# INTRODUCTION TO JAVA

## Course outline

1. **Programming in Java – 34 Lessons (8 Weeks)**

* **Java Basics**
* What is Java
* JVM
* Installing JDK Components
* The Java Editor (Netbeans, Visual Studio Code)
* First Java Program
* **Java Basic Language Constructs**
* Java Integer Variables
* Double Variables
* Short and Float
* Operator\_Precedence
* **Java String**
* Java String
* Upper and Lower Case
* The compare Method
* The contact Method
* The IndexOf Method
* The substituting Method
* The equals Method
* The charAt Method
* The replace Method
* Printing Formatted Strings
* **Control Flow**
* If Statements on Java
* If Else Statements
* Boolean Values
* Switch Statements
* Java for Loops
* Java While Loops
* **Arrays in Java**
* Java Arrays
* Arrays and Loops
* Sorting Arrays
* Arrays and Strings
* Multi-Dimensional Arrays
* Array List
* **Java Methods**

**-**Java Methods

**-** Calling your Java Methods

**-** Passing Values to Methods

**-** Multiple Method Parameters

* **Java Object Oriented Concepts**

-Oops Concepts in Java

-Object and Classes

-Methods

-Inheritance

Polymorphism

-Abstraction in OOPs

-Encapsulation in Java OOPs

* **Java Classes**
  + Writing your own Java Classes
  + Field Variables
  + The Java Class Constructor
  + Accessing Class Variables
  + More on Java Class Methods
  + Java and Inheritance
  + Java Enumerations (Intermediate Lesson)
  + A Random Number generator program (Intermediate Lesson)
* **Java Error Handling**
* Error Handling and Exceptions
* Stack Trace
* Handling Logic Errors in Java
* **Java Calculator Project**
* **Java Forms Controls**
  + Java and Combo Boxes
  + Java Check Boxes
  + Radio Buttons
  + Adding Menus to a Java Form
  + Open File Dialogue Boxes
  + Opening Files
  + Save File Dialogue Boxes
* **Java and Databases**
  + Java and Databases
  + Creating a Database with Java
  + Adding Records to a Table
  + SQL Commands
  + Connect to a Database with Java code
  + Connect to the Table
* **Databases and Java Forms**
  + Databases and Java forms
  + Databases Scrolling Buttons
  + Move back Through the Database
  + Move to the First and Last Records
  + Update a Record
  + Add a New Record
  + Save a New Record
  + Delete a Record from a Database
* **Java Memory Management**
  + Garbage Collection in Java
  + Java Static Variables
  + Memory Allocation in Java
* **Java Project**

## INTRO TO JAVA

Java is a high level, class-based, object-oriented and very secure programming language.

It is class-based since everything runs in class(s). It is object-based since everything runs on an object.

It is a platform **independent** since it requires its own compilation irrespective of the physical platform running on it.

## Applications of Java

They are many devices where java is currently used to day.

Example

* Desktop application.
* Mobile application.
* Web-based applications
* Enterprise applications
* Embedded systems.
* Smartcards
* Mobile games
* Robots

### Types of JAVA application

They are 4 types:

1. Standalone – they are also known as window-based application.
2. Web-based application – they are application that runs on a server side and displays the output on a dynamic page.
3. Enterprise applications – this are application that are distributed in nature, have high level security and are used in high-end applications such as banking.
4. Mobile application – This are application made for mobiles. E.g. Android.

## JAVA building blocks

The building blocks of JAVA consist of:

1. Editing platform – VS code, net beans, eclipse, dream weaver.
2. Runtime environment – JDK, extensions.
3. Output

For the runtime environment they are several components that are inseparable to each other, they include:

* **JVM -** it is the acronym for Java Virtual Machine it is an abstract machine that enables your computer to run a Java Program, it contains the Java compiler that compile (Javac) a Java code to Byte code. **JVM** translate the bytecode into native machine code.
* **JRE** – it is the acronym for Java runtime environment, it’s a software package that provides Java libraries, JVM and other components required to run a Java application. It is a super set of **JVM**.
* **JDK -** it is the acronym for Java Development Kit, it is a software tool kit that is require to develop applications in Java, it contains the JRE in it. It also contains a number of development tools. i.e., Javac, Javadoc, Java debugger.

