Java Programming Basics

Part 1: Introduction to Java

1. What is Java? Explain its significance in modern software development.

- Java is a popular programming language used to build applications for web, mobile, desktop, and more. It is important because it is fast, secure, and works on different devices without changes.

2. List and explain the key features of Java.
- Platform Independent – Runs on any OS (Windows, Linux, Mac).
- Object-Oriented – Uses objects and classes to organize code.
- Simple & Secure – Easy to learn and protects data.
- Multithreading – Can do multiple tasks at the same time.
- Memory Management – Handles memory automatically using Garbage Collection.

3. What is the difference between compiled and interpreted languages? Where does Java fit in?
- Compiled – Converts full code to machine code before running.
- Interpreted – Translates code line by line while running .
- Java – Uses both! First, it compiles code to Bytecode. Then, the JVM interprets it on any OS.

4. Explain the concept of platform independence in Java.
- Java code runs anywhere without changes because of the JVM (Java Virtual Machine). The code is compiled into Bytecode, which the JVM runs on any OS.

5. What are the various applications of Java in the real world?
- Web Apps – Amazon, Netflix use Java.
- Mobile Apps – Android apps are built in Java.
- Game Development – Games like Minecraft.
- Banking & Finance – Secure banking apps.
- Big Data & AI – Used in Hadoop, Spark for data proce

Part 2: History of Java

1. Who developed Java and when was it introduced?
- Java was developed by James Gosling at Sun Microsystems in 1995.

2. What was Java initially called? Why was its name changed?
- It was first called "Oak." The name was changed to Java because "Oak" was already used for another product.

3. Describe the evolution of Java versions from its inception to the present.
- Java 1.0 (1996) – Basic version.
- Java 5 (2004) – Added Generics and improved performance.
- Java 8 (2014) – Introduced Lambda expressions and Streams.
- Java 17 (2021) – Long-term support version with security updates.

4. What are some of the major improvements introduced in recent Java versions?
- Faster Performance – JIT compiler, Garbage Collection improvements.
- More Security – Enhanced encryption and security updates.
- New Features – Records, Pattern Matching, etc.

5. How does Java compare with other programming languages like C++ and Python in terms of evolution and usability?
- Java vs C++ – Java manages memory automatically, C++ does not.
- Java vs Python – Python is easier to write, but Java is faster.
- Java is the best for large applications!

Part 3: Data Types in Java

1. Explain the importance of data types in Java.
- They specify what kind of data a variable can hold (numbers, text, decimals) to ensure memory efficiency and error prevention.

2. Differentiate between primitive and non-primitive data types.
- Primitive – Simple values like int, char, float.
- Non-Primitive – Complex types like Strings, Arrays, Classes.

3. List and briefly describe the eight primitive data types in Java.

Data Type	Size	Example
byte	1B	byte b = 10;
short	2B	short s = 200;
int	4B	int i = 50000;
long	8B	long l = 100000L;
float	4B	float f = 5.75f;
double	8B	double d = 19.99;
char	2B	char c = 'A';
boolean	1B	boolean b = true;

4. Provide examples of how to declare and initialize different data types.
- int age = 25;
- double price = 99.99;
- char letter = 'J';
- boolean isJavaFun = true;

5. What is type casting in Java? Explain with an example.
- Converting one data type into another.
- Implicit (Automatic) – Small → Large type (int → double).
  e.g. double num = 9.78;
- Explicit (Manual) – Large → Small type (double → int).
  e.g. int intNum = (int) num;

6. Discuss the concept of wrapper classes and their usage in Java.
- They allow primitive types to be used as objects.
  e.g. Integer obj = 10;

7. What is the difference between static and dynamic typing? Where does Java stand?
   - Static (Java, C++) – Variable types are fixed at compile-time.
   - Dynamic (Python, JavaScript) – Variable types change at runtime.
   - Java is statically typed.

Coding Questions on Data Types:

1. Write a Java program to declare and initialize all eight primitive data types and print their values.

```
class Primitive{
    public static void main(String args[]){
        int a = 5;
        float b = 5.33f;
        double c = 5.367387827d;
        char d= 1;
        boolean e = true;

        System.out.println(a);
        System.out.println(b);
        System.out.println(c);
        System.out.println(d);
        System.out.println(e);
    }
}
```

```
C:\Users\Mohd Anas\Desktop\Cdac\Java\Assignments\Assignment_1>java
 Primitive
5
5.33
5.367387827

true
```

2. Write a Java program that takes two integers as input and performs all arithmetic operations on them.

Assignment1.java

```java
import java.util.Scanner;
class Arithematic{
    public static void main(String args[]){
        Scanner obj = new Scanner(System.in);
        System.out.print("Enter First Number: ");
        int num1 = obj.nextInt();
        System.out.print("Enter First Number: ");
        int num2 = obj.nextInt();
        System.out.println("Addition of the numbers num1 and num2: "+(num1+num2));
        System.out.println("Subtraction of the numbers num1 and num2: "+(num1-num2));
        System.out.println("Multiplication of the numbers num1 and num2: "+(num1*num2));
        System.out.println("Division of the numbers num1 and num2: "+(num1/num2));
        System.out.println("Modulus of the numbers num1 and num2: "+(num1%num2));
    }
}
```

```
C:\Users\Mohd Anas\Desktop\Cdac\Java\Assignments\Assignment_1>javac Assignment1.java

C:\Users\Mohd Anas\Desktop\Cdac\Java\Assignments\Assignment_1>java Arithematic
Enter First Number: 15
Enter First Number: 10
Addition of the numbers num1 and num2: 25
Subtraction of the numbers num1 and num2: 5
Multiplication of the numbers num1 and num2: 150
Division of the numbers num1 and num2: 1
Modulus of the numbers num1 and num2: 5

C:\Users\Mohd Anas\Desktop\Cdac\Java\Assignments\Assignment_1>
```

3. Implement a Java program to demonstrate implicit and explicit type casting.

```java
class Widening {
    public static void main(String args[]) {
        int n = 10;
        double d = n;
        System.out.println(d);
    }
}
```

```
C:\Users\Mohd Anas\Desktop\Cdac\Java\Assignments\Assignment_1>javac Assignment1.java

C:\Users\Mohd Anas\Desktop\Cdac\Java\Assignments\Assignment_1>java Widening
10

C:\Users\Mohd Anas\Desktop\Cdac\Java\Assignments\Assignment_1>
```

```java
class Narrowing {
    public static void main(String args[]) {
        double n = 10.9847;
        int num = (int)n;
        System.out.println(num);
    }
}
```

```
C:\Users\Mohd Anas\Desktop\Cdac\Java\Assignments\Assignment_1>java Narrowing
10

C:\Users\Mohd Anas\Desktop\Cdac\Java\Assignments\Assignment_1>
```

4. Create a Java program that converts a given integer to a double and vice versa using wrapper classes.

```
class WrapperConversion {
    public static void main(String[] args) {

        Integer intNum = 50;
        Double doubleNum = intNum.doubleValue();

        System.out.println("Integer value: " + intNum);
        System.out.println("Converted to Double: " + doubleNum);

        Double dblNum = 99.99;
        Integer intValue = dblNum.intValue();

        System.out.println("\nDouble value: " + dblNum);
        System.out.println("Converted to Integer: " + intValue);
    }
}
```

```
C:\Users\Mohd Anas\Desktop\Cdac\Java\Assignments\Assignment_1>java WrapperConversion
Integer value: 50
Converted to Double: 50.0

Double value: 99.99
Converted to Integer: 99
```

5. Write a Java program to swap two numbers using a temporary variable and without using a temporary variable.

```
import java.util.Scanner;
class SwappingNumbers {
    public static void main(String[] args) {
         Scanner sc = new Scanner(System.in);
         System.out.print("Enter first number: ");
         int a = sc.nextInt();
         System.out.print("Enter second number: ");
         int b = sc.nextInt();
        // swapping using temp variable
         int temp = a;
         a = b ;
         b = temp;
         System.out.println("\nAfter Swapping (Using Temp Variable):");
         System.out.println("First number: " + a + ", Second number: " + b);

         //swapping without temp varible
         System.out.print("\nRe-enter first number: ");
         a = sc.nextInt();

         System.out.print("Re-enter second number: ");
         b = sc.nextInt();
         a=a+b;
         b = a-b;
         a=a-b;
         System.out.println("\nAfter Swapping (Without Temp Variable):");
         System.out.println("First number: " + a + ", Second number: " + b);
    }
}
```

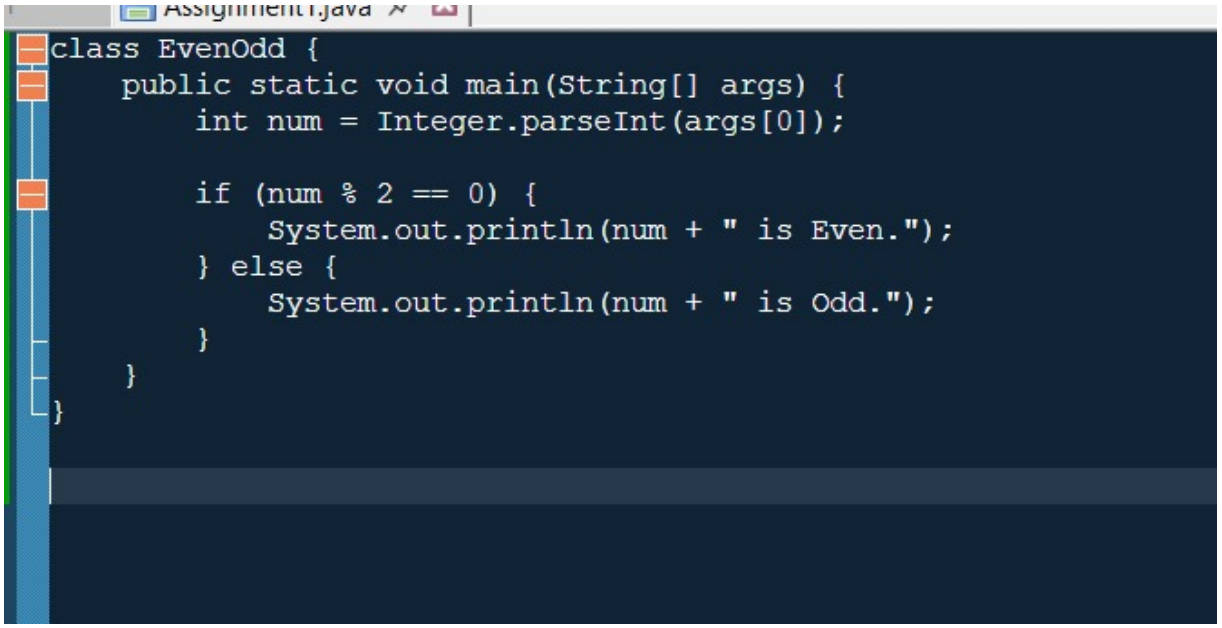
```
C:\Users\Mohd Anas\Desktop\Cdac\Java\Assignments\Assignment_1>java SwappingNumbers
Enter first number: 10
Enter second number: 20

After Swapping (Using Temp Variable):
First number: 20, Second number: 10

Re-enter first number: 10
Re-enter second number: 20

After Swapping (Without Temp Variable):
First number: 20, Second number: 10

C:\Users\Mohd Anas\Desktop\Cdac\Java\Assignments\Assignment_1>
```

6. Develop a program that takes user input for a character and prints whether it is a vowel or consonant.

```java
import java.util.Scanner;

class VowelOrConsonant {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a character: ");
        char ch = sc.next().charAt(0);
        ch = Character.toLowerCase(ch);
        if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u') {
            System.out.println(ch + " is a Vowel.");
        }

        else if (ch >= 'a' && ch <= 'z') {
            System.out.println(ch + " is a Consonant.");
        }

        else {
            System.out.println("Invalid input! Please enter an alphabet.");
        }


    }
}
```

```
C:\Users\Mohd Anas\Desktop\Cdac\Java\Assignments\Assignment_1>java VowelOrConsonant
Enter a character: a
a is a Vowel.

C:\Users\Mohd Anas\Desktop\Cdac\Java\Assignments\Assignment_1>java VowelOrConsonant
Enter a character: z
z is a Consonant.
```

7. Create a Java program to check whether a given number is even or odd using command-line arguments.



```
class EvenOdd {
    public static void main(String[] args) {
        int num = Integer.parseInt(args[0]);

        if (num % 2 == 0) {
            System.out.println(num + " is Even.");
        } else {
            System.out.println(num + " is Odd.");
        }
    }
}
```

```
C:\Users\Mohd Anas\Desktop\Cdac\Java\Assignments\Assignment_1>java EvenOdd 2
2 is Even.

C:\Users\Mohd Anas\Desktop\Cdac\Java\Assignments\Assignment_1>java EvenOdd 3
3 is Odd.
```

Part 4: Java Development Kit (JDK)

1. What is JDK? How does it differ from JRE and JVM?

- JDK (Java Development Kit) – Includes tools for developing, compiling, and running Java programs (JRE + Compiler + Debugger).
- JRE (Java Runtime Environment) – Allows running Java programs (JVM + Libraries).
- JVM (Java Virtual Machine) – Executes Java bytecode, making Java platform-independent.

2. Explain the main components of JDK.

- JVM – Runs Java programs.
- Compiler (javac) – Converts Java code into bytecode.
- Debugger – Helps find errors in code.
- Libraries – Pre-built classes for development.

3. Describe the steps to install JDK and configure Java on your system.

- Download JDK from the official Oracle/OpenJDK website.
- Install it by following on-screen instructions.
- Set PATH variable (Add `bin` folder location to system PATH).

4. Write a simple Java program to print "Hello, World!" and explain its structure.

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

5. What is the significance of the `PATH` and `CLASSPATH` environment variables in Java?

- PATH – Helps the system find Java tools like `javac` and `java`.
- CLASSPATH – Helps Java find user-defined and external classes.

6. What are the differences between OpenJDK and Oracle JDK?

Feature	OpenJDK	Oracle JDK
License	Open-source (free)	Requires a license for commercial use.
Updates	Community-driven	Official support & security updates
Performance	Almost Same	Slight Optimizations

7. Explain how Java programs are compiled and executed.

- Write Code → Save as .java file.
- Compile → Convert code to bytecode using javac.
- Run → JVM executes the bytecode using java

.

8. What is Just-In-Time (JIT) compilation, and how does it improve Java performance?

- Converts bytecode to machine code at runtime for faster execution.
- Reduces repeated interpretation, improving speed.

9. Discuss the role of the Java Virtual Machine (JVM) in program execution.

- Converts bytecode → machine code.
- Manages memory (Garbage Collection).
- Ensures platform independence.