

CSC 110 Software Design and Programming

Lecture 01

Salem State University
Computer Science Department

Fall 2022

Instructor: Dr. Fatema Nafa

Course Information

2

Instructor	Dr. Fatema Nafa
Email	fnafa@Salemstate.edu
Office Hours	Tuesday- Thursday 02:00 PM – 03:00 PM via ZOOM & by appointment
Time	Wednesday - Friday 08:00AM – 09:15AM
Place	MH210

Friday	December 16	8:00AM - 10:00AM	MH 210
---------------	--------------------	-------------------------	---------------

Course Information

3

Prerequisites

High school algebra I & II, plus experience with a window-based operating system and the use of email and a word processor

Textbook



① **REQUIRED**

Java How to Program, Early Objects

Edition: 11th

ISBN: 9780134748559

Author: Deitel

Publisher: Pearson

Formats: BryceWave Format

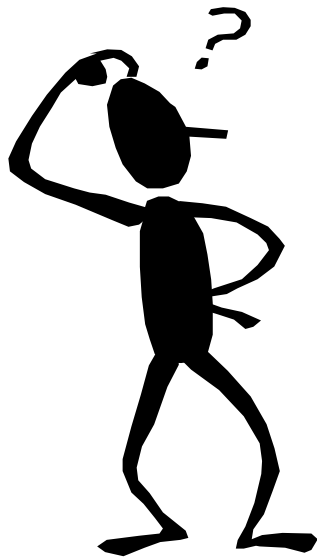
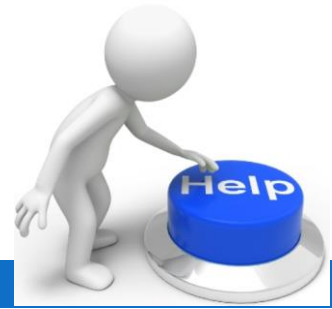
Rent

Digital [?](#)
[Requirements](#)

\$49.00
180 Days



Getting Help



- Help is always available!
- **Option 1:** Come to my Office Hours
 - ▣ Thursdays 2:00 pm – 4:00 pm & **by appointment**(I may change the time of my office hours)
 - ▣ **Location:** ZOOM
 - ▣ I get bored when nobody visits!
 - ▣ If you cannot make my office hours, I will be happy to make an appointment with you. Please try to give me advance warning when you need an appointment.
- **Option 2:** Check if I am online on Zoom you can chat with me any time.

Course Information

6

Reading the class Syllabus
Introduction to Java

Course Information

Let us read the syllabus together

Grading

Zoom Participations	5%
Labs	25%
Programming Assignments	25%
Quizzes	15%
Examination	30%

CSC 110 Topics

- ❑ **Computation/Programming**
- ❑ Variables, I/O
- ❑ Expressions
- ❑ Arrays
- ❑ Control Flow, Conditionals, Loops
- ❑ Methods
- ❑ Exceptions, File I/O
- ❑ Testing and Debugging

What Makes Up a Computer?

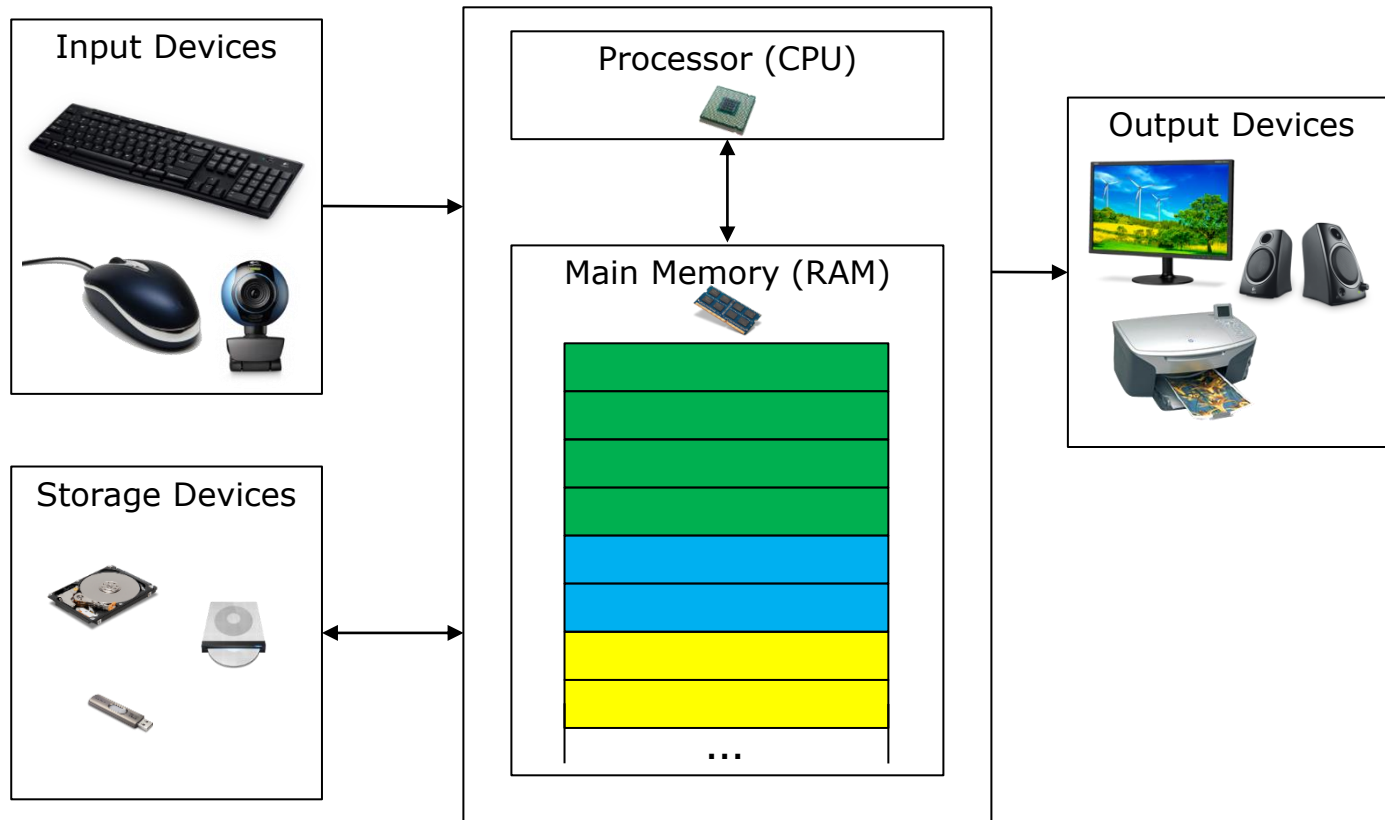
- **Hardware**

- Physical components
- Wide variety of types and manufacturers
- Abstracted to a simple set of ideas for Computer Science