When compiling Java code, **JVM** converts the source code into byte code, then the Java compiler (Javac) converts the byte code into machine language where the CPU process the output and reconverts it back to a language readable to humans.

## Creating the first program

The following are **essential** point to note in Java programming:

* ***Every application in JAVA begins with a class definition, since JAVA is a class-based programming language.***
* ***The name of the class MUST be the same as the file name in Java.***
* *Every block of code is sandwiched btwn a pair of calibrates.*
* *Java Statements are terminated using semi-colon (;).*

### Parameters used in the first Java program

public class java1 {

    public static void main(String args []){

        System.out.println("Hello programmer, welcome to your first Java Program!");

    }

}

Class – **class** keyword is used to declare a class in Java.

Public – **public** keyword is an access modifier that represent visibility. It means it is visible to all.

Static – **static** keyword, if you declare any method as static, it is known as the static method. A static method in Java does not require creation of an object to invoke it. The main () is executed by the **JVM** and therefore does not require creating an object to invoke it.

Void – **void** is a return type of the method. It means it does not return any value.

Main – **main** is the most important. It represents the starting point of the program.

String args [] – **String args []** it is used for command line argument; the arguments are streamed as string literals.

**System.out.print or System.out.println** – is used to print statement in JAVA. Here, **system** is a class, **out** is an object of printStream class, **print or println** is a method of printStream class.

## JAVA Variables.

A variable is a container that holds data of a particular program. It can also be defined as the name of the memory location that is equivalent to its data type.

They are 3 types of variables in JAVA:

1. Local variable
2. Instance variable
3. Static variable

### Local variable

It is a variable that is declared inside of the body. It can only be accessed with in that method.

N/B: A local variable cannot be defined as static.

Example:

public void myage(){

        int age = 1;

        age += 22;

        System.out.println("Hey there! My age is " + age);

    }

### Instance variable

An instance variable is declared in a class but outside a method, a constructer or any block of code. They are created when an object is created by the keyword new and destroyed when the object is destroyed.

They are declared in a class level before and after use. Instance variables are visible to all methods, constructors and block of code in a class.

Example:

public class instance\_var {

    String country;

    String county;

    public void mycounty() {

        county = "Meru";

        System.out.println("Welcome to my county " + county);

    }

    public void mycountry() {

        country = "Kenya";

        System.out.println("And " + county + ", is in " + country + " which is a country in East Africa.");

    }

    public static void main(String[] args) {

        instance\_var iv = new instance\_var();

        iv.mycounty();

        iv.mycountry();

    }

}

### Static variable

A static variable is a variable that belongs to the class its self and not to any other instance of the class. It creates only one copy of the variable in the memory regardless of how many times is used.

N/B: Static variables are created at a class level only; a static variable is declared using a keyword ***STATIC***.

## Data types in JAVA

Data types are used to specify different sizes and values that can be stored in a variable.

They are 2 main types of data types in JAVA:

1. Primitive data types
2. Non-primitive data types

JAVA is a Statically typed language, i.e., you must declare the data type when defining the variable.

### Primitive data types

Primitive data types are only single values that have no special capabilities. Java define 8 types of primitive data categorized into 4 groups;

1. Integers – this group include
   1. Byte
   2. Short
   3. int
   4. long.
2. Floating-point numbers – this group represents decimal numbers.
   1. Float
   2. double
3. Character – this group represents symbols in character set.
4. Boolean – this group represent Boolean.

### Non primitive

A non-primitive data type contains data such as stings, array structure or a class.

## Operators in JAVA //assignment

Operator is a symbol to be used

They are several types of operators in JAVA:

* Unary operators
* Arithmetic operators
* Relational operators
* Shift operators
* Bitwise operators
* Logical operators
* Ternary operators

### JAVA keywords. //assignment

Also known as reserve words.

They are particular words that act as a key to a code.

## Control flow statements in JAVA

Control statements are used to control the flow of Java code.

