1. What is Java?

* History
* Features

1. JDK, JRE and JVM
2. How to set path?
3. Hello Java Program
4. Compile & Runtime errors
5. Command line arguments
6. Naming Convention
7. Data types

* Primitive
* Non Primitive (Wrapper classes)

1. Variables & its types

* Local
* Instance
* Static

1. Decision Making in Java

* Simple if Statement
* if…else Statement
* if...else if…else statement
* Switch statement

**What is Java?**

* **Java** is a **programming language.**
* **Object oriented programming language**.
* Everything is considered as **Objects**.
* Everything in Java is associated with **classes** and **objects**, along with its attributes and methods.
* **A class is a group of objects which have common properties**. It is a template or blueprint from which objects are created. It is a logical entity. It can't be physical.
* **An object is an instance of a class.** A class is a template or blueprint from which objects are created. So, an object is the instance(result) of a class.

For example: in real life, a car is an object. The car has attributes, such as weight and color, and methods, such as drive and brake.

* A Class is like an object constructor, or a "blueprint" for creating objects.
* Java is a high level language.
* **History**
* Java was developed by Sun Microsystems (which is now the subsidiary of Oracle) in the year 1995.
* James Gosling is known as the father of Java.
* Before Java, its name was Oak. Since Oak was already a registered company, so James Gosling and his team changed the name from Oak to Java.
* **Features**

1. **Simple**

* Java is very easy to learn, and its syntax is simple, clean and easy to understand. According to Sun Microsystem, Java language is a simple programming language because:
* Java syntax is based on C++ (so easier for programmers to learn it after C++).
* Java has removed many complicated and rarely-used features, for example, explicit pointers, operator overloading, etc.
* There is no need to remove unreferenced objects because there is an Automatic Garbage Collection in Java.

