

Java Programming Basic.

classmate

Date _____
Page _____

Unit 1 - Introduction to Java.

1. What is Java? Explain its significance in modern software development.
→ Java is a high level object-oriented & indeed platform independent programming language.
It has very much significance as it is most commonly used programming language with latest LTS versions it increases its significance as it follows the principle of WORA architecture i.e. write once run anywhere.
2. List and explain key features of Java.
→
 1. follows WORA principle.
— this means that the WORA principle is write once run anywhere. It just needed to be written ~~any~~ one time and can run everywhere.
 2. High-level programming language.
 3. Platform independent → can run on any platform like mac os, Linux, & windows.
 4. keeps getting update of latest versions.
3. What is the difference between compiled & interpreted language? Where does Java fit in?
→ Compiled languages translate the entire source code into machine code before execution. (for ex. C, C++) which makes them faster but requires a compilation step.

Interpreted language on other hand translates and execute the code line by line for ex. Java making them more flexible but a bit slower.

with respect to java it is both it first compiled into byte code (by the compiler) & then interpreted/executed by JVM (Java virtual machine). which makes it platform-independent.

4. Explain the concept of Platform Independence in Java.
→ Platform independence in Java means that the Java programs can run on any operating system without any modification. which is achieved through (JVM) which executes the byte code which can run on any operating system.

As long as a system has a proper JVM.

5. What are various applications of Java in the real world?
→ The various applications of Java are as follows-
- 1) web development
 - 2) mobile application.
 - 3) game development
 - 4) cloud computing.
 - 5) cybersecurity.
 - 6) IoT
 - 7) scientific computing.

Part 2: History of Java

1. Who developed Java and when was it introduced?
→ Java was developed by James Arthur Gosling at Sun-microsystem and was introduced in the year 1995.

2. What was Java initially called? Why was its name changed?
→ Java was initially called 'Oak' as there was an oak tree outside James Gosling's office.

The name was changed to Java because Oak was already trademarked by another company.

3. Describe the evolution of Java versions from its inception to the present?

-
- 1996 → first Java Development Kit (JDK 1.0) launched.
 - 1997 → Java became the official language for web development.
 - 1999 → Java 2 (J2SE, J2EE) introduced, bringing significant improvements.
 - 2006 → Sun Microsystems made Java open source & GPL.
 - 2010 → Oracle acquired Sun Microsystems taking over Java development.
 - 2014 → Java 8 released, introducing Lambda expression & Stream API.
 - 2017 → Oracle switched to a faster Java release cycle (every 6 months).
 - 2018 → Java 11 became a long-term support (LTS) version.
 - 2021 → Java 17 released as the next LTS version with modern features.
 - 2024 → Java 21 (latest LTS version) released, bringing virtual threads & pattern matching.

4. What are some of the major improvements introduced in recent Java versions?

- There are some significant improvements.
- Java 21 → Virtual threads for better concurrency, pattern matching, Record Patterns.
 - Java 22 → Enhanced foreign function & memory API, improved Vector API for performance.
 - Java 23 → String templates for better string handling, Variables for cleaner code.

5. How does Java compare with other programming language like C++ and Python in terms of evolution & usability?

→ Java simplifies C++ by removing complexities like pointer and manual memory management while having platform independence via the JVM.

Compared to Python, Java is more structured and optimized for enterprise applications, while Python excels in simplicity, scripting and AI.

Java balances performance, portability and usability making it ideal for large-scale systems.

I Part 3: Data Types in Java

1. Explain the importance of data types in Java.

→ Data types in Java define the type of data a variable can hold, ensuring memory efficiency, type safety & optimized performance.

They help prevent errors, enabling efficient operations & improve code readability.

Java has primitive (e.g. int, double, char) and reference (objects, arrays) data types ensuring structured & reliable programming.

2. Differentiate between primitive & non-primitive data types.

→ Primitive

1) Basic data types that stores values directly

2) Ex. int, double, char, float

3) cannot be null

4) fixed size, faster performance

5) requires less memory, stores actual value.