- **Software**

- Programs (i.e., instructions)
- Wide variety of purposes

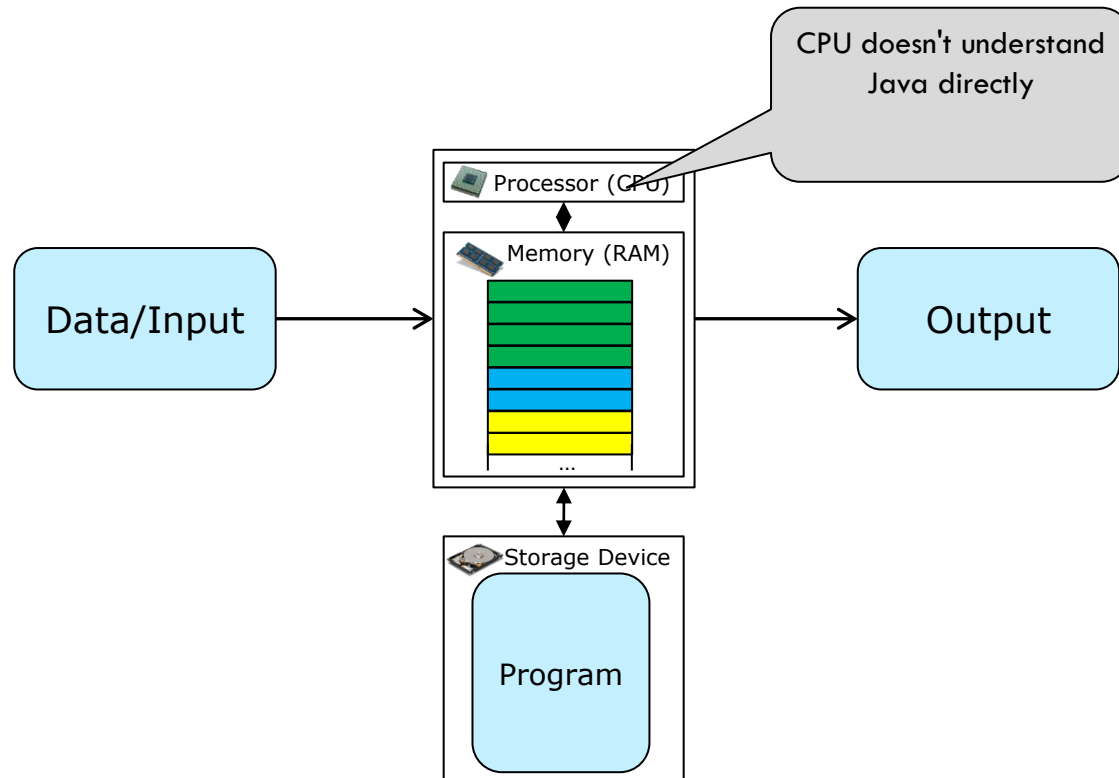
High Level Hardware View



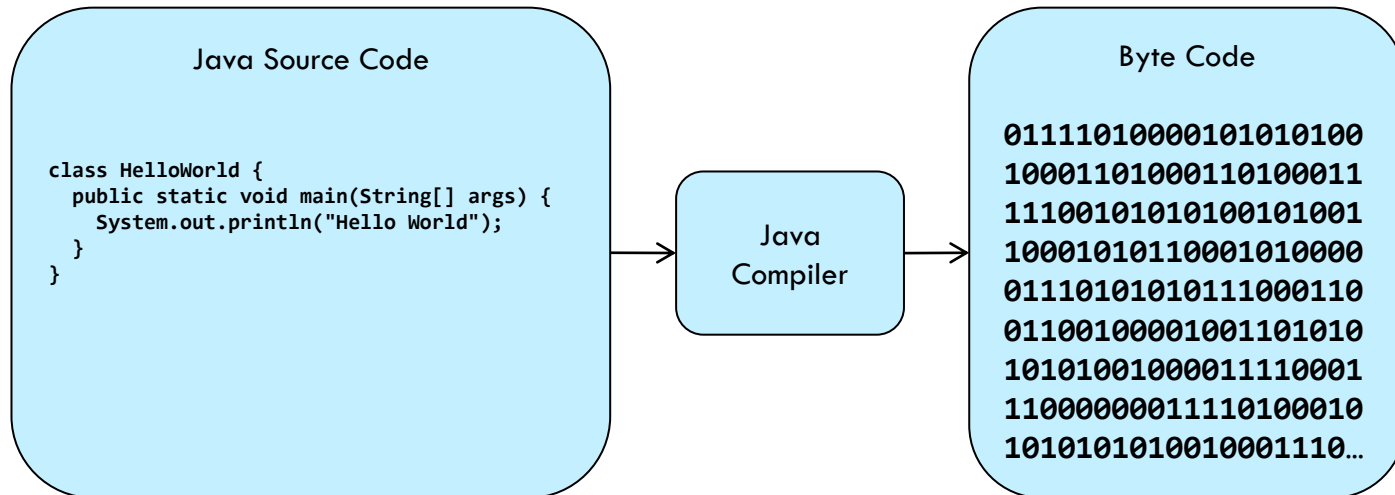
Main Memory (RAM)

Addresses	byte 0	01110011	4 bytes at address 0
	byte 1	01110100	
	byte 2	01100001	
	byte 3	01110010	
	byte 4	01110111	2 bytes at address 4
	byte 5	01100001	
	byte 6	01110010	2 bytes at address 6
	byte 7	01110011	

