

- A programming language which uses statements consisting of English-like keywords such as "FOR", "PRINT" or “IF“, ...etc
- Each statement corresponds to several machine language instructions (one-to-many correspondence).
- Much easier to program than in assembly language.
- Data are referenced using descriptive names
- Operations can be described using familiar symbols
- Example:

  Cost := Price + Tax

Programming Paradigms

- Structured Programming (Top-down programming)
  - To solve a large problem, break the problem into several pieces and work on each piece separately.
  - To solve each piece, treat it as a new problem which can itself be broken down into smaller problems .
  - Repeat until we reach to pieces that cannot be more decomposed.
  - More focus on instructions than on data structures

Programming Paradigms

- Object Oriented Programming
  - Start by identifying the objects involved in a problem and the messages that those objects should respond to.
  - The program that results is a collection of objects, each with its own data and its own set of responsibilities.
  - The objects interact by sending messages to each other.

Programming Paradigms

- Object Oriented Programming
  - The central concept of object-oriented programming is the object , which is a kind of module containing data and subroutines.
  - An object is a kind of self- sufficient entity that has an internal state (the data it contains) and that can respond to messages (calls to its subroutines).

Why OOP?

- Save development time (and cost) by reusing code
  - once an object class is created it can be used in other applications
- Easier debugging
  - classes can be tested independently
  - reused objects have already been tested

Java

- Java is high level programming language that follows OOP methodology.

- A program written in a high-level language cannot be run directly on any computer. First, it has to be translated into machine language.

- Two types of translators:
  - Compilers
  - Interpreters

  A compiler takes a high-level-language program and translates it into an executable machine-language program.

Compilers & Interpreters

- **Compiler**
  - A program that converts another program from some source language (high-level language) to machine language (object code).
  - Each input statement, in general, correspond to more than one machine instruction.
- **Interpreter**
  - another way to translate source instruction-by-instruction to object code.
  - Translation is “on-line,” i.e. at run time.
  - Usually, interpreted programs are slower than compiled programs.

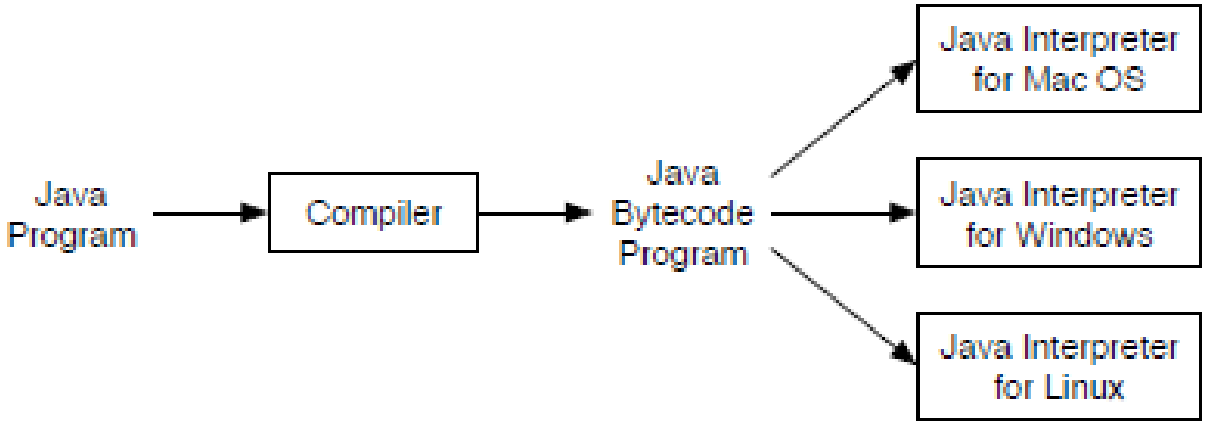
Java Program Translation

- **Both Compilation and Interpretation**

- Programs written in Java are compiled into a machine language called (Java bytecode).

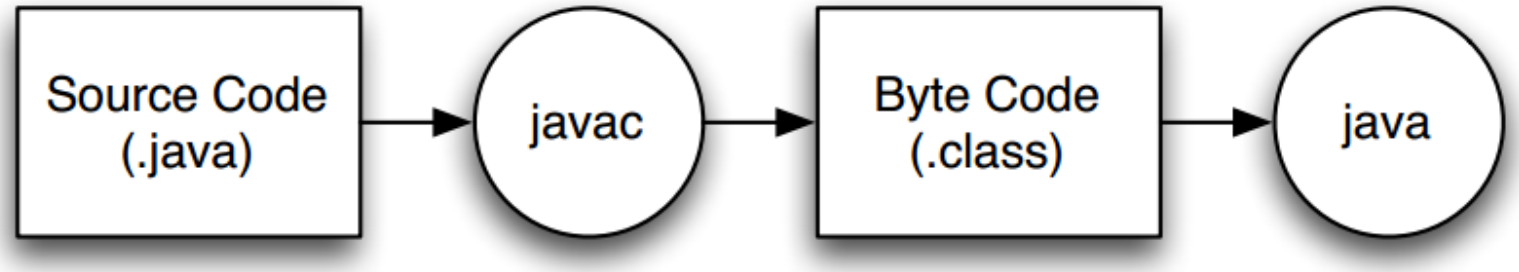
- Java bytecode is a machine language for a computer that doesn't really exist – (Java Virtual Machine, or JVM).