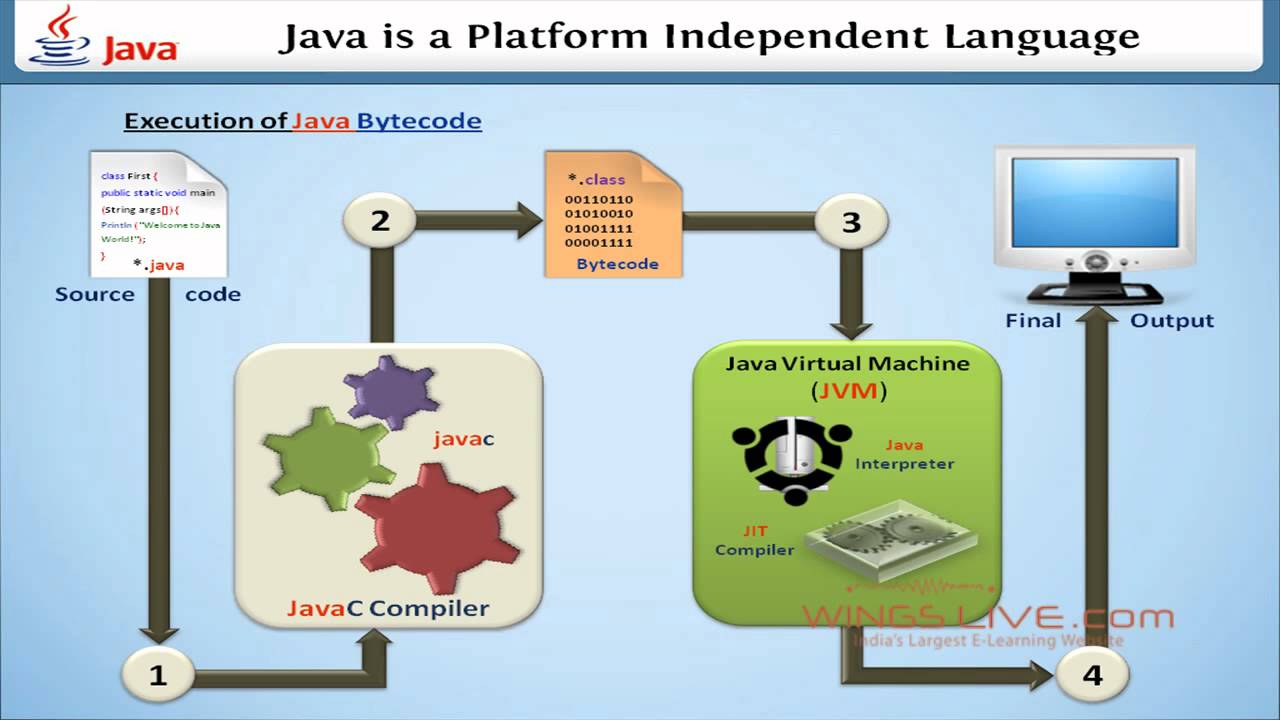
1. **Object Oriented**

* Java is an object-oriented programming language.
* Everything in Java is an object.
* Object-oriented means we organize our software as a combination of different types of objects that incorporate both data and behaviour.
* OOPs is a methodology that simplifies software development and maintenance by providing some rules.
* Basic concepts of OOPs are:

Object, Class, Inheritance, Polymorphism, Abstraction & Encapsulation

1. **Platform Independent**

* Languages like C, C++ compiled into platform specific machines.
* While Java is a write once, run anywhere language.
* A platform is the hardware or software environment in which a program runs.

****

1. **Secured**

* Java is best known for its security.
* With Java, we can develop virus free systems.
* Java is secured because:
* No explicit pointer
* Java Programs run inside a virtual sandbox.

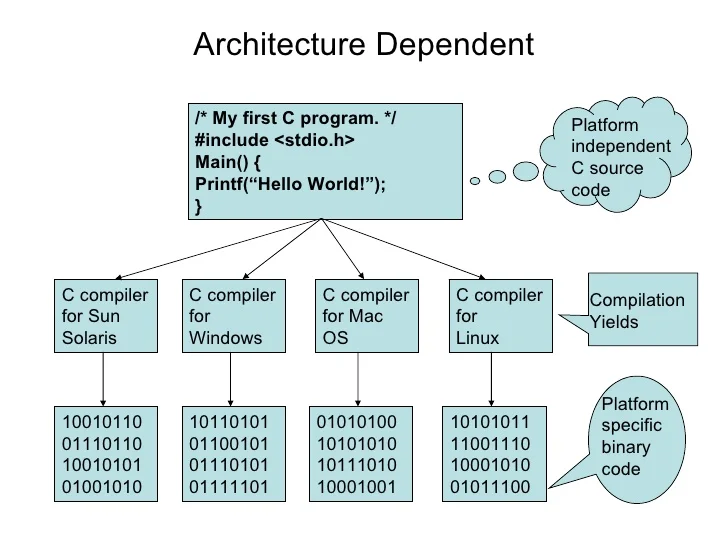
****

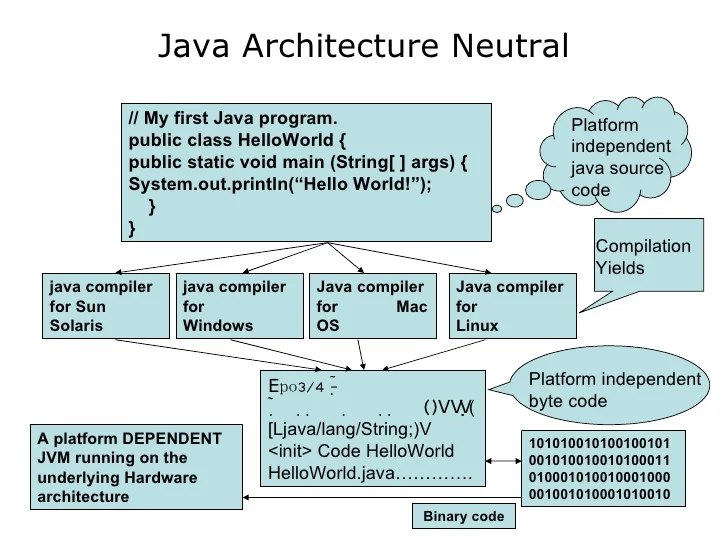
1. **Robust**

* The English mining of Robust is strong. Java is robust because:
* It uses strong memory management.
* There is a lack of pointers that avoids security problems.
* Java provides automatic garbage collection which runs on the Java Virtual Machine to get rid of objects which are not being used by a Java application anymore.
* There are exception handling and the type checking mechanism in Java. All these points make Java robust.

1. **Architecture-neutral**

* No implementation dependent features, for example, the size of primitive types is fixed.
* In C programming, int data types occupies
* 2 bytes of memory for 32-bit architecture
* 4 bytes of memory for 64-bit architecture
* 4 bytes of memory for
* Both 32 and 64-bit architecture in Java.

