INTRODUCTION TO

What is Java?

- Java is an object-Oriented programming language.
- Java programming language was originally developed by Sun Microsystems which was initiated by James Gosling and released in 1995 as core component of Sun Microsystems' Java platform (Java 1.0 [J2SE]).
- Java was released as a language that guarantees the “write once run anywhere” (WORA) capability.
  - This means that the software created using Java can run in any device despite hardware and software architecture.
- In 2006 sun microsystems released java as a free and open source software under the terms of the General public use license.
- Java was later acquired by Oracle and is currently maintained by Oracle.

Java Configurations

- With the advancement of Java and its widespread popularity, multiple configurations were built to suit various types of platforms.
- They include:
  - **Java EE**(Java Enterprise Edition)- used to develop enterprise applications such as servlets, java server pages, among other online software .
  - **Java ME** (Java Micro Edition)- used for mobile application development
  - **Java SE** (Java standard Edition)- used for development of desktop applications.
- Java SE is the most used JAVA variant. JAVA SE 17.0.1 is the latest version of the SE.
- Our aim here is to develop desktop applications thus we will focus purely on the JAVA standard Edition

Features of Java

- **Object Oriented:** In Java, everything is an Object and such an object must belong to a class. Java can be easily extended since it is based on the Object model.
- **Platform Independent:** Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into platform specific machine, rather into platform independent byte code.
  - This byte code is distributed over the web and interpreted by the **Java Virtual Machine (JVM)** on whichever platform it is being run on. JVM is discuss later in this lesson.
  - This makes it possible for Java to run any machine regardless of the software and hardware architecture
- **Simple:** Java is designed to be easy to learn. If you understand the basic concept of OOP, Java would be easy to master.
- **Secure:** With Java's security features it is possible to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.

Cont.

- **Architecture-neutral:** Java compiler generates an architecture-neutral object file format, which makes the compiled code executable on many processors, with the presence of Java runtime system.
- **Portable:** Being architecture-neutral and being implementation neutral makes Java portable. Thus applications developed in Java can be moved from one machine to another easily.
- **Robust:** Java makes an effort to eliminate error prone situations by emphasizing mainly on compile time error checking and runtime checking.
- **Multithreaded:** With Java's multithreaded feature it is possible to write programs that can perform many tasks simultaneously. This design feature allows the developers to construct interactive applications that can run smoothly.

Cont.

- **Interpreted:** Java is translated from high-level language to machine code one statement at a time.
- **High Performance:** In order to improve performance, Just-In-Time compilers interact with the Java Virtual Machine (JVM) at run time and compile suitable bytecode sequences into native machine code.
- **Dynamic:** Java is considered to be more dynamic than C or C++ since it is designed to adapt to an evolving environment.
  - Java programs can carry extensive amount of run-time information that can be used to verify and resolve accesses to objects on run-time.
  - Also the three configurations of Java make it possible to develop software in almost any scope.

Tools Needed to program in Java

- **Java Development Kit (JDK)-**
  - is a set of tools for developing Java applications.
  - Every JDK always includes a compatible JRE, because running a Java program is part of the process of developing a Java program.
- **Java Runtime Environment (JRE)-**
  - it is a software layer that runs on top of a computer's operating system software.
  - It provides the class libraries and other resources that a specific Java program needs to run.
  - The JRE combines Java code created using the JDK with the necessary libraries required to run it on a JVM and then creates an instance of the JVM that executes the resulting program.
  - JVMs are available for multiple operating systems, and programs created with the JRE will run on all of them.
  - In this way, the JVM within the Java Runtime Environment is what enables a Java program to run in any operating system without modification.

JRE Runtime Architecture

- ClassLoader
  - The Java ClassLoader dynamically loads all classes necessary to run a Java program. Since Java classes are only loaded into memory when they're required, the JRE uses ClassLoaders to automate this process on demand.
- Bytecode verifier
  - Bytecode is a code that lies between low-level and high-level language. The bytecode is not processed by the processor. It is processed by the Java Virtual Machine (JVM)
  - The bytecode verifier ensures the format and accuracy of Java code before it passes to the interpreter. In the event that code violates system integrity or access rights, the class will be considered corrupted and won't be loaded.
- Interpreter
  - After the bytecode successfully loads, the Java interpreter creates an instance of the JVM(calls the JVM) that allows the Java program to be executed natively on the underlying machine.
  - This is where the Just-in-time compilers are used to optimize performance

JVM

- **The *Java Virtual Machine (JVM)***
- Executes live Java applications.
  - Every JRE includes a default JVM, but developers are free to choose another that meets the specific resource needs of their applications.
- **JVM can be viewed in three dimensions**
  1. **A specification** where working of Java Virtual Machine is specified. But implementation provider is independent to choose the algorithm.
  2. **An implementation** Its implementation is known as JRE (Java Runtime Environment). Its implementation has been provided by Oracle.
  3. **Runtime Instance** Whenever you write java command on the command prompt to run the java class, an instance of JVM is created.

Cont.

- Java applications are called WORA (Write Once Run Anywhere).
  - This means a programmer can develop Java code on one system and can expect it to run on any other Java-enabled system without any adjustment.
  - This is all possible because of JVM.
- JVM performs the following activities
  - Loads code
  - Verifies code
  - Executes code
  - Provides runtime environment

Cont.

