



# **FULL STACK DEVELOPMENT: WORKSHEET - 2**

**Q1. Java method overloading implements the OOPS concept**

- A. Encapsulation
- B. Inheritance
- C. Polymorphism
- D. Abstraction

**Ans.** Method overloading in Java implements the OOPS concept of "Polymorphism."

**Q2. Data members and member functions of a class are private by default.**

- A. True
- B. False
- C. Depend on code
- D. None

**Ans.** A. True

**Q3. Which of the following functions can be inherited from the base class?**

- A. Constructor
- B. Static
- C. All
- D. None

**Ans.** A. Constructor

**Q4. Identify the feature, which is used to reduce the use of nested classes.**

- A. Binding
- B. Abstraction
- C. Inheritance
- D. None

**Ans.** Abstraction

**Q5. Which concept of Java is achieved by combining methods and attributes into a class?**

- A. Encapsulation
- B. Inheritance
- C. Polymorphism
- D. Abstraction

**Ans.** A. The concept of Java achieved by combining methods and attributes into a class is "Encapsulation."

**Q6. Which of the following declarations does not compile?**

- A. double num1, int num2 = 0;
- B. int num1, num2;
- C. int num1, num2 = 0;
- D. int num1 = 0, num2 = 0;

**Ans.** A. The declaration that does not compile is double num1, int num2 = 0;

**Q7. Which of these interface must contain a unique element?**

- A. Set
- B. List
- C. Array
- D. collection

**Ans.** A. Set

**Q8.Predict the output?**

```
package main;  
  
class T {  
  
    int t = 20;  
  
}  
  
class Main {  
  
    public static void main(String args[]) {  
  
        T t1 = new T();  
  
        System.out.println(t1.t);  
  
    }  
  
}
```

- A. 20
- B. 0
- C. COMPILE ERROR

**Ans.** The output of the given Java code will be: A. 20

Explanation:

1. We have a class called T, and its instance variable t has the value 20 set to it.
2. Using the constructor new T(), we create an object of type T called t1 in the main function of the Main class.
3. Then, we use System.out.println(t1.t); to access the instance variable t of the t1 object.

The t1 object will output 20 when we access it since t is initialised to 20 in the T class. As a result, option A produces the desired result. The provided code compiles without errors.

**Q9. What is the output of the below Java program?**

**//bingo.java file**

**public class Hello**

**{**

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

**{**

**System.out.println("BINGO");**

**}**

**}**

- A. BINGO
- B. bingo
- C. 0
- D. Compile Error

**Ans.** The output of the given Java program will be:

- A. BINGO

Explanation:

- 1.The programme creates a Hello class with a main function.
  - 2.System.out.println("BINGO"); is used in the main function to print the string "BINGO" to the console.
  - 3.As "BINGO" and "bingo" are regarded as separate strings in Java due to case-sensitivity, the programme expressly writes "BINGO" in capital letters.
- As a result, "BINGO" (option A) is the program's output, and the given code has no compilation errors.

**Q10.What will be the output of the following Java program?**

```
class variable_scope  
  
{  
  
    public static void main(String args[])  
  
    {  
  
        int x;  
  
        x = 5;  
  
        {  
  
            int y = 6;  
  
            System.out.print(x + " " + y);  
  
        }  
  
        System.out.println(x + " " + y);  
  
    }  
  
}
```

A. Compilation Error

B. Runtime Error

C. 5 6 5 6

D. 5 6 5

**Ans.** The output of the given Java program will be:

A. Compilation Error

Explanation:

- 1.The programme creates the variable\_scope class, which has a main function.
- 2.In the main procedure, the statements `int x;` and `x = 5;` are both present.
- 3.The next block of code, which is encased in curly brackets, declares and initialises a new variable, `int y = 6;`
- 4.It prints `x + " " + y` inside this block, printing "5 6" as a result of printing the values of x and y inside this block.

There will be a compilation issue if it tries to print `x + " " + y` outside of the block. This is so that it is impossible to access the variable y outside of the block, which contains its declaration. As a result, the code cannot be compiled, and option A (Compilation Error) is shown.