****

****

1. **Portable**
2. **High-performance**
3. **Distributed**

* Java is distributed because it facilitates users to create distributed applications in Java
* RMI and EJB are used for creating distributed applications.
* This feature of Java makes us able to access files by calling the methods from any machine on the internet.

1. **Multi-threaded**

* A thread is like a separate program, executing concurrently.
* We can write Java programs that deal with many tasks at once by defining multiple threads.
* The main advantage of multi-threading is that it doesn’t occupy memory for each thread.
* It shares a common memory area.
* Threads are important for multi-media, Web applications, etc.

1. **Dynamic**

* Java is a dynamic language.
* It supports the dynamic loading of classes.
* It means classes are loaded on demand.
* It also supports functions from its native languages, i.e., C and C++.

**JVM, JRE and JDK**

* **JVM**
* JVM (Java Virtual Machine) is an abstract machine.
* It is called a virtual machine because it doesn't physically exist.
* It is a specification that provides a runtime environment in which Java bytecode can be executed.
* It can also run those programs which are written in other languages and compiled to Java bytecode.
* JVM (Java Virtual Machine) is an abstract machine.
* It is called a virtual machine because it doesn't physically exist.
* It is a specification that provides a runtime environment in which Java bytecode can be executed.
* It can also run those programs which are written in other languages and compiled to Java bytecode.
* The JVM performs the following main tasks:
* Loads code
* Verifies code
* Executes code
* Provides runtime environment
* **JRE**
* JRE is an acronym for Java Runtime Environment.
* It is also written as Java RTE.
* The Java Runtime Environment is a set of software tools which are used for developing Java applications.
* It is used to provide the runtime environment.
* It is the implementation of JVM. It physically exists. It contains a set of libraries + other files that JVM uses at runtime.

****

* **JDK**
* JDK is an acronym for Java Development Kit.
* The Java Development Kit (JDK) is a software development environment which is used to develop Java applications and [applets](https://www.javatpoint.com/java-applet).
* It physically exists. It contains JRE + development tools.
* JDK is an implementation of any one of the below given Java Platforms released by Oracle Corporation:
* Standard Edition Java Platform
* Enterprise Edition Java Platform
* Micro Edition Java Platform
* The JDK contains a private Java Virtual Machine (JVM) and a few other resources such as an interpreter/loader (java), a compiler (javac), an archiver (jar), a documentation generator (Javadoc), etc. to complete the development of a Java Application.

****

**How to set path?**

* To set the path of JDK, following steps:

1. Open the Control Panel by clicking the Start button and searching for “**Control Panel**”.
2. In the Control Panel, click on “**System and Security**” and then click on “**System**”.
3. Click on “**Advanced system settings**” in the left sidebar.
4. In the System Properties dialog box, click on the “**Environment Variables**” button.
5. In the Environment Variables dialog box, under “**System variables**”, locate the “**Path**” variable and click on “**Edit**”.
6. In the “**Edit** **Environment** **Variable**” dialog box, click on “**New**” and enter the path of your JDK installation folder.

For example, “**C:\Program Files\Java\jdk-16.0.1\bin**”.

1. Click “**OK**” to save the changes.
2. Close all the dialog boxes and open a new command prompt or terminal window.
3. Type the following command to verify that Java is working:

**java -version**

**Hello Java Program**

* 1. **class Helloworld**
  2. **{**
  3. **public static void main(String args[])**
  4. **{**
  5. **String a="A";**
  6. **String b="B";**
  7. **System.out.println("Hello" + a);**
  8. **System.out.println(“Hello”+b)**
  9. **}**
  10. **}**

**Compile & Runtime errors**

**Compile errors**

Compile-time errors occur when there are syntactical issues present in application code, for example, missing semicolons or parentheses, misspelled keywords or usage of undeclared variables.

These syntax errors are detected by the Java compiler at compile-time and an error message is displayed on the screen. The compiler prevents the code from being executed until the error is fixed. Therefore, these errors must be addressed by debugging before the program can be successfully run.

A runtime error in Java is an application error that occurs during the execution of a program. A runtime error occurs when a program is syntactically correct but contains an issue that is only detected during program execution. These issues cannot be caught at compile-time by the Java compiler and are only detected by the Java Virtual Machine (JVM) when the application is running.

