**Java Basics Learning Document**

**1. Structure of Java Program**

A Java program is typically structured into classes and methods. The entry point of every program is the 'main' method. Java code is written inside classes, and each file must have the same name as the public class it contains.

Example:

public class HelloWorld {  
 public static void main(String[] args) {  
 System.out.println("Hello, World!");  
 }  
}

**2. JDK, JRE, JVM**

Java is a platform-independent language due to its architecture. Understanding JDK, JRE, and JVM is important for learners:

• JDK (Java Development Kit): Provides tools for developing Java programs including compiler (javac).

• JRE (Java Runtime Environment): Provides libraries and JVM to run Java programs.

• JVM (Java Virtual Machine): Executes Java bytecode and makes Java platform-independent.

**3. Difference between nextLine() and next()**

Both nextLine() and next() are methods of the Scanner class in Java used to read input from the user, but they work differently:

• next(): Reads input until a space is encountered. It cannot read strings with spaces.

• nextLine(): Reads the entire line of input including spaces until the end of the line.

• Use next() for single words and nextLine() for full sentences.

**4. Arithmetic Program Example**

Example program to perform addition, subtraction, multiplication, and division:

import java.util.Scanner;  
  
public class ArithmeticDemo {  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.in);  
 int a = sc.nextInt();  
 int b = sc.nextInt();  
 System.out.println("Addition: " + (a+b));  
 System.out.println("Subtraction: " + (a-b));  
 System.out.println("Multiplication: " + (a\*b));  
 System.out.println("Division: " + (a/b));  
 }  
}

**5. Leap Year Program**

Leap year is a year divisible by 4 but not by 100, except when it is also divisible by 400.

public class LeapYear {  
 public static void main(String[] args) {  
 int year = 2024;  
 if ((year % 400 == 0) || (year % 4 == 0 && year % 100 != 0)) {  
 System.out.println(year + " is a Leap Year");  
 } else {  
 System.out.println(year + " is not a Leap Year");  
 }  
 }  
}

**6. Student Result Program**

A program to calculate the result of a student based on marks in different subjects:

public class StudentResult {  
 public static void main(String[] args) {  
 int marks1 = 80, marks2 = 75, marks3 = 90;  
 int total = marks1 + marks2 + marks3;  
 double percentage = total / 3.0;  
 System.out.println("Total: " + total);  
 System.out.println("Percentage: " + percentage);  
 }  
}

**7. Odd-Even Program**

Program to check if a number is odd or even:

public class OddEven {  
 public static void main(String[] args) {  
 int num = 10;  
 if (num % 2 == 0) {  
 System.out.println(num + " is Even");  
 } else {  
 System.out.println(num + " is Odd");  
 }  
 }  
}

**8. Exception Handling in Java**

Exceptions are unexpected events that occur during program execution. Java provides a strong mechanism to handle exceptions using try-catch blocks and custom exceptions.

• Checked Exceptions: Must be handled at compile-time (e.g., IOException).

• Unchecked Exceptions: Occur at runtime (e.g., NullPointerException).

• Error: Serious issues not usually handled by programs (e.g., OutOfMemoryError).

Example of try-catch:  
try {  
 int result = 10 / 0;  
} catch (ArithmeticException e) {  
 System.out.println("Cannot divide by zero");  
}

**9. Object-Oriented Programming Concepts**

Java follows Object-Oriented Programming (OOP) principles which make it modular and reusable:

• Encapsulation: Wrapping data and methods together in a class.

• Inheritance: Acquiring properties from a parent class.

• Polymorphism: Ability to take many forms (method overloading and overriding).

• Abstraction: Hiding implementation details and exposing only necessary features.

**10. Collections in Java**

Collections framework provides data structures and algorithms to manage groups of objects. Commonly used classes:

• List: An ordered collection that allows duplicates (e.g., ArrayList).

• Stack: LIFO (Last In First Out) data structure for storing objects.

• Queue: FIFO (First In First Out) structure for scheduling tasks.

• Vector: Legacy class similar to ArrayList but synchronized.