**Q11.What will be the output of the following Java code?**

```
class String_demo  
  
{  
  
    public static void main(String args[])  
  
    {  
  
        char chars[] = {'a', 'b', 'c'};  
  
        String s = new String(chars);  
  
        System.out.println(s);  
  
    }  
  
    }
```

- A. abc
- B. a
- C. b
- D. c

**Ans.** The output of the given Java code will be:

A. abc

Explanation:

- 1.The programme defines a String\_demo class that has a main function.
- 2.It generates the character array char chars[] = 'a', 'b', 'c'; within the main function.
- 3.Then, using the character array chars, a new String object s is generated.
- 4.Finally, it uses System.out.println(s); to print the value of s.



Since the String constructor is used to produce strings from character arrays, it will produce a string that contains the characters in the arrays 'a', 'b', and 'c', resulting in the console output of "abc" (option A).

**Q12. What will be the output of the following Java program?**

```
final class A

{
    int i;
}

class B extends A

{
    int j;

    System.out.println(j + " " + i);
}

class inheritance
{
    public static void main(String args[])
    {
        B obj = new B();

        obj.display();
    }
}
```

A. 2 2

B. 3 3

C. Runtime Error

D. Compilation Error

**Ans.** The given Java program has several issues and will result in a compilation error. Therefore, the correct answer is:

D. Compilation Error

Explanation:

1.The i variable is declared in the A class, but no constructor is provided to initialise it.

2.You are trying to access j and i in the B class outside of any method or constructor when you use a print statement. Java doesn't allow this. A method or constructor must contain the code that should be run.

3.You are attempting to call the B object's nonexistent function display() in the inherited class's main method. The B class doesn't have a display() function defined.

4.Since the B class lacks a constructor, it will try to apply the default constructor of A. However, because A is declared as final, it cannot be extended, which creates a different problem.

The proper response is option D (Compilation Error), as the code cannot be compiled as a result of these compilation issues.

**Q13.What is output of following program**

```
public class Test  
  
{  
  
    public int getData() //getdata() 1  
  
    {  
  
        return 0;  
  
    }  
  
    public long getData() //getdata 2  
  
    {  
  
        return 1;  
  
    }  
  
    public static void main(String[] args)  
  
    {  
  
        Test obj = new Test();  
  
        System.out.println(obj.getData());  
  
    }  
  
}
```

- A. 1
- B. 0
- C. Runtime Error
- D. Compilation Error

**Ans.** The given Java program has several issues and will result in a compilation error. Therefore, the correct answer is:

D. Compilation Error

Explanation:

The Java programme offered defines the class "Test" and its two methods as follows:

1.getData() is a public method that returns an integer (int).

2.getData() is a public method that returns a long integer (long).

Both methods have the same name (getData) and argument list (none), but they differ in their return types (int and long), which is the problem in this case. Method overloading is allowed in Java when there are differences in the parameters (number or type), but it isn't allowed when there are just changes in the return types. The compiler then has trouble deciding which getData method to call as a result.

The code will not compile as a result, and a compilation error will be generated (option D).

**Q14. What is the output of the following program?**

```
public class Test{  
    static int start = 2;  
    final int end;  
    public Test(int x) {  
        x = 4;  
        end = x;  
    }  
    public void fly(int distance) {  
        System.out.println(end-start+" ");  
        System.out.println(distance);  
    }  
    public static void main(String []args){  
        new Test(10).fly(5);  
    }  
}
```

- A. [2 5]
- B. [0 0]
- C. [5 2]
- D. [0 2]

**Ans.** The output of the given Java program will be:

C. [5 2]

Explanation:

1.A new Test object is created in the main procedure using the constructor new Test(10). This constructor sets the value of x, which is 10, to the end instance variable. End thus becomes 10.

2.Then, the newly created Test object receives a call to the fly(5) function. In this method:

- Calculating end-start, which is 10-2, results in the number 8.
- The printed distance is 5.

So, as a consequence, the programme prints "8" (including the space) then "5", resulting in the output "[5 2]" (option C).

**Q15.What is the output of the following program?**

```
String john = "john";
```

```
String jon = new String(john);
```

```
System.out.println((john==jon) + " "+ (john.equals(jon)));
```

A. true true

B. true false

C. false true

D. false false

**Ans.** The output of the given Java program will be:

C. false true

Explanation:

1.String john = "john";: The string "john" is created as a string literal and assigned to the variable john in this line.

2. `John = new String(string jon);` Using the new keyword and the value of the john variable, this line creates a new string object named jon. It's important to observe that despite having the same content as the string literal "john," this produces a new string object on the heap.

3. `(John==Jon)` compares John and Jon's references. Since `(john==jon)` evaluates to false because john is a string literal and jon is a new string object, their references in memory are different.