Runtime errors are a category of exception that contains several more specific error types. Some of the most common types of runtime errors are:

* IO errors
* Division by zero errors
* Out of range errors
* Undefined object errors

runtime errors occur during program execution (the interpretation phase), after compilation has taken place. Any code that throws a runtime error is therefore syntactically correct.

The most common causes of runtime errors in Java are:

* Dividing a number by zero.
* Accessing an element in an array that is out of range.
* Attempting to store an incompatible type value to a collection.
* Passing an invalid argument to a method.
* Attempting to convert an invalid string to a number.
* Insufficient space in memory for thread data.

**Command line arguments**

The java command-line argument is an argument i.e. passed at the time of running the java program.

The arguments passed from the console can be received in the java program and it can be used as an input.

So, it provides a convenient way to check the behavior of the program for the different values. You can pass **N** (1,2,3 and so on) numbers of arguments from the command prompt.

Simple example of command-line argument in java

|  |
| --- |
| In this example, we are receiving only one argument and printing it. To run this java program, you must pass at least one argument from the command prompt. |

1. **class** CommandLineExample{
2. **public** **static** **void** main(String args[]){
3. System.out.println("Your first argument is: "+args[0]);
4. }
5. }

**Naming Convention**

**Data types**

* **Primitive**
* **Non Primitive (Wrapper classes)**

**Variables & its types**

* **Local**
* **Instance**
* **Static**

**Decision Making in Java**

* **Simple if Statement**
* **if…else Statement**
* **if...else if…else statement**
* **Switch statement**

**1. Sample Hello world.**

**class Simple**

**{**

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

**{**

**System.out.println("Hello World");**

**}**

**}**

**2. Write a program to add two numbers. Accept numbers using command line arguments.**

**public class Command**

**{**

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

**{**

**int a = Integer.parseInt(args[0]);**

**int b = Integer.parseInt(args[1]);**

**int sum = a + b;**

**System.out.println("Sum is " + sum);**

**}**

**}**

**3. Write a program to**

**a) add two integer numbers**

**class SumTwo**

**{**

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

**{**

**int x = 5;**

**int y = 6;**

**int sum = x + y;**

**System.out.println(sum);**

**}**

**}**

**b) add one float number and one integer number**

**import java.util.Scanner;**

**class AddFloatAndInteger {**

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

**Scanner sc = new Scanner(System.in);**

**System.out.print("Enter a float number: ");**

**float floatNumber = sc.nextFloat();**

**System.out.print("Enter an integer number: ");**

**int integerNumber = sc.nextInt();**

**// Calculate the sum of the two numbers**

**float result = floatNumber + integerNumber;**

**// Display the result**

**System.out.println("The sum of " + floatNumber + " and " + integerNumber + " is: " + result);**

**//sc.close(); // Close the scanner to release resources**

**}**

**}**

In this Java program:

1. We import the **java.util.Scanner** class to read input from the user.
2. We create a **Scanner** object to read user input.
3. We prompt the user to enter a float number and an integer number using **System.out.print** and **scanner.nextFloat()** and **scanner.nextInt()** methods, respectively.
4. We calculate the sum of the two numbers and store it in the **result** variable.
5. We display the result using **System.out.println**.
6. Finally, we close the **Scanner** object to release resources.

**c) print your name and age**

**import java.util.Scanner;**

**class NameAndAge {**

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

**Scanner sc = new Scanner(System.in);**

**System.out.print("Enter your name: ");**

**String name = sc.nextLine();**

**System.out.print("Enter your age: ");**

**int age = sc.nextInt();**

**// Display your name and age**

**System.out.println("Your name is: " + name);**

**System.out.println("Your age is: " + age);**

**//sc.close(); // Close the scanner to release resources**

**}**

**}**

In this Java program:

1. We import the **java.util.Scanner** class to read input from the user.
2. We create a **Scanner** object to read user input.
3. We prompt the user to enter their name and age using **System.out.print**, **scanner.nextLine()**, and **scanner.nextInt()** methods.
4. We display the name and age using **System.out.println**.
5. Finally, we close the **Scanner** object to release resources.