- **Integrated Development Environment:**
- Used for editing Java programs
  - Some of the most popular Java IDE include
    - NetBeans
    - Eclipse
    - Jcreator
    - Visual Studio
    - Dr. Java
- Netbeans is the most recommended IDE because it has powerful tools contained in its Abstract Window toolkit (AWT) and Swing packages that make it possible to create powerful GUI through drug and drop.
  - It also supports Java Database Connectivity (JDBC) which makes it possible to link java to MySQL database.

How to install the Necessary tools

- Download the latest versions of JDK and JRE from oracle official website.
- Download Netbeans IDE.
- Install the software in the following order
  - JDK
  - JRE
  - Netbeans
- Test the success by running a simple java program in Netbeans IDE.

Java Basic Syntax

When we consider a Java program, it can be defined as a collection of objects that communicate via invoking each other's methods.

The syntax revolves around the following elements.

- **Object –** Objects are real world entities that have identity, states and behaviors.
  - Example: Tommy which belongs to class dog has states - color, breed as well as behavior such as wagging their tail, barking, eating.
  - An object is an instance of a class.
- **Class -** A class can be defined as a template/blueprint that describes the behavior/state that the object of its type supports.
- **Methods -** A method is basically a behavior.
  - A class can contain many methods. It is in methods where the logics are written, data is manipulated and all the actions are executed.
  - For example: barking(), eating ().

Cont.

- **Instance Variables –**They are created when objects of a class are created. Each object has its unique set of instance variables. An object's state is created by the values assigned to these instance variables.
  - Example: if we take a class known as dog with the state as color, name, breed and behavior as barking.
  - Then we can create two objects namely Tommy, and Bosco. Now the instance variables can be as follows
    - Tommy.color= black
    - Tommy. Breed= German _shephard
    - Bosco.color=Brown
    - Bosco.Breed=Golden_Retriver
  - So in this case color and breed are instance variables. As you can see each object has its own values for the instance variables.

Java Syntax Rules

- **Case Sensitivity -** Java is case sensitive, which means identifier **Hello** and **hello** would have different meaning in Java.
- **Class Names -** For all class names the first letter should be in Upper Case. If several words are used to form a name of the class, each inner word's first letter should be in Upper Case.
  - Every Java Program must have a class.
  - **Example**: *class MyFirstJavaClass*
- **Method Names -** All method names should start with a Lower Case letter. If several words are used to form the name of the method, then each inner word's first letter should be in Upper Case.
  - **Example**: *public void myMethodName()*
  - *Public string barking()*
- *U*se Comments to enhance code readability. Java uses same comments syntax as C and C++

Syntax Rules Cont.

- **Program File Name -** Name of the program file should exactly match the class name.
  - When saving the file, you should save it using the class name (Remember Java is case sensitive) and append '.java' to the end of the name (if the file name and the class name do not match, your program will not compile).
  - **Example**: Assume 'MyFirstJavaProgram' is the class name. Then the file should be saved as *'MyFirstJavaProgram.java'*
- **public static void main(String args[]) –** This is the main method.
  - Java program processing starts from the main() method which is a mandatory part of every Java program.
- **Ensure all open carry braces and parenthesis are closed-** hanging braces will result to a syntax error.
- **Follow the rules of naming variables-** any variable declared must adhere to the variable naming conventions.
- **Terminate all executable statements using a semicolon**

Key Points to Note

- The name of the class is usually the name you have saved the file as.
  - For example if you create a project and save it as java101 then the name of the package in which it will be saved will be package java101 while the class name will be public class Java101
- Just like in C and C++ the code to be executed is usually written inside the braces of the main method.
- Global variables, methods, and functions can be created outside the main method.
- Any libraries and packages that are to be used are imported when needed and are placed between the package name and the class name

Writing a Simple Java Program

- Open NetBeans
- Go to files and select new project
- Select Java
- Select Java application
- Click Next
- Name the project as java101
- Select use the name as the class name
- Then the following code will be automatically generated for you by NetBeans

Simple Java Program Structure.

```
package java101;

public class Java101 {

public static void main(String[] args) {
        // TODO code application logic here

    }

}
```

Interpreting the code

```
package java101; /*this represents the package where this program will be stored*/
public class Java101 { /* this represents the class. Public means that it is accessible to all parts of the program */


public static void main(String[] args) { //main method or main function
// body of the program
 // write what you want the program to do here
}
   }
```

A simple Java program.

```
package java101;

public class Java101 {

public static void main(String[] args) {

 System.out.println("Welcome To
Java");
    }
}
```

- The following program outputs “Welcome to Java” in console.
- System.out.println- is the outputting function used in Java.
- It is similar to printf in C

Example two: A program to output I love Java

```
package java101;

public class Java101 {

public static void main(String[] args) {

 System.out.println(“I love Java");
    }
}
```

Practice Exercise

- Write a Java program to output the following lines of code

  My name is Evans

  I am Learning Java

- Write a Java program to output the following in different lines. Note, try an minimize the number of lines of codes used by suing escape sequence
  - Your Name
  - Your home county
  - Your home sub-county
  - Your constituency