4. Compares the contents of john and jon using `(john.equals(jon))`. Because both strings in this case include "john," which is what the equals function looks for, the evaluation of `(john.equals(jon))` returns true.

As a result, the programme outputs "false true" (option C).

**Q16. Given that Student is a class, how many reference variables and objects are created by the following code?**

```
Student studentName, studentId;
```

```
studentName = new Student();
```

```
Student stud_class = new Student();
```

- A. Three reference variables and two objects are created.
- B. Two reference variables and two objects are created.
- C. One reference variable and two objects are created.
- D. Three reference variables and three objects are created.

**Ans.** The output of the given Java program will be:

- B. Two reference variables and two objects are created.

Explanation:

The following code creates: 1. Two reference variables: studentName and studentId.

2. Two objects of the Student class: one when studentName is initialized, and another when stud\_class is initialized.

So, the correct answer is: B. Two reference variables and two objects are created.

### Q17. Write a java program to check even or odd number.

**Ans.** import java.util.Scanner;

```
class EvenOddChecker {  
  
    public static void main(String[] args) {  
  
        Scanner scanner = new Scanner(System.in);  
  
        System.out.print("Enter a number: ");  
  
        int number = scanner.nextInt();  
  
        if (number % 2 == 0) {  
  
            System.out.println(number + " is even.");  
  
        } else {  
  
            System.out.println(number + " is odd.");  
  
        }  
  
        scanner.close();  
  
    }  
  
}
```

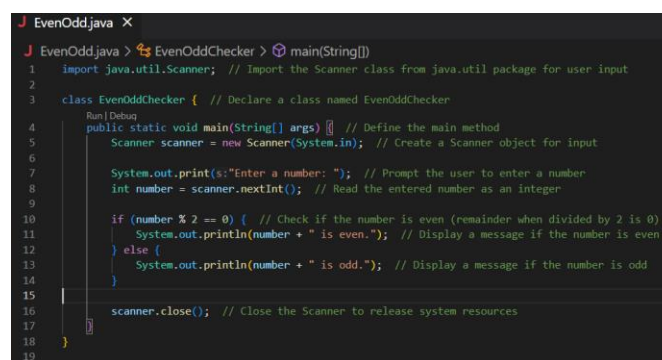


Figure 1: Even and Odd number



The Java programme in this code determines if a given integer is even or odd. Let's walk through the code piece by piece and describe each component:

- **import java.util.Scanner:** This line imports the Scanner class, which is used to read input from the user, from the java.util package.
- **class EvenOddChecker:** This specifies the EvenOddChecker class. Each class in a Java programme has methods that specify the program's behaviour, and a Java programme normally consists of one or more classes.
- **public static void main(String[] args):** This is the class's main function and the place where the Java programme starts. When the programme is run, it is the method that is executed.
- **Scanner scanner = new Scanner(System.in):** This statement prompts the user to enter a number by printing the message "Enter a number:".
- **int number = scanner.nextInt():** This line uses the nextInt() function of the Scanner class to read an integer input from the user and stores it in the number variable.
- **If (number% 2 == 0):** The if statement is introduced by this line. It determines if the criteria for an even number—that the remainder of the number divided by two equals zero—is met.
- **System.out.println(number + " is even."):** If the condition in the if statement is true, this line is executed. The input number is even, according to the message that is printed.
- **else:** This line starts the else block, which includes code that will run if the if statement's condition is false.
- **System.out.println(number + "is odd."):** This line is performed if the if statement's condition is false. The oddness of the input number is indicated by a message that is printed.
- **Scanner.close():** This line closes the Scanner object and releases all of its resources.

This programme will ask you for a number when you run it. When a number is entered, the programme checks to see if it's even or odd, and then it displays the relevant message.