Java Virtual Machine



- A different Java bytecode interpreter is needed for each type of computer.
- The same Java bytecode program can be run on any computer that has such an interpreter.
- This is one of the essential features of Java: the same compiled program can be run on many different types of computers.

Compiling Java



First Java Program

```
public class Hello {
        public static void main(String[] arguments) {
            // Program execution begins here
            System.out.println("Hello world.");
        }
    }
```

Program Structure

```
public class CLASSNAME {
        public static void main(String[] arguments) {
            STATEMENTS
        }
    }
```

- Every executable Java program consists of a **class** that contains a **method** named main that contains the **statements** (commands) to be executed

Java Terminology

- **class**: A module that can contain executable code.
  - Every program you write will be a class.
- **statement**: An executable command to the computer.
- **method**: A named sequence of statements that can be executed together to perform a particular action.
  - A special method named main signifies the code that should be executed when your program runs.
  - Your program can have other methods in addition to main. (seen later)

System.out.println

- System.out.println : A statement to instruct the computer to print a line of output on the console.

- Two ways to use System.out.println :

  System.out.println("<***Message***>");

  - Prints the given message as a line of text on the console.

  System.out.println();

  - Prints a blank line on the console.

Comments – Two Kinds

- First kind of comments:

```
/* This is one kind of comment
that can span several lines. Don't
forget to put the closing
characters at the end.
*/
```

- Second kind of comments

```
// This is the other type of comment.
// It covers the entire line
// and requires a new set
// of slashes for each new line.
```

Syntax

- **syntax**: The set of legal structures and commands that can be used in a particular programming language.

- some Java syntax:
  - every basic Java statement ends with a semicolon ;
  - The contents of a class or method occur between { and }

Syntax errors

- **syntax error** or **compiler error**: A problem in the structure of a program that causes the compiler to fail.
  - If you type your Java program incorrectly, you may violate Java's syntax and cause a syntax error.

```
1 public class Hello {
2    pooblic static void main(String[] args) {
3       System.owt.println("Hello, world!")_
4    }
5 }
```

compiler output:

```
Hello.java:2: <identifier> expected
    pooblic static void main(String[] args) {
           ^
Hello.java:5: ';' expected
}
^
2 errors
```

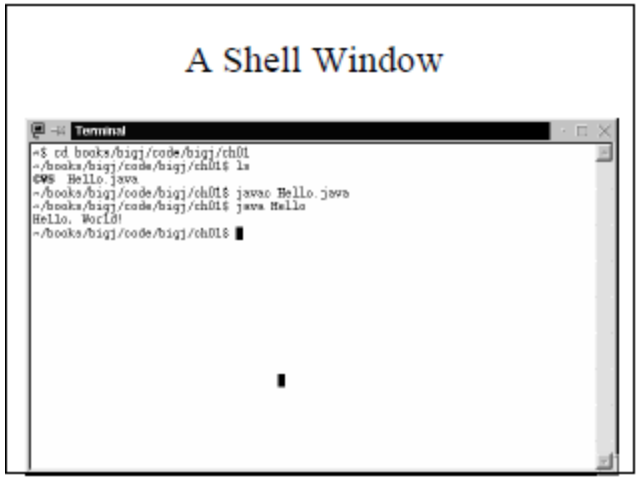
Other Error Types

- Runtime error:
  - When there are no syntax errors, but the program can't complete execution
    - Divide by zero
    - Invalid input data
- Logical errors:
  - The program completes execution, but delivers incorrect results
  - Incorrect usage of parentheses

Editors

- To write programs, you need a piece of software called an editor. They come in two flavors:
  - simple source code editors
  - complex integrated development environments (IDE)

A Shell Window



An Integrated Development Environment

Reading

- Read Chapter 1 from:
  - Introduction to Programming Using Java, 6th edition, David J. Eck, 2011, ISBN: 1441419764
    http://math.hws.edu/eck/cs124/downloads/javanotes6-linked.pdf