Running a Program



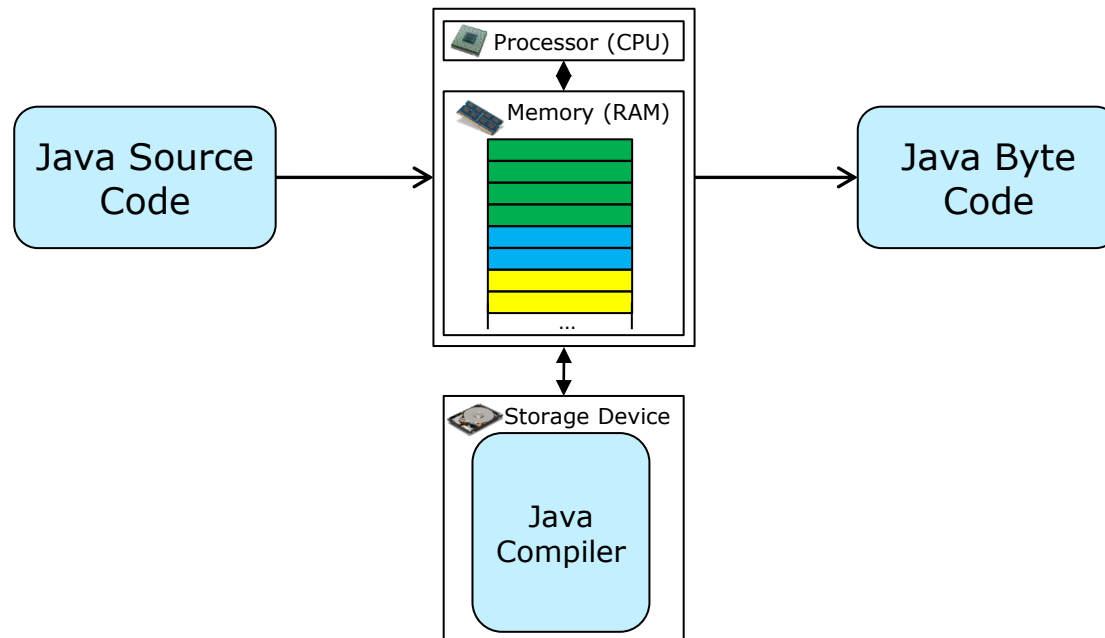
Compilers



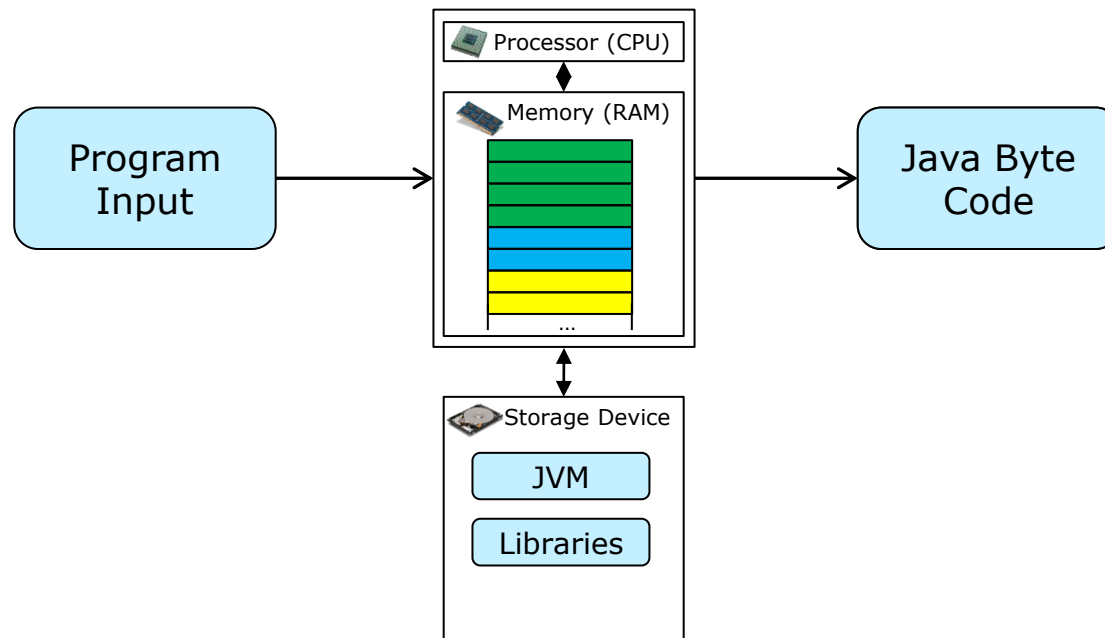
Java Virtual Machine

- Java byte code also can't be executed by a CPU directly
- Instead, the Java Virtual Machine (JVM) is another program that interprets the byte code and translates it into the native CPU language
 - ▣ Allows a program to be compiled once and run on all types of computers (that have a JVM available and installed)
- Other high-level languages work differently

Building a Java Program



Running a Java Program



Introduction

- ❑ Computers can not use human languages, and programming in the binary language of computers is a very difficult, tedious process
- ❑ Therefore, most programs are written using a programming language and are converted to the binary language used by the computer
- ❑ Two major categories of prog languages:
 - ▣ Low level languages
 - ▣ High level languages
- ❑ What is the level about?
- ❑ The level indicates the amount of abstraction between machine language and programming language

Introduction

□ Low level languages:

- Machine language is a set of instruction executed directly by the CPU, and
- Almost or none abstraction.
- More control over the code.
- More code to write.
- The only language computer is cable to understand it.
 - Assembly language
 - Machine code
- Any code written by low level language is executable directly by the hardware without any **compiling** or **linking**.
- Need an assembler to convert the assembly code to machine code.