**Q18. Write a java program to find average of two numbers.**

**Ans.** import java.util.Scanner;

```
public class AverageCalculator {  
    public static void main(String[] args) {  
        Scanner scanner = new Scanner(System.in);  
  
        // Step 1: Prompt the user to enter the first number  
        System.out.print("Enter the first number: ");  
        double num1 = scanner.nextDouble(); // Step 2: Read the first number  
  
        // Step 3: Prompt the user to enter the second number  
        System.out.print("Enter the second number: ");  
        double num2 = scanner.nextDouble(); // Step 4: Read the second number  
  
        // Step 5: Calculate the average of the two numbers  
        double average = (num1 + num2) / 2; // Step 6: Calculate the average  
  
        // Step 7: Display the result  
        System.out.println("The average of " + num1 + " and " + num2 + " is: " + average);  
        scanner.close();  
    }  
}
```

```
J AverageCalculator.java X
J AverageCalculator.java > ...
1  import java.util.Scanner;
2
3  public class AverageCalculator {
4      public static void main(String[] args) {
5          Scanner scanner = new Scanner(System.in);
6
7          // Step 1: Prompt the user to enter the first number
8          System.out.print(s:"Enter the first number: ");
9          double num1 = scanner.nextDouble(); // Step 2: Read the first number
10
11         // Step 3: Prompt the user to enter the second number
12         System.out.print(s:"Enter the second number: ");
13         double num2 = scanner.nextDouble(); // Step 4: Read the second number
14
15         // Step 5: Calculate the average of the two numbers
16         double average = (num1 + num2) / 2; // Step 6: Calculate the average
17
18         // Step 7: Display the result
19         System.out.println("The average of " + num1 + " and " + num2 + " is: " + average);
20
21         scanner.close();
22     }
23 }
24
```

Figure 2: Average of two numbers

Explanation of the code:

- **import java.util.Scanner;** : The Scanner class from the java.util package is imported in this line, allowing us to read user input.
- **public class AverageCalculator {** : The AverageCalculator Java class is introduced at this line.
- **public static void main(String[] args) {** : The main method, which is the program's entry point, is declared on this line. Although it accepts an array of strings as arguments (args), this programme does not use it.
- **Scanner scanner = new Scanner(System.in);** : Here, we make a new Scanner object called scanner to read data from the keyboard's standard input stream.
- **System.out.print("Enter the first number: ");** : Using System.out.print, this line produces a prompt asking the user to input the first number before moving to the next line.
- **double num1 = scanner.nextDouble();** : The initial number that the user input using the scanner is read out on this line.nextDouble() and stores it in the variable num1 as a double-precision floating-point value.
- **System.out.print("Enter the second number: ");** : Instead of going to the next line, this line creates a message box asking the user to input the second number.
- **double num2 = scanner.nextDouble();** : The second number input by the user using a scanner is read on this line.nextDouble() and stores it in the variable num2 as a double-precision floating-point value.
- **double average = (num1 + num2) / 2;** : Here, we add num1 and num2, then divide the result by 2, to determine the average of the two integers. The variable average contains the calculated average.

- **System.out.println("The average of " + num1 + " and " + num2 + " is: " + average);:** System.out.println() is used in this line to provide the calculated average and the actual input data in a clear and understandable manner. To generate the output message, it applies the + operator to strings.
- **scanner.close(); :** To save up system resources, we close the Scanner object finally.

Two numbers are entered into the programme, which then calculates their average and shows the result in an easily understood way.

**Q19. Write a java program to swap two numbers.**

**Ans.** import java.util.Scanner; // Import the Scanner class from java.util package for user input

```
public class SwapNumbers { // Declare a class named SwapNumbers

    public static void main(String[] args) { // Define the main method

        Scanner scanner = new Scanner(System.in); // Create a Scanner object for input


        // Prompt the user to enter the first number

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

        int num1 = scanner.nextInt(); // Read the entered first number as an integer


        // Prompt the user to enter the second number

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

        int num2 = scanner.nextInt(); // Read the entered second number as an integer


        System.out.println("Before swapping:"); // Print a message before swapping

        System.out.println("First number: " + num1); // Print the value of the first number

        System.out.println("Second number: " + num2); // Print the value of the second number


        // Swap the numbers using a temporary variable

        int temp = num1; // Create a temporary variable and store the value of num1

        num1 = num2; // Assign the value of num2 to num1

        num2 = temp; // Assign the original value of num1 (stored in temp) to num2
```

```
System.out.println("After swapping:"); // Print a message after swapping
```