Javac execute the code from top to bottom in the order they appear, java provide 3 types of control flow statements:

1. Decision making statement – they are statement they decide which block of code to execute and when. The evaluate Boolean expression. They are 2 types:
   1. If statement.
   2. Switch
2. Loop statement – loop statement is used to execute a set of instructions in a repeated order. The executed depends on a particular condition. They are 3 types of loops in java:
   1. For loop
   2. While loop
   3. Do-while loop
3. Jump statement – jump statement are used to transfer the control of the program to a specific statement, JAVA supports 2 types of jump statement:
   1. Break
   2. Continue

## If statement

The if statement is used to evaluate a condition, the condition of if statement evaluates a Boolean value, a true or false. They are 4 types of if statement in JAVA:

1. If statement
2. If-else-statement
3. If-else-if statement
4. Nested if statement

### Simple if statement

Most basic statement it only executes block of code if the condition is true.

public class simple\_if{

    public static void main(String[] args){

        int age = 41;

        if (age >= 18) {

            System.out.println("You are eligible to vote");

        }

    }

}

### If-else statement

It is used to execute the block of code if the condition is true and only give an alternative on the else block if the condition of the if block is false.

public class if\_else {

    public static void main(String[] args) {

        int no = -6;

        if (no >= 0) {

            System.out.println("Thats a positive number");

        } else {

            System.out.println("Thats a negative number.");

        }

    }

}

### If-else-if statement

It is used to execute a block of code on multiple condition and only execute the condition that turns true. It is also known as if-else-if ladder, else is used to give an alternative when all the if statements turn false.

public class if\_else\_if {

    public static void main(String[] args) {

        System.out.println("Welcome to Movie max");

        String movies = "The game";

        if (movies == "Star wars") {

            System.out.println("Star wars is one of the best fantasy movies which began in the 1990's.");

        } else if (movies == "Game of Thrones") {

            System.out.println("Game of Thrones is one of the best fantasy series which began in 2011.");

        } else if (movies == "Fast & Furious") {

            System.out.println("Fast & Furious is one of the best action movies which began in the 2000's");

        } else {

            System.out.println("List your favorite movies down below to list it in the above categories.");

        }

    }

}

**N/B: JAVA provides a special operator which is called a ternary operator which simplifies the if-else statement. A ternary operator evaluates the test condition and executes a block of code, based on the result of the condition. It takes 3 operands (condition? expression1: expression2).**

import java.util.Scanner;

public class ternary\_if {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.println("Welcome to Fun land.");

        System.out.println("Guess a number from 0 to 9 in the game to win.");

        int no = sc.nextInt();

        // ternary operator to be used below.

        String result = (no == 7)? "win": "lose";

        System.out.println("You have " + result + " the game");

        sc.close();

    }

}

### Nested if statement

It is a statement that contains an if or else if statement inside another if or else if.

Example:

import java.util.\*;

public class nested\_if {

    public static void main(String[] args) {

        double num1 = 15.23, num2 = 60, num3 = -123.9, largest;

        // check if num1 is >= num2

        if (num1 >= num2) {

            // and if num1 is >= num3

            if (num1 >= num3) {

                largest = num1;

            } else {

                largest = num3;

            }

        } else {

            if (num2 >= num3) {

                largest = num2;

            } else {

                largest = num3;

            }

        }

        System.out.println(largest + " is the largest of them all.");

    }

}

## Loop statements

A loop statement is used to execute set of instruction in a repeated order.

The execute of the set of instruction depends upon the set condition.

Java supports 3 types of loops:

1. For loop
2. While loop
3. Do While loop

### For loop

It is used to run a block of code for a certain number of times. It is used when the number of alliterations is known.

Syntax: For (initial expression; condition expression; update expression) {

*//block of code;*

}

### Nested for loop

It is a for loop inside another for loop.

The inner for loop executes the on all possible alliterations for every single outer loop.

### While loop

A while loop is used to run a specific code until a certain condition is met. It is commonly used when the number of alliterations is not known.

Syntax:

While(condition) {

// block of code.

}

Example:

while (no >= 0) {

            // add positive numbers only.

            sum += no;

            System.out.println("Enter another number ");

            no = sc.nextInt();

        }

### Do while loop

It is used to alliterate part of the program repeatedly until a special condition turns false. Similar to php, it is known as an exit control loop. Therefore, the body of the do while loop is executed at least once.

Syntax:

Do {

// executes the code

}

While (condition)

### For each loop

Used to aliterate elements of arrays and collections. Also known as the enhanced for loop.