Non-Primitive.

Reference types that store memory addresses of objects

- String, array, class

- can be null

- Dynamic size more flexible.

- Requires more memory, stores reference.

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

→ 1. byte = 8 bit

1. int data types. → 32 bytes.
Range → ~~00000000000000000000000000000000~~. -2^{31} to $2^{31}-1$

2. short → 16 bits.
Range → -32768 to 32767.

3. long → 64 bits.
Range → $\pm 9.2 \times 10^{18}$

4. float → $\pm 3.4 \times 10^{-38}$
↳ used where regular calculation & small calculations.

5. double → $\pm 1.7 \times 10^{308}$
↳ used where precision is important.

6. char → 0 to 65,535.

7. boolean → true/false.
↳ default value is considered false.

8. byte → stores 8-bit signed integers.
Range → -128 to 127.

4. Provide examples of how to declare & initialize different data types.

→ How to declare.

// Primitive data types.

byte b = 100;

short s = 2000;

Date _____
Page _____

int i = 100000;
long l = 100000000L;
float f = 10.5f;
double d = 99.99;
char c = 'A';
boolean = true;

// Non-Primitive data types.

String str = "Hello, Java!"

int [] arr = {1, 2, 3, 4, 5}; // Array.

Integer objInt = 500; // wrapper class.

5. What is type casting in Java? Explain with example.

→ Type casting is converting one data type into another.

1. Implicit casting (Widening)

- conversion from a smaller to a larger type.

2. Explicit casting (Narrowing)

- Manual conversion from larger to a smaller type.

Ex.

public class TypecastingEx {

 public static void main (String [] args) {

 int num = 100;
 double dbl = num;
 System.out.println ("Widening: "+ dbl);

 double val = 99.99;

 int intVal = (int) val;

 System.out.println ("Narrowing: "+ intVal);

6. Discuss the concept of wrapper classes & their usage in Java.

→ Wrapper classes convert primitive data types into objects. Each primitive type has a corresponding wrapper class. (e.g. → int → Integer, double → Double)

Usage

1. Auto boxing & unboxing → automatic conversion between primitives & objects.
2. Collections framework → used in data structures like ArrayList which work with object, not primitive.
3. Utility methods → Wrapper classes provide useful methods.

Ex.

```
Integer num = 10;
int val = num;
```

7. What is the difference between static & dynamic typing? Where does Java stand?

→

Static typing

Dynamic typing.

Definition → Type checking at compile-time.

flexibility → Less flexible, but safer.

Errors → catches errors early.
detection

Type checking at runtime.

more flexible, but riskier

Errors appear at
runtime.

Ex. → Java, C, C++
language

Python, JavaScript, Ruby

→ JAVA is statically typed meaning variables must have a defined type at compile-time, ensuring better performance.

Part 4 |

1. What is JDK? How does it differ from JRE & JVM?
- JDK refers to Java development kit, which contains both JRE & JVM i.e. Java Runtime environment and Java virtual machine.

JDK is used for developing Java applications.

JRE

JVM

Used for running Java applications.

Executes Java bytecode.

~~JRE~~ JVM + core libraries.

Just the engine that runs bytecode.

End-users (to run Java applications)

Internally used by JRE to execute program.

2. Explain the main components of JDK.
- Java development kit.

1. JRE → It includes JVM & libraries to run Java applications.

2. JVM → Converts the bytecode into machine code for execution.

3. Java compiler (javac) → Converts Java source code into bytecode to compile the Java code.

4. Java Debugger - Helps in debugging Java programs.
5. Java Archive Tool (jar) - packages Java classes into JAR files.
6. Javadoc (Java documentation) - generates documentation from comments.

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

→ 1. Download JDK.

- visit the oracle JDK website
- select the version required as per system.
- Download & run the installer.

2. Installation JDK.

- follow all the steps on installation wizard & complete the setup.
- Note the installation path.