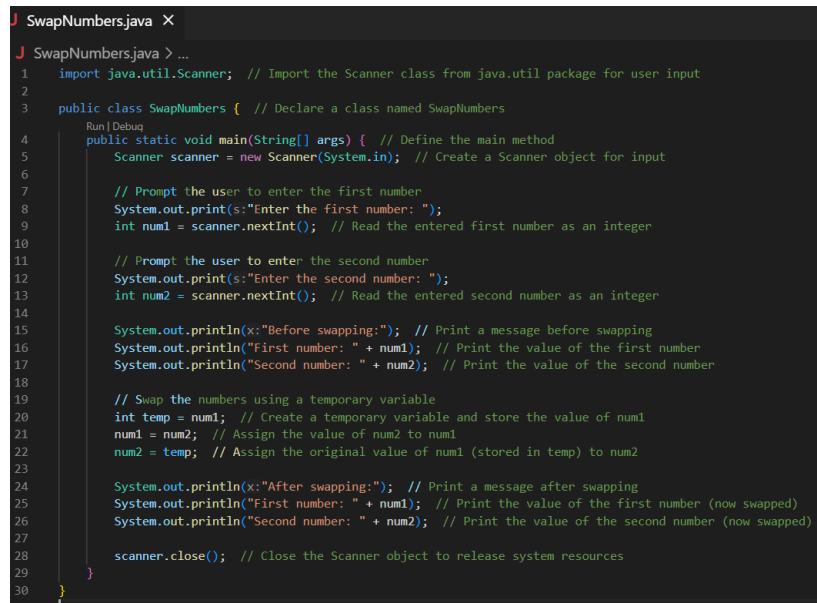
```
System.out.println("First number: " + num1); // Print the value of the first number (now swapped)
```

```
System.out.println("Second number: " + num2); // Print the value of the second number (now swapped)
```

```
scanner.close(); // Close the Scanner object to release system resources
```

```
}
```

```
}
```



```
SwapNumbers.java X
SwapNumbers.java > ...
1 import java.util.Scanner; // Import the Scanner class from java.util package for user input
2
3 public class SwapNumbers { // Declare a class named SwapNumbers
4     public static void main(String[] args) { // Define the main method
5         Scanner scanner = new Scanner(System.in); // Create a Scanner object for input
6
7         // Prompt the user to enter the first number
8         System.out.print(s:"Enter the first number: ");
9         int num1 = scanner.nextInt(); // Read the entered first number as an integer
10
11        // Prompt the user to enter the second number
12        System.out.print(s:"Enter the second number: ");
13        int num2 = scanner.nextInt(); // Read the entered second number as an integer
14
15        System.out.println(x:"Before swapping:"); // Print a message before swapping
16        System.out.println("First number: " + num1); // Print the value of the first number
17        System.out.println("Second number: " + num2); // Print the value of the second number
18
19        // Swap the numbers using a temporary variable
20        int temp = num1; // Create a temporary variable and store the value of num1
21        num1 = num2; // Assign the value of num2 to num1
22        num2 = temp; // Assign the original value of num1 (stored in temp) to num2
23
24        System.out.println(x:"After swapping:"); // Print a message after swapping
25        System.out.println("First number: " + num1); // Print the value of the first number (now swapped)
26        System.out.println("Second number: " + num2); // Print the value of the second number (now swapped)
27
28        scanner.close(); // Close the Scanner object to release system resources
29    }
30 }
31
```

Figure 3: Swap of two numbers

Explanation of the code :

- To accept user input, import the java.util.Scanner class from the java.util package.
- Create the SwapNumbers class to include the program's functionality.
- Define the main method, which acts as the program's entry point.
- To read user input, create a Scanner object called scanner.
- Request the initial number be entered by the user using System.out."Enter the first number:" is printed.

- Scanner reads the entered value as an integer.nextInt(), and then save it in the num1 variable.
- Request the second number from the user using System.out."Enter the second number:" is printed.
- Scanner reads the entered value as an integer.the result of nextInt() into the variable num2.
- Use System.out.println("Before swapping:"); to print a message stating that the numbers are being shown before swapping.
- Use System.out.println("First number: " + num1) and System.out.println("Second number: " + num2) to print the values of the first and second numbers, respectively.
- Make a temporary variable called temp to hold the value of num1 and use it to switch the numbers.
- By assigning num2's value to num1, you may effectively swap their values.
- Give number two the initial value of number one (stored in temporary).
- Use System.out.println("After swapping:"); to print a message indicating that the numbers are being shown after swapping.
- Use System.out.println("First number: " + num1) and System.out.println("Second number: " + num2) to print the values of the first and second integers, which have now been swapped.
- Close the scanner-enabled item.To free up system resources, use close().

The programme accepts two numbers as input, uses a temporary variable to swap the values, and then shows the original and swapped values. It makes sure that the user is guided through the procedure by clear signals.

