Lecture 2: Intro to Java

By Ahmed E. Khaled
Recap to previous lecture!

- What is a Computer, and Machine Language?
- What is a Programming language, and why we need it?
Lecture 02, intro to java

• Introducing important hardware parts of a computer.
• Bits and Bytes.
• History of Java, and how program is compiled and executed.
• Structure of Java program, main function and printing messages.
• Types of errors.
Good news, if you learn to program using one language, you will find it very easy to work with other languages.

The key is to learn how to solve problems using a programming approach.

This class is about the software part of the computer (programming), But it is good to get the basic idea of computer hardware.
Central Processing Unit - CPU

CPU is the computer’s *brain* where processing and executions take place.

CPU usually has two components:

- **Control unit**: extracts instructions from memory, decodes and executes them.

- **Arithmetic and logic unit**: performs numeric operations (addition, subtraction, multiplication, division) and logical operations (comparisons).
Speed of CPU is defined in a measurement unit named hertz (Hz) – clock speed, the higher the more instructions can be executed in second. Intel’s newest processors run at about 3 GHz.

CPUs were developed with one core. The core is the part of the processor that performs the reading and executing of instructions. In order to increase CPU processing power, chip manufacturers are now producing CPUs that contain multiple cores.
Bits and Bytes!

Computer is nothing more than a set of switches, either on / off.

These 0’s and 1’s are interpreted as digits in the binary number system and are called bits.

Eight bits form Byte, which is the minimum storage unit in a computer.

Data of various kinds (numbers and characters), are encoded as a series of bytes. E.g. character C is represented as 01000011 in one byte.
A computer’s storage capacity is measured in multiples of the byte:

- Kilobyte (KB) is about 1,000 bytes.
- Megabyte (MB) is about 1 million bytes.
- Gigabyte (GB) is about 1 billion bytes.
- Terabyte (TB) is about 1 trillion bytes.
Ordered sequence of bytes for storing programs as well as data that the program is working with, think of it as the computer’s work area for executing a program.

Every byte in the memory has a unique address to locate the byte for storing and retrieving the data. Since the bytes in the memory can be accessed in any order, the memory is also referred to as random-access memory (RAM).

RAM is a volatile, any information that has been stored in memory is lost when the system’s power is turned off.

Programs storage devices and data are permanently stored on storage devices (hard disk, USB, CD and DVD).
Diagram showing the interaction between CPU and memory. The CPU has a program counter set to 1011100001. Data is transferred to memory at locations 0, 1, 2, 3, and 10. The memory locations contain binary data such as 00101110, 11010011, 01010011, etc.
High-level languages (HLL)

HLL are platform independent where you can write a program once and run it in different types of machines.

A program written in a high-level language is called a *source program*, that must be translated into machine code for execution. The translation can be done using another programming tool called an *interpreter* or a *compiler*.
• *Interpreter* reads one statement from the source program at a time, translates it to the machine code, and then executes it right away.

• *Compiler* translates the entire source program into a machine-code, and the machine-code file is then executed.
Java is using a mixed form between compiler and interpreter, that is not dedicated to specific machine.

Java programs are compiled into machine language, but it is for a “virtual” machine known as the Java Virtual Machine (JVM).

The machine language for the JVM is called *Java bytecode*, thus the same compiled program can be run on many different *types* of computers.
Why, they are using Java bytecode?

• The compiler is a complex program that has to understand Java, making compiler to **every new type of computer is a hard work**.

• On the other hand, **Java bytecode interpreter** is a fairly small and simple program and easy to write one for new type of computers to run any compiled Java program.
Recap!

Bits vs. bytes?

What is a compiler?

What is an interpreter?

What is the execution process?
Java is a powerful and general-purpose programming language for developing software running on mobile devices, desktop computers, and servers.

It is employed not only for Web programming but also for developing standalone desktop applications.

Java was developed by a team at Sun Microsystems. Java was designed in 1991 for use in embedded chips in consumer electronic appliances and in 1995 was redesigned for developing Web applications.
Important keywords

• The Java Development Kit (JDK) is the software for developing and running Java programs through terminal.

• An Integrated development environment (IDE) is a development platform with powerful graphical user interface for Editing, compiling, building and debugging your program (e.g., NetBeans and Eclipse).

• The application program interface (API), also known as Java libraries, contains predefined functions and services for developing Java programs.
Ok, let's start Java programming!

Let's begin with a simple Java program that displays the message “Welcome to Java!” on the console (What is that?).
Console is an old computer term that refers to *text-only user interface* (e.g. terminal and command line interface of some operating systems). IDEs have their own consoles.

Console input means to receive input from the *keyboard*, and console output means to display output on the *monitor.*
Console vs. GUI.