3. Configure environment variables.

- open sys. properties → advanced → environment variables
- under sys. variable find Path & add the JDK bin.
files\java\Jdk-xx\bin

4. Create a new sys. variable.

- Variable name - JAVA_HOME.
- Variable value: C:\Program Files\java\Jdk-xx

5. Verify installation.

- java -version.
- javac -version.

4. Write a simple Java program to print "Hello World" & explain its structure.



public class HelloWorld {

 public static void main(String[] args) {

 System.out.println("Hello, World!");

}
}.

* public class HelloWorld.

- Defines a class named HelloWorld.

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

- The main method, which is the entry point of the program.

- public - accessible from anywhere

- static - runs without creating an object

- void - does not return a value.

- String[] args - command line arguments.

* System.out.println("Hello, World!").

↓
Prints "Hello, World!"

- System.out → standard output stream.

- println() → prints a new line.

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



— PATH Environmental variable.

- Specifies the directory of Java executables (java, javac).
- Allows running Java commands from any location in the terminal or command prompt.

— CLASSPATH Environmental variables.

- Specifies the location of Java class files & libraries.
- Ensure Java can find & load required classes during execution.

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



OpenJDK

Oracle JDK.

License open source.

— free for development, but requires a subscription for commercial use.

Cost completely free.

— paid for commercial use.

Updates community driven updates.

— official Oracle support with security patches.

Support No official support community based.

— Enterprise-level support from Oracle.

7. Explain how Java programs are compiled & executed.



1. Write the Java code

• Create .java file containing Java code.

2. Compilation (javac).

- The Java compiler (javac) converts the source code into bytecode (.class file).

3. Execution (JVM).

- The JVM reads & executes the bytecode.

8. What is just-in-time (JIT) compilation, & how does it improve Java performance?

→ Just-in-time compilation is a technique used by the Java Virtual Machine to improve performance by compiling bytecode into native machine code at runtime.

⇒ JIT improves Java performance.

1. faster execution.

- converts frequently used bytecode into machine code reducing interpretation.

2. optimization

- uses techniques like

3. Adaptive compilation

- detects & optimize hot spots for better speed.

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

→ JVM is responsible for running Java programs by converting bytecode into machine code. ~~provides~~ provides essential runtime services.

Key responsibilities of JVM.

1. loads Bytecode

2. interprets & execute Bytecode

3. memory management.

4. security & Exception handling.