Introduction

□ Low level languages:

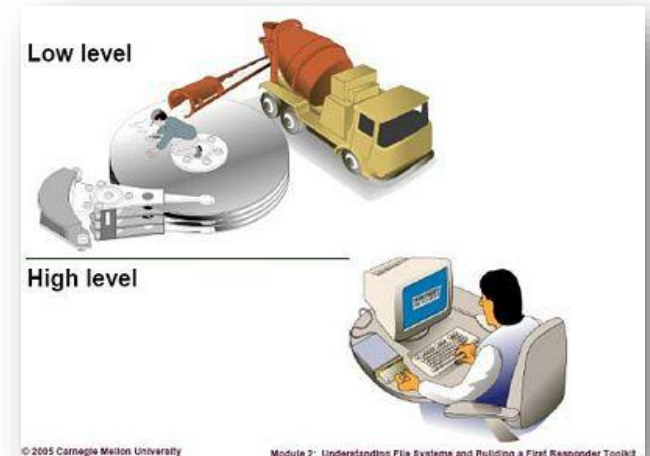
```
global _start ;must be declared for using gcc

_start:      ;tell linker entry point
    mov A,3
    mov B,40
    add A,B
    int 0x80 ;call kernel
```

<https://schweigi.github.io/assembler-simulator/>

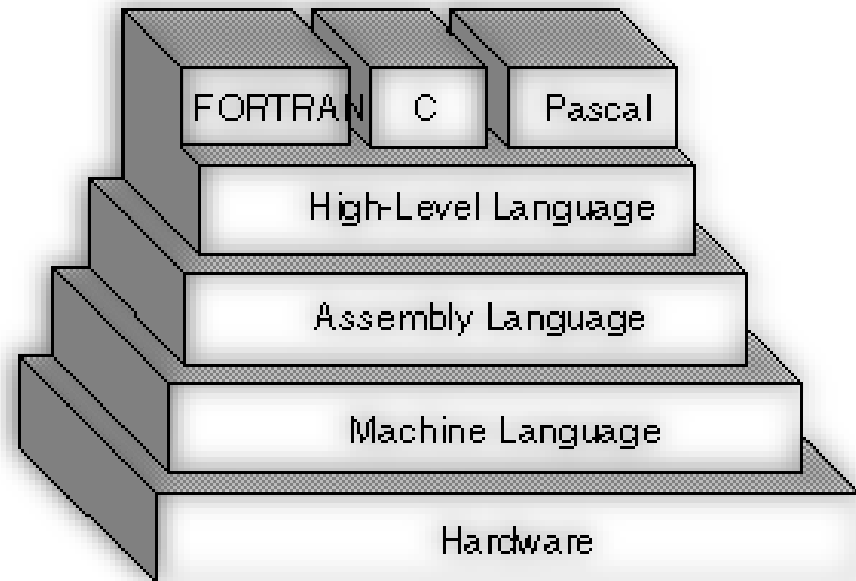
Introduction

- Low Level Languages
 - ▣ Very close to machine language
 - ▣ Concentrate on machine architecture
- High Level Languages
 - ▣ Machine-independent programming language
 - ▣ Concentrate of the logic of problem



Introduction

- Low Level Language
 - ▣ Machine language
 - ▣ Assembly language
- High Level Language
 - ▣ C
 - ▣ C++
 - ▣ BASIC
 - ▣ Java



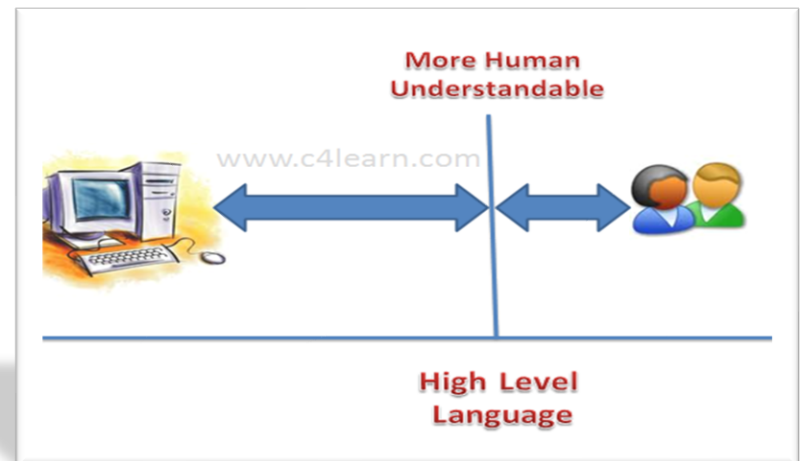
Introduction



- Differences in terms of:
 - i. Understandable
 - ii. Ease of writing
 - iii. Running speed
 - iv. Writing format

Introduction

- Low Level Language:
 - ▣ Mnemonic, binary, hexadecimal
- High Level Language:
 - ▣ Simple English and mathematics symbols



Introduction

Adds two numbers and stores the result

□ Low Level Language:

```
global _start  
_start:  
mov A,5  
mov B,10  
add A,B  
int 0x80
```

■ High Level Language:

```
int main()  
{  
    //assign to the variable A  
    the value of 5 + 10  
    int A = 5 + 10;  
    return 0;  
}
```

Introduction

- **Low Level Language:**

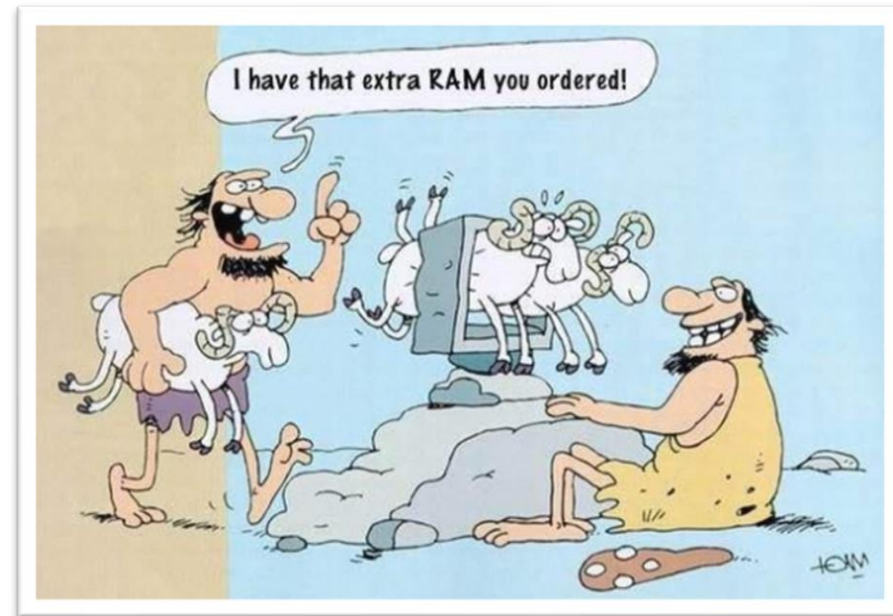
- ▣ Designed for the ease of the computer running the language.
- ▣ Difficult for human to read and write