More advanced step is to build a **Graphical User Interface** (GUI), type of user interface that allows users to interact with your program through graphical icons and visual indicators.
Welcome to Java, Console application.

```java
package welcomejava;

public class WelcomeJava {

    public static void main(String[] args) {
        // TODO code application logic here
        System.out.println("Welcome to Java !");
    }
}
```
Structure java program.

When you create a java program, the IDE automatically creates a class for you that contains the main method.

Think of class as a structure for now, the class name is WelcomeJava (Line 3).

The program is executed from the main method, that is the entry point where the program begins execution (Line 5).

```
public class Welcome {
    public static void main(String[] args) {
        System.out.println("Welcome to Java!");
    }
}
```

A package is a way to group java classes.
• The main method in this program contains the System.out.println statement (Java API method to print *String* on console).

• String is a sequence of *characters*, enclosed in double quotation marks (e.g. “java”).

• Every statement in Java ends with a *semicolon* (;), known as the statement terminator.
Line 6 is a comment that documents what the program is and how it is constructed (help programmers to understand the program).

Comments are not programming statements and thus are ignored by the compiler.

In Java, comments are:

- Preceded by two slashes (//) on a line, called a **line comment**.
- Enclosed between /* and */ on one or several lines, called a **block comment**.

```java
// This is a line comment
/* This is a block comment */
```
Try to guess the output!

Program 1:
```java
public class WelcomeJava {
    public static void main(String[] args) {
        // TODO code application logic here
        System.out.println(2 * 5);
    }
}
```

Program 2:
```java
public class WelcomeJava {
    public static void main(String[] args) {
        // TODO code application logic here
        System.out.println((2 + 3) * 2);
    }
}
```
Try to guess the output!

Program 3:
```java
public class WelcomeJava {
    public static void main(String[] args) {
        System.out.println( "Welcome to Java!" );
        System.out.println( "This is a hello program!" );
    }
}
```

Program 4:
```java
public class WelcomeJava {
    public static void main(String[] args) {
        System.out.println( "what is the result of: " );
        System.out.println( 5 + 3 );
    }
}
```
Try to guess the output!

Program 5:

```java
public class WelcomeJava {
    public static void main(String[] args) {
        // TODO code application logic here
        System.out.println( "what is the result of : " );
        System.out.println( " 5 + 3 ");
    }
}
```
Try to guess the output!

Program 6:

```java
public class WelcomeJava {
    public static void main(String[] args) {
        // TODO code application logic here
        System.out.println( "what is the result of : " );
        //System.out.println( " 5 + 3 ");
    }
}
```
• Java is *case-sensitive*.

• Reserved words (*keywords*) have a specific meaning to the compiler and cannot be used for other purposes in the program. (e.g. when the compiler sees the word `class`, it understands that the word after `class` is the name for the class).

• Whitespace makes things easier to read, and ignored by compiler.
Programming errors!

Errors in programming can be categorized into three types: syntax errors, runtime errors, and logic errors.
Try to guess the error!

```java
public class WelcomeJava {
    public static void main(String[] args) {
        // TODO code application logic here
        System.out.println("welcome to java!");
    }
}
```
Syntax error!

Syntax errors result from errors in *code construction*, such as mistyping a keyword, omitting some necessary punctuation, or using an opening brace without a corresponding closing brace.

These errors are *easy* to detect because the compiler tells you where they are and what caused them.
Try to guess the error!

```java
public class WelcomeJava {
    public static void main(String[] args) {
        // TODO code application logic here
        System.out.println("try out this division: ");
        System.out.println(1 / 0);
    }
}
```
Runtime error!

Errors cause a program to *terminate abnormally (crash)*, occur while a program is *running* and the environment detects an operation that is *impossible to carry out*.

Input mistakes typically cause runtime errors, the program is asking the user’s name, but the user enters his phone number! (data-type errors)

Another example of runtime errors is *division by zero*, when the divisor is zero for integer divisions.
Try to guess the error!

```java
public class WelcomeJava {

    public static void main(String[] args) {
        // TODO code application logic here
        System.out.println( "try out this division: ");
        System.out.println( 2 + 10 / 2 );
    }
}
```
Logic error!

Logic errors occur when a program *does not perform the way it was intended to*.

This kind of errors is also known as “bugs”, where they drive the application to operate incorrectly (undesired outputs or behavior), and not to terminate abnormally (or crash).

Finding logic errors, on the other hand, can be very challenging.
In summary

In the Java programming language:

- Java program is made up of one or more *classes*
- One Main Method in the entire java program, your entry point.
- Different comments to describe the different parts of your program.

Different types of errors arise.

Each keyword will be explored in more detail throughout the course.
Try this out!

Write a program that displays the following table:

<table>
<thead>
<tr>
<th>a</th>
<th>a^2</th>
<th>a^3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>27</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>64</td>
</tr>
</tbody>
</table>

Write a program that displays the following pattern:

Write a program that displays the result of: \( 9.5 \times 4 - 2.5 \times 3 \div 45.5 - 3.5 \).