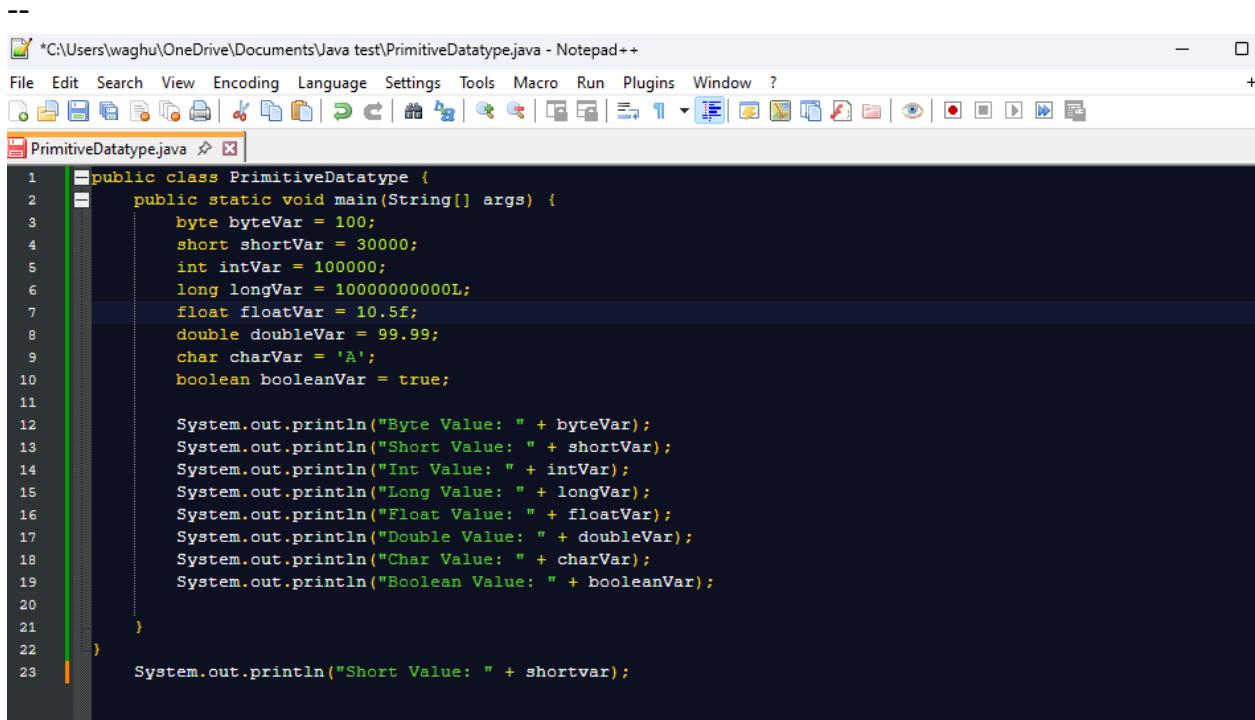
5. Platform Independence

⇒ Execution process → compilation, class loading & execution.

JAVA PROGRAMMING BASCIS

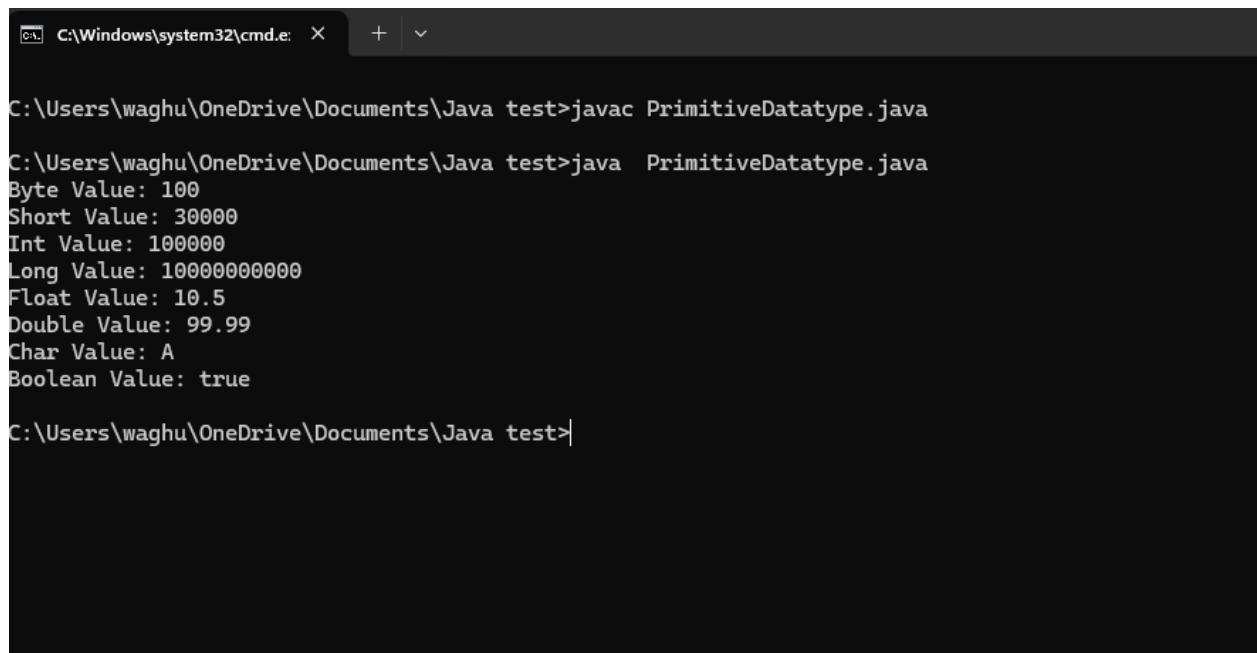
Coding Questions on Data Types:

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



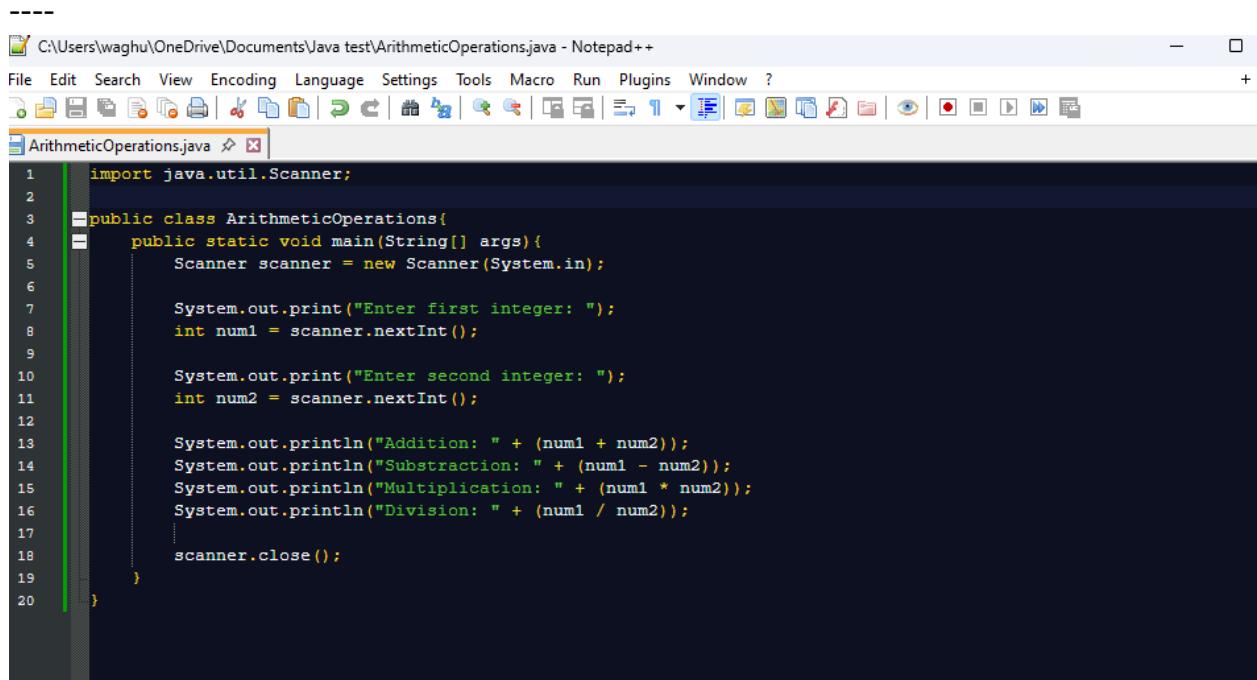
The screenshot shows a Notepad++ window with the file "PrimitiveDatatype.java" open. The code defines a class named "PrimitiveDatatype" with a main method. Inside the main method, various primitive variables are declared and initialized, followed by println statements to print their values. The code is as follows:

```
1 public class PrimitiveDatatype {
2     public static void main(String[] args) {
3         byte byteVar = 100;
4         short shortVar = 30000;
5         int intVar = 100000;
6         long longVar = 1000000000L;
7         float floatVar = 10.5f;
8         double doubleVar = 99.99;
9         char charVar = 'A';
10        boolean booleanVar = true;
11
12        System.out.println("Byte Value: " + byteVar);
13        System.out.println("Short Value: " + shortVar);
14        System.out.println("Int Value: " + intVar);
15        System.out.println("Long Value: " + longVar);
16        System.out.println("Float Value: " + floatVar);
17        System.out.println("Double Value: " + doubleVar);
18        System.out.println("Char Value: " + charVar);
19        System.out.println("Boolean Value: " + booleanVar);
20    }
21
22 }
23 System.out.println("Short Value: " + shortvar);
```



```
C:\Users\waghu\OneDrive\Documents\Java test>javac PrimitiveDatatype.java
C:\Users\waghu\OneDrive\Documents\Java test>java PrimitiveDatatype.java
Byte Value: 100
Short Value: 30000
Int Value: 1000000
Long Value: 10000000000
Float Value: 10.5
Double Value: 99.99
Char Value: A
Boolean Value: true
```