- **High Level Language:**

- ▣ Designed for the ease of the person writing the language.
- ▣ Using language that human can understand, English

Introduction

- **Low Level Language:**
 - ▣ Faster
 - ▣ No need to compile
 - ▣ More efficient
- **High Level Language:**
 - ▣ Need compiler or interpreter
 - ▣ Translate into machine code
 - ▣ Lower speed execution

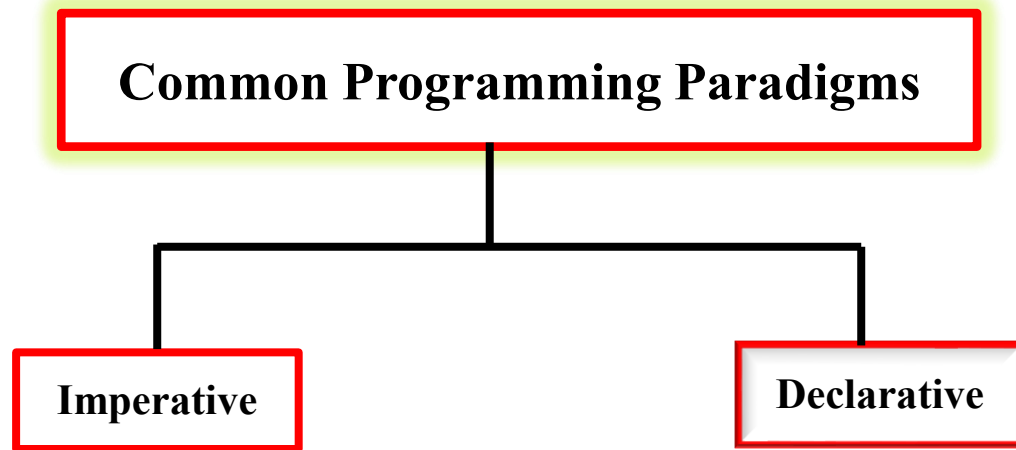


Introduction

- Low Level Language:
 - ▣ Set of instructions for processor
- High Level Language:
 - ▣ Grammar rules



Programming Paradigms



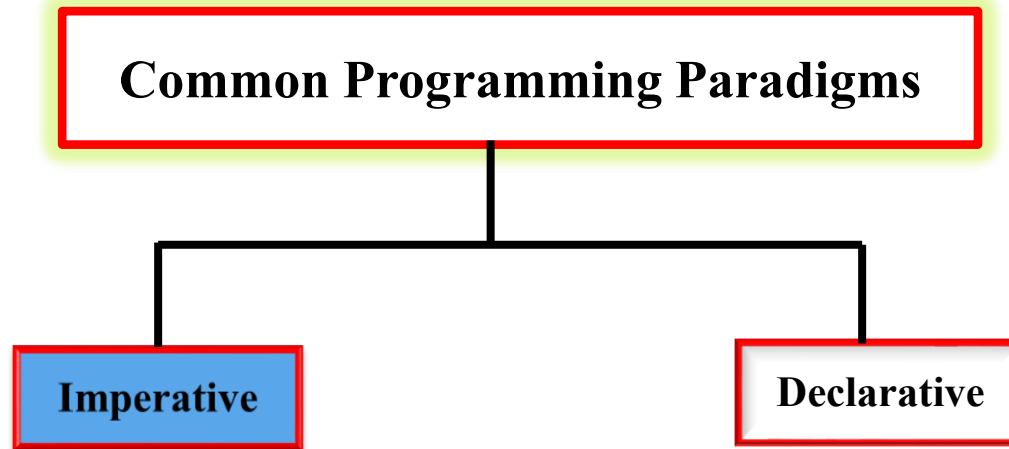
Paradigms are **not** meant to be mutually exclusive; a single program can feature multiple paradigms!

Paradigm: model or mental framework for thinking about something

Programming Paradigms

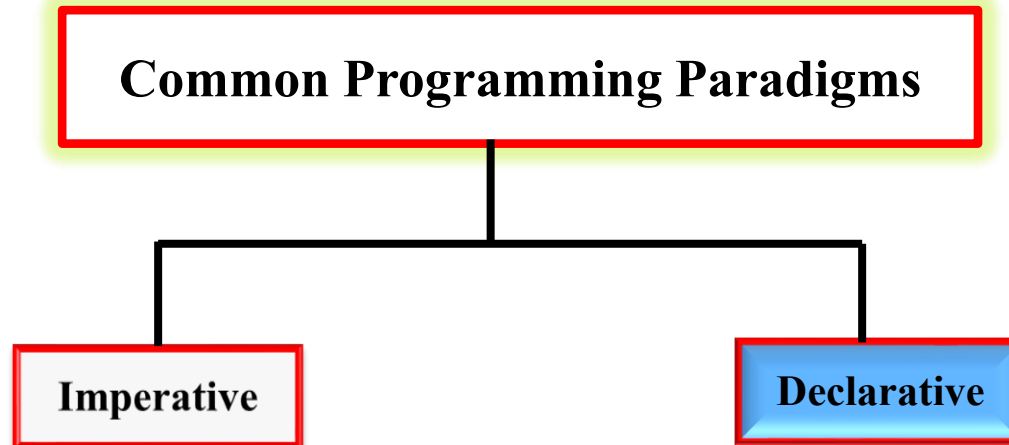
- **Paradigm:** model or mental framework for thinking about something
- Procedural programming paradigm
 - ▣ A program is a sequence of detailed instructions, accessing and modifying memory locations
- Functional programming paradigm
 - ▣ A program is a series of transformations on items

Programming Paradigms



Imperative programming is a programming paradigm that uses statements that change a program's state. In much the same way that the imperative mood in natural languages expresses commands, an imperative program consists of commands for the computer to perform. Imperative programming focuses on **describing how to find a solution.**

Programming Paradigms



Declarative programming is a programming paradigm ... that expresses the **logic** of a computation without describing its control flow.

Imperative programming is a programming paradigm that uses statements that change a program's state.

Why Java?

33

Our Choice: Java

Java features

- Widely used.
- Widely available.
- Continuously under development since early 1990s.
- Embraces full set of modern abstractions.
- Variety of automatic checks for mistakes in programs.