Syntax:

For(new datatype : old variable)

JAVA Object Oriented Concepts

Java is an object-oriented programming language. Its main aim is to create software’s that will implements real world entities. The basic meaning of object-oriented programming is as follow.

1. Object – it means a real word entity or anything that has got status or properties.

2. Object-oriented programming – its is a methodology used to design a program using classes and object. It has several concepts such as.

I. Object

II. Class

III. Inheritance

IV. Polymorphism

V. Abstraction

VI. Encapsulation

Object

It is any entity that has state and behavior. It can also be defined as an instance of a class. It contains an address and takes up some space in memory. It is a physical entity.

Class

Collection of objects is called class. A class is a blueprint from which you create an individual object.

It is a logical entity. Class in Java contains:

I. Fields

II. Methods

III. Constructors

IV. Block

V. Nested classes and interfaces

Fields

A field contains variables. They are several examples of variables in JAVA depending on the point of definitions.

Instance variables – it’s created inside a class. But outside the method. Instance variables do not get memory at compile time. However, a memory is allocated at run time when an object is allocated to it.

Main outside the class – java support multiple classes in a single however, if you define multiple classes ensure you save the file name with the class name which has the main method.

Initializing an object, they are 3 ways:

1. By reference variable

2. By Method

3. By Constructor

By reference variable.

Means storing data in to the object. That an instance variable can be initialized through the object created as a pointer.

N/B: you can also create multiple object and store information in it through reference variables.

JAVA CONSTRUCTORS

* Is a block of code similar to the method.
* It is called when a instance of a class is created therefore , a constructor is a special type of method used to initialize the objects.
* Every time an object is created using key word new at least one constructor is called.

# Rules for creating java constructors

* The constructor name must be the same as it class name.
* A constructor must have no a return type.
* Java constructor cannot be abstract, static, final or cyclonite.

Examples:

public class konstructor {

   private String dn;

   //create a constructor

   konstructor(){

    System.out.println("......calling the constructor.....");

    dn="Cathy Lynn Ejikon";

   }

   public static void main(String[] args) {

    konstructor kn =new konstructor();

    System.out.println("The deputy President at Kenswed TVTC is:"+kn.dn);

   }

}

# TYPES OF CONSTRUCTORS

There are three types of java

1. No-arg constructors
2. Parameterized
3. Defaults

# NO-ARG CONSTRUCTORS

* the constructor which take no argument is called the no arg constructor.
* It is automatically invokes when the object is created.
* If there are no constructors in java class, the java compiler provides a default no-arg constructor.

Example:

public class student {

   private String name,reg\_no;

   private int age;

   student(){

    name="Dennis Njuguna";

    reg\_no="ICP/435/2023";

    age=21;

   }

   public void sayhello(){

    System.out.println("Hello guys my name is:"+" "+name+"whose registration number is:"+" "+reg\_no+"Iam"+" "+age+"years old!!!");

   }

   public static void main(String[] args) {

    student st =new student();

    st.sayhello();

   }

}

public class mathafu {

    private double radius, area;

    mathafu() {

        radius = 14;

        area = 3.14 \* radius \* radius;

    }

    void calculateArea() {

        System.out.println("The area of the circle with radius " +radius+" is  " +area);

    }

    public static void main(String[] args) {

        mathafu mf = new mathafu();

        mf.calculateArea();

    }

}

# PARAMETERLISED

* It is a constructor that initialize the object with specific value.
* It allows the developers to provide more flexibility when constructing objects.

Example:

public class student1 {

   private String name,regno;

   private int age;

   public student1(String nam,String reg\_no,int ag){

    name=nam;

    regno=reg\_no;

    age=ag;

   }

   public void sayhello(){

    System.out.println("Hello guys my name is:"+" "+name+"whose registration number is:"+" "  +regno+"Iam"+" "+age+"years old!!!");

   }

   public static void main(String[] args) {

    student1 st1 =new  student1("Serrics", "music", 17);

    st1.sayhello();

   }

}

JAVA METHOD

* A java method is a collection of statement that are grouped together to perform an operation.
* When you call the system.out.println method , the system actually executes several statement in order to display in the console

#### Creating a method