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



```
-----  
C:\Users\waghu\OneDrive\Documents\Java test\ArithmeticOperations.java - Notepad++  
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?  
ArithmeticOperations.java  
1 import java.util.Scanner;  
2  
3 public class ArithmeticOperations{  
4     public static void main(String[] args){  
5         Scanner scanner = new Scanner(System.in);  
6  
7         System.out.print("Enter first integer: ");  
8         int num1 = scanner.nextInt();  
9  
10        System.out.print("Enter second integer: ");  
11        int num2 = scanner.nextInt();  
12  
13        System.out.println("Addition: " + (num1 + num2));  
14        System.out.println("Subtraction: " + (num1 - num2));  
15        System.out.println("Multiplication: " + (num1 * num2));  
16        System.out.println("Division: " + (num1 / num2));  
17  
18    }  
19    scanner.close();  
20}
```

```
C:\Windows\system32\cmd.e: X + ▾

C:\Users\waghu\OneDrive\Documents\Java test>javac ArithmeticOperations.java
C:\Users\waghu\OneDrive\Documents\Java test>java ArithmeticOperations.java
Enter first integer: 10
Enter second integer: 30
Addition: 40
Subtraction: -20
Multiplication: 300
Division: 0

C:\Users\waghu\OneDrive\Documents\Java test>
```

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

```
-----
C:\Users\waghu\OneDrive\Documents\Java test\TypeCasting.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
TypeCasting.java ✘ x
1  public class TypeCasting {
2      public static void main(String[] args) {
3          int intValue = 100;
4          double doubleVal = intValue;
5
6          System.out.println("Implicit Type Casting:");
7          System.out.println("Int value:" + intValue);
8          System.out.println("Converted Double value:" + doubleVal);
9
10         double doubleNum = 99.99;
11         int intNum = (int) doubleNum;
12
13
14         System.out.println("\nExplicit Type Casting:");
15         System.out.println("Double value: " + doubleNum);
16         System.out.println("Converted Int value: " + intNum);
17     }
18 }
19 }
```

```
C:\Windows\system32\cmd.exe + ^

C:\Users\waghu\OneDrive\Documents\Java test>javac TypeCasting.java

C:\Users\waghu\OneDrive\Documents\Java test>java TypeCasting.java
Implicit Type Casting:
Int value:100
Converted Double value:100.0

Explicit Type Casting:
Double value: 99.99
Converted Int value: 99

C:\Users\waghu\OneDrive\Documents\Java test>
```

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

```
----
C:\Users\waghu\OneDrive\Documents\Java test\WrapperClass.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
WrapperClass.java ✘ x
1  public class WrapperClass {
2      public static void main(String[] args) {
3
4          Integer intValue = 100;
5          Double doubleValue = intValue.doubleValue();
6
7          System.out.println("Integer to Double Conversion:");
8          System.out.println("Integer value: " + intValue);
9          System.out.println("Converted Double value: " + doubleValue);
10
11
12          Double doubleNum = 99.99;
13          Integer intNum = doubleNum.intValue();
14
15          System.out.println("\nDouble to Integer Conversion:");
16          System.out.println("Double value: " + doubleNum);
17          System.out.println("Converted Integer value: " + intNum);
18      }
19  }
20
```

```
C:\Users\waghlu\OneDrive\Documents\Java test>javac WrapperClass.java
C:\Users\waghlu\OneDrive\Documents\Java test>java WrapperClass.java
Integer to Double Conversion:
Integer value: 100
Converted Double value: 100.0

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

C:\Users\waghlu\OneDrive\Documents\Java test>
```