**d) calculate area of Triangle**

**import java.util.Scanner;**

**public class TriangleArea {**

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

**Scanner scanner = new Scanner(System.in);**

**System.out.print("Enter the base of the triangle: ");**

**double base = scanner.nextDouble();**

**System.out.print("Enter the height of the triangle: ");**

**double height = scanner.nextDouble();**

**// Calculate the area of the triangle**

**double area = 0.5 \* base \* height;**

**// Display the area of the triangle**

**System.out.println("The area of the triangle is: " + area);**

**scanner.close(); // Close the scanner to release resources**

**}**

**}**

In this program:

1. We import the **java.util.Scanner** class to read input from the user.
2. We create a **Scanner** object to read the base and height of the triangle as double values.
3. We prompt the user to enter the base and height of the triangle using **System.out.print** and **scanner.nextDouble()**.
4. We calculate the area of the triangle using the formula: Area = (0.5) \* base \* height.
5. We display the area of the triangle using **System.out.println**.
6. Finally, we close the **Scanner** object to release resources.

**4. Write a program to swap two numbers with temporary variable. Print the numbers before and after swap.**

**import java.util.Scanner;**

**public class SwapNumbers {**

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

**Scanner scanner = new Scanner(System.in);**

**System.out.print("Enter the first number: ");**

**int num1 = scanner.nextInt();**

**System.out.print("Enter the second number: ");**

**int num2 = scanner.nextInt();**

**// Print the numbers before the swap**

**System.out.println("Before swapping:");**

**System.out.println("First number: " + num1);**

**System.out.println("Second number: " + num2);**

**// Swap the numbers using a temporary variable**

**int temp = num1;**

**num1 = num2;**

**num2 = temp;**

**// Print the numbers after the swap**

**System.out.println("\nAfter swapping:");**

**System.out.println("First number: " + num1);**

**System.out.println("Second number: " + num2);**

**scanner.close(); // Close the scanner to release resources**

**}**

**}**

In this program:

1. We prompt the user to enter two numbers using **System.out.print** and **scanner.nextInt()**.
2. We print the numbers before the swap using **System.out.println**.
3. We swap the two numbers using a temporary variable **temp**.
4. We print the numbers after the swap using **System.out.println**.
5. Finally, we close the **Scanner** object to release resources.

**5. Write a program to swap two numbers „without‟ temporary variable. Print the numbers before and after swap.**

import java.util.Scanner;

public class SwapNumbersWithoutTemp {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the first number: ");

int num1 = scanner.nextInt();

System.out.print("Enter the second number: ");

int num2 = scanner.nextInt();

// Print the numbers before the swap

System.out.println("Before swapping:");

System.out.println("First number: " + num1);

System.out.println("Second number: " + num2);

// Swap the numbers without a temporary variable

num1 = num1 + num2;

num2 = num1 - num2;

num1 = num1 - num2;

// Print the numbers after the swap

System.out.println("\nAfter swapping:");

System.out.println("First number: " + num1);

System.out.println("Second number: " + num2);

scanner.close(); // Close the scanner to release resources

}

}

In this program:

1. We prompt the user to enter two numbers using **System.out.print** and **scanner.nextInt()**.
2. We print the numbers before the swap using **System.out.println**.
3. We swap the two numbers without using a temporary variable by using arithmetic operations:
   * **num1 = num1 + num2** adds the two numbers and stores the result in **num1**.
   * **num2 = num1 - num2** subtracts the original value of **num2** from the sum in **num1** to get the original value of **num1**, and stores it in **num2**.
   * **num1 = num1 - num2** subtracts the original value of **num1** (now stored in **num2**) from the sum in **num1** to get the original value of **num2**, and stores it in **num1**.
4. We print the numbers after the swap using **System.out.println**.
5. Finally, we close the **Scanner** object to release resources.

6.Write a program to check the given number is positive. 7. Write a program to check whether the candidate is eligible for driving license. 8. Write a program to check whether the given number is Odd/Even. 9. Write a program to find largest of three Numbers 10. Write a program to find the grade of a Student based on total marks • Mark less than 40- Failed • 40 to 60–Grade D • 61 to 70-Grade C • 71 to 80-Grade B • 81 to 100-Grade A 11. Write a program to check whether the given character is Vowel/not (Use switch-case).

11.Looping Statements in Java • For • While • Do…while • For-Each Loop 12. Branching Statements in Java • Break • Continue 13. Java Comments 14. Java Array • Single Dimensional • Multi-Dimensional

12. Write a program to print Fibonacci series[0 1 1 2 3 5]. 13. Write a program to find the reverse of number. 14. Write a program to check whether the given number is Palindrome/Not. 15. Write a program to check whether the given number is Armstrong/Not. 16. Write a program to check whether the given number is Prime/Not. 17. Write a program to find the count of even numbers and odd numbers from 10 to 20.