Facts of life

- No language is perfect.
- You need to start with *some* language.

“There are only two kinds of programming languages: those people always [gripe] about and those nobody uses.”



– Bjarne Stroustrup

Our approach

- Use a minimal subset of Java.
- Develop general programming skills that are applicable to many languages.

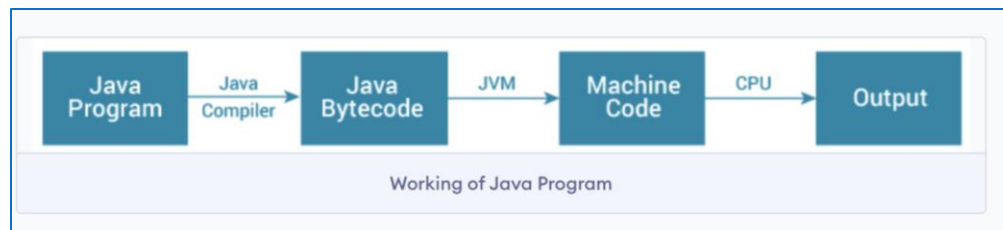
It's not about the language!

Java JDK, JRE and JVM

34

- ❑ **What is JVM?**
- ❑ **JVM (Java Virtual Machine)** is an abstract machine that enables your computer to run a Java program.
- ❑ When you run the Java program, Java compiler first compiles your Java code to bytecode. Then, the JVM translates bytecode into native machine code (set of instructions that a computer's CPU executes directly).
- ❑ Java is a platform-independent language. It's because when you write Java code, it's ultimately written for JVM but not your physical machine (computer). Since JVM executes the Java bytecode which is platform-independent, Java is platform-independent.

❑



What is JRE?

35

- JRE (Java Runtime Environment) is a software package that provides Java class libraries, Java Virtual Machine (JVM), and other components that are required to run Java applications.



What is JDK?

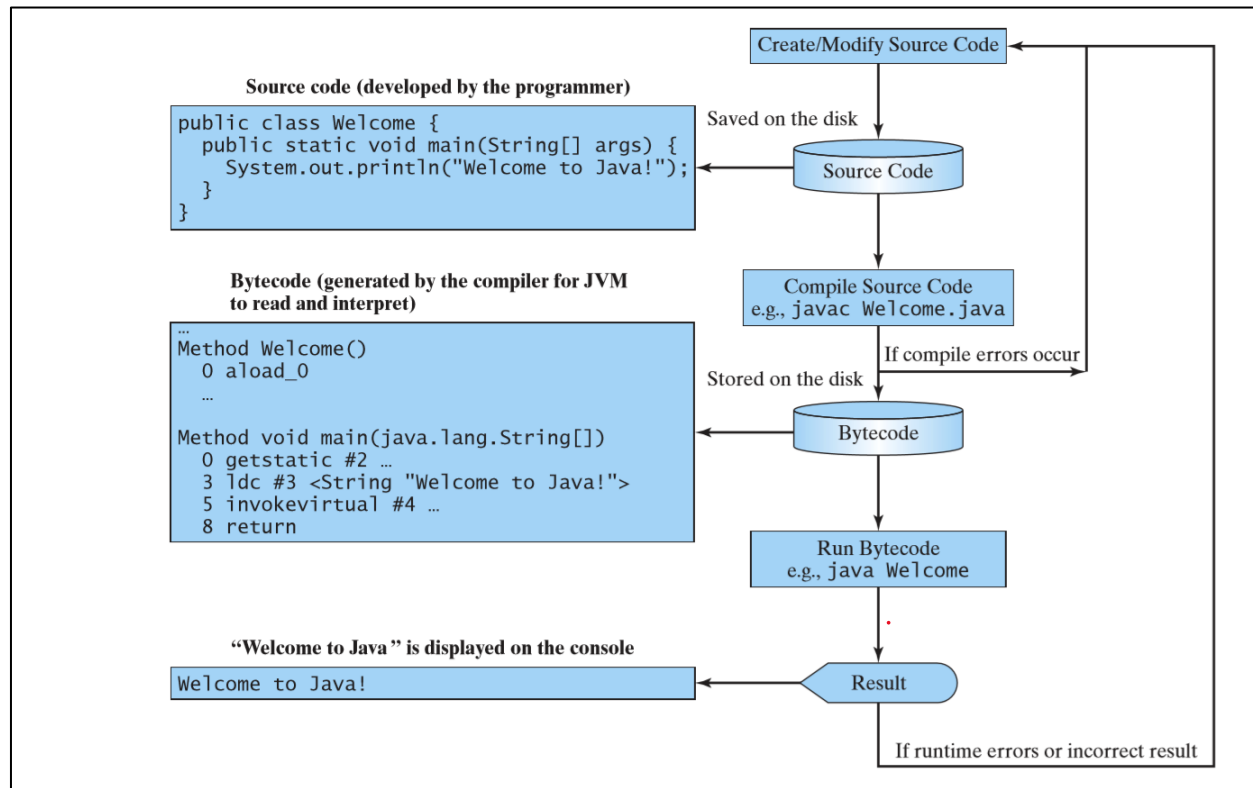
36

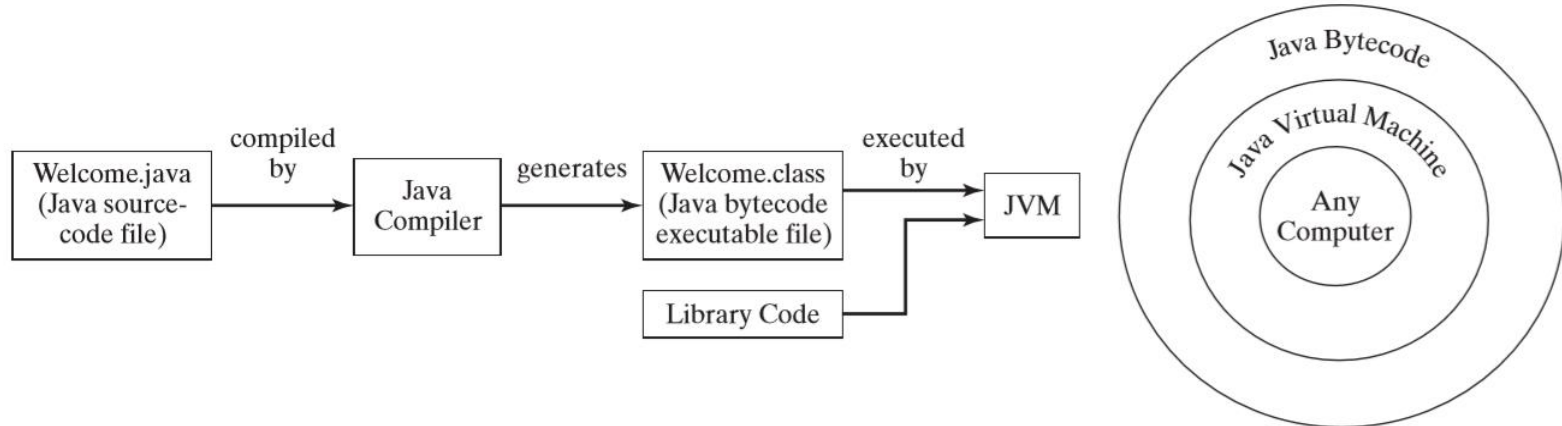
- (Java Development Kit) is a software development kit required to develop applications in Java. When you download JDK, JRE is also downloaded with it.
- In addition to JRE, JDK also contains a number of development tools (compilers, JavaDoc, Java Debugger, etc).



The Java Language Specification JVM, JRE and JDK

37





Take Home Points

- ❑ Computers have 5 main components: Processor, Main Memory, Input Devices, Output Devices, Storage Devices
- ❑ 1 byte = 8 bits (binary digits)
- ❑ Main Memory is a sequence of bytes, each with a memory address
- ❑ The Java compiler turns source code into byte code
- ❑ The JVM uses that byte code along with additional libraries in order to execute your program

Take Home Points

- ❑ Computers have 5 main components: Processor, Main Memory, Input Devices, Output Devices, Storage Devices
- ❑ 1 byte = 8 bits (binary digits)
- ❑ Main Memory is a sequence of bytes, each with a memory address
- ❑ The Java **compiler** turns **source code** into **byte code**
- ❑ The JVM uses that byte code along with additional libraries in order to execute your program





- What is the Java language specification?
- What does JDK stand for?
- What does IDE stand for?

Main Outline

43

Outline

2.1 Introduction

2.2 Your First Program in Java: Printing a Line of Text

2.3 Modifying Your First Java Program

2.4 Displaying Text with `printf`

2.5 Another Application: Adding Integers

2.6 Memory Concepts

2.7 Arithmetic

2.8 Decision Making: Equality and Relational Operators

2.9 Wrap-Up

Summary | Self-Review Exercises | Answers to Self-Review Exercises | Exercises | Making a Difference

References for You

44

- ❑ https://cscircles.cemc.uwaterloo.ca/java_visualize/#mode=display
- ❑ <https://www.hackerrank.com/challenges/java-if-else/problem>
- ❑ <https://docs.oracle.com/javase/1.5.0/docs/api/>
- ❑ https://cscircles.cemc.uwaterloo.ca/java_visualize/
- ❑ <http://bcs.wiley.com/he-bcs/Books?action=index&itemId=0471697044&bcsId=2215>
- ❑ https://www.w3schools.com/java/java_comments.asp
- ❑ <https://javatutoring.com/java-programs/>

Java Origins

- Java was developed by James Gosling at Sun Microsystems in the early 1990s
- It was derived largely from the C++ programming language with several enhancements
- Java is a *high-level* programming language
 - ▣ Provides many useful features that make it easier to write complex code
 - ▣ As opposed to *low level* languages that provide direct access to computer subsystems like memory but require much more care from programmers

Java

- Like most programming languages, Java programs are written using a **fixed syntax**
 - ▣ **Syntax**: a grammar that distinguished well formed statements from those that are not
- Special **keywords** and characters are used to tell the computer how to do what you want it to do
- Java forces the programmer to think as logically as the CPU
 - ▣ Step-by-step, following a strict order

Hello World

```
public class HelloWorld
```

```
{
```

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

```
{
```

```
        System.out.println("Hello World!");
```

```
    }
```

```
}
```

Tells Java that this file contains
executable code

"main" is where you
actually starts execution
now simply use it as
"start here"

Print the line "Hello World" to
the screen

Eclipse: New Project and Source File

- **File** menu -> **New** -> **Java Project** → click 'next'
 - ▣ Enter a **Project name**, for example: Hello World
 - ▣ Click **Finish**
- The project will show up in the left pane (**Package Explorer**)
- **Right click** on the project -> **New** -> **Class**
 - ▣ Enter a **Name**, for example: HelloWorld
 - ▣ Click the button labelled **public static void main(String[] args)**
 - ▣ Click **Finish**

Adding Code

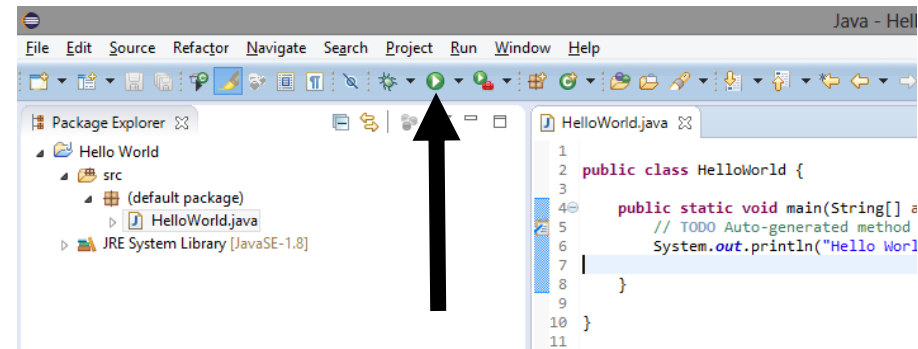
- You can add your program code in between the curly braces after the main line
- For now, add the following line:

```
System.out.println("Hello  
World!");
```

- Note that you can ignore the line that starts with `//`
 - ▣ Lines that start with `//` are *comments* that are ignored by the compiler

Eclipse: Running Your Program

- When you are ready to test your program, you can run it directly in Eclipse
- Click the Run button
 - ▣ Or **Run** menu -> **Run**
 - ▣ Or (Windows) **Ctrl-F11**
 - ▣ Or (Mac) **Shift-Cmd-F11**
- You'll be prompted to save your changes
- Then you'll see your program output in the **Console** pane at the bottom





Tip

An opening brace must be matched by a closing brace. Whenever you type an opening brace, immediately type a closing brace to prevent the missing-brace error. Most Java IDEs automatically insert the closing brace for each opening brace.

match braces



Caution

Java source programs are case sensitive. It would be wrong, for example, to replace `main` in the program with `Main`.

case sensitive

You have seen several special characters (e.g., `{ }`, `//`, `;`) in the program. They are used in almost every program. Table 1.2 summarizes their uses.

special characters

The most common errors you will make as you learn to program will be syntax errors. Like any programming language, Java has its own syntax, and you need to write code that

common errors

<i>Character</i>	<i>Name</i>	<i>Description</i>
{ }	Opening and closing braces	Denote a block to enclose statements.
()	Opening and closing parentheses	Used with methods.
[]	Opening and closing brackets	Denote an array.
//	Double slashes	Precede a comment line.
" "	Opening and closing quotation marks	Enclose a string (i.e., sequence of characters).
;	Semicolon	Mark the end of a statement.

syntax rules

conforms to the *syntax rules*. If your program violates a rule—for example, if the semicolon is missing, a brace is missing, a quotation mark is missing, or a word is misspelled—the Java compiler will report syntax errors. Try to compile the program with these errors and see what the compiler reports.



Note

You are probably wondering why the `main` method is defined this way and why `System.out.println(...)` is used to display a message on the console. For the time being, simply accept that this is how things are done. Your questions will be fully answered in subsequent chapters.

Printing a Line of Text (Cont.)

Declaring a class

□ Class declaration

```
public class HelloWorld
```

- Every Java program consists of at least one class that you define.
- **class keyword** introduces a class declaration and is immediately followed by the **class name**.
- **Keywords** (Appendix C) are reserved for use by Java and are always spelled with all lowercase letters.

Filename for a `public` Class

- A `public` class must be placed in a file that has a filename of the form *ClassName.java*, so class `Welcome1` is stored in the file `Welcome1.java`.

Class Names and Identifiers

- By convention, begin with a capital letter and capitalize the first letter of each word they include (e.g., `SampleClassName`).
- A class name is an **identifier**—a series of characters consisting of letters, digits, underscores (`_`) and dollar signs (`$`) that does not begin with a digit and does not contain spaces.
- Java is **case sensitive**—uppercase and lowercase letters are distinct—so `a1` and `A1` are different (but both valid) identifiers.

Underscore (`_`) in Java 9

- As of Java 9, you can no longer use an underscore (`_`) by itself as an identifier.

Class Body

- A **left brace**, **{**, begins the **body** of every class declaration.
- A corresponding **right brace**, **}**, must end each class declaration.

Printing a Line of Text (Cont.)

Declaring a Method

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

- ❑ Starting point of every Java application.
- ❑ **Parentheses** after the identifier `main` indicate that it's a program building block called a **method**.
- ❑ Java class declarations normally contain one or more methods.
- ❑ `main` must be defined as shown; otherwise, the JVM will not execute the application.
- ❑ Methods perform tasks and can return information when they complete their tasks.
- ❑ Keyword **void** indicates that this method will not return any information.



Good Programming Practice 2.6

Indent the entire body of each method declaration one “level” between the braces that define the method’s body. This emphasizes the method’s structure and makes it easier to read.

□ Body of the method declaration

- ▣ Enclosed in left and right braces.

□ Statement

```
System.out.println("Welcome to Java Programming!");
```

▣ Instructs the computer to perform an action

- Display the characters contained between the double quotation marks.
- ▣ Together, the quotation marks and the characters between them are a **string**—also known as a **character string** or a **string literal**.
- ▣ White-space characters in strings are *not* ignored by the compiler.
- ▣ Strings *cannot* span multiple lines of code.

Printing a Line of Text (Cont.)

- `System.out` object
 - ▣ Standard output object.
 - ▣ Allows a Java application to display information in the `command window` from which it executes.
- `System.out.println` method
 - ▣ Displays (or prints) a line of text in the command window.
 - ▣ The string in the parentheses the `argument` to the method.
 - ▣ Positions the output cursor at the beginning of the next line in the command window.
- Most statements end with a semicolon.

Compiling Your First Java Application

- ❑ Open a command window and change to the directory where the program is stored.
- ❑ Many operating systems use the command `cd` to change directories.
- ❑ To compile the program, type

```
javac Welcome1.java
```
- ❑ If the program contains no compilation errors, preceding command creates a `.class` file (known as the **class file**) containing the platform-independent Java bytecodes that represent the application.
- ❑ When we use the `java` command to execute the application on a given platform, these bytecodes will be translated by the JVM into instructions that are understood by the underlying operating system.



Common Programming Error 2.4

The compiler error message “class `Welcome1` is public, should be declared in a file named `Welcome1.java`” indicates that the filename does not match the name of the `public` class in the file or that you typed the class name incorrectly when compiling the class.



Error-Prevention Tip 2.2

When the compiler reports a syntax error, it may not be on the line that the error message indicates. First, check the line for which the error was reported. If you don't find an error on that line, check several preceding lines.

Executing the HelloWorld Application


- To execute this program in a command window, change to the directory containing HelloWorld.java
 - ▣ C:\examples\HelloWorld.java on Microsoft Windows or
 - ▣ ~/Documents/ on Linux/macOS.
- Next, type HelloWorld.java.
- This launches the JVM, which loads the HelloWorld.class file.
- The command *omits* the .class file-name extension; otherwise, the JVM will *not* execute the program.
- The JVM calls class HelloWorld's main method.



Error-Prevention Tip 2.3

When attempting to run a Java program, if you receive a message such as “Exception in thread “main” java.lang.NoClassDefFoundError: Welcome1,” your CLASSPATH environment variable has not been set properly. Please carefully review the installation instructions in the Before You Begin section of this book. On some systems, you may need to reboot your computer or open a new command window after configuring the CLASSPATH.

```
C:\Users\fatem\Desktop\Java_Exa>javac HelloWorld.java
```



```
C:\Users\fatem\Desktop\Java_Exa>java HelloWorld  
Hello World
```