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

```
----
SwapNumbers.java
1 import java.util.Scanner;
2
3 public class S0 {
4     public static void main(String[] args) {
5         Scanner scanner = new Scanner(System.in);
6
7         System.out.print("Enter first number: ");
8         int num1 = scanner.nextInt();
9
10        System.out.print("Enter second number: ");
11        int num2 = scanner.nextInt();
12
13
14        int temp = num1;
15        num1 = num2;
16        num2 = temp;
17
18        System.out.println("\nAfter swapping using temporary variable:");
19        System.out.println("First number: " + num1);
20        System.out.println("Second number: " + num2);
21
22
23        num1 = num1 + num2;
24        num2 = num1 - num2;
25        num1 = num1 - num2;
26
27        System.out.println("\nAfter swapping without using temporary variable:");
28        System.out.println("First number: " + num1);
29        System.out.println("Second number: " + num2);
30
31        scanner.close();
32
33    }
34}
```

```
C:\Windows\system32\cmd.exe X + ^

C:\Users\waghu\OneDrive\Documents\Java test>javac SwapNumbers.java

C:\Users\waghu\OneDrive\Documents\Java test>java SwapNumbers.java
Enter first number: 50
Enter second number: 10

After swapping using temporary variable:
First number: 10
Second number: 50

After swapping without using temporary variable:
First number: 50
Second number: 10

C:\Users\waghu\OneDrive\Documents\Java test>
```

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

```
-----
* C:\Users\waghu\OneDrive\Documents\Java test\VowelorConsonant.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
VowelorConsonant.java ✘

1 import java.util.Scanner;
2 public class VowelorConsonant {
3     public static void main(String[] args) {
4         Scanner scanner = new Scanner(System.in);
5
6
7         System.out.print("Enter a character: ");
8         char ch = scanner.nextLine().toLowerCase().charAt(0);
9
10
11        if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u') {
12            System.out.println(ch + " is a vowel.");
13        } else if (Character.isLetter(ch)) {
14            System.out.println(ch + " is a consonant.");
15        } else {
16            System.out.println("Invalid input. Please enter an alphabet.");
17        }
18
19        scanner.close();
20    }
21 }
```

```
C:\Users\waghu\OneDrive\Documents\Java test>javac VowelorConsonant.java
C:\Users\waghu\OneDrive\Documents\Java test>java VowelorConsonant.java
Enter a character: a
a is a vowel.

C:\Users\waghu\OneDrive\Documents\Java test> java VowelorConsonant.java
Enter a character: b
b is a consonant.

C:\Users\waghu\OneDrive\Documents\Java test>
```

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

```
----
C:\Users\waghu\OneDrive\Documents\Java test\EvenOrOdd.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
File Project View Insert Tools Run Plugins Window Help
EvenOrOdd.java
1 public class EvenOrOdd {
2     public static void main(String[] args) {
3         if (args.length == 0) {
4             System.out.println("Please enter a number.");
5             return;
6         }
7
8         int number = Integer.parseInt(args[0]);
9
10        if (number % 2 == 0) {
11            System.out.println(number + " is even.");
12        } else {
13            System.out.println(number + " is odd.");
14        }
15    }
16
17 }
```

```
C:\Windows\system32\cmd.exe + ^

C:\Users\waghu\OneDrive\Documents\Java test>javac EvenOrOdd.java

C:\Users\waghu\OneDrive\Documents\Java test>java EvenOrOdd.java
Please enter a number.

C:\Users\waghu\OneDrive\Documents\Java test> java EvenOrOdd 5
5 is odd.

C:\Users\waghu\OneDrive\Documents\Java test>java EvenOrOdd 8
8 is even.

C:\Users\waghu\OneDrive\Documents\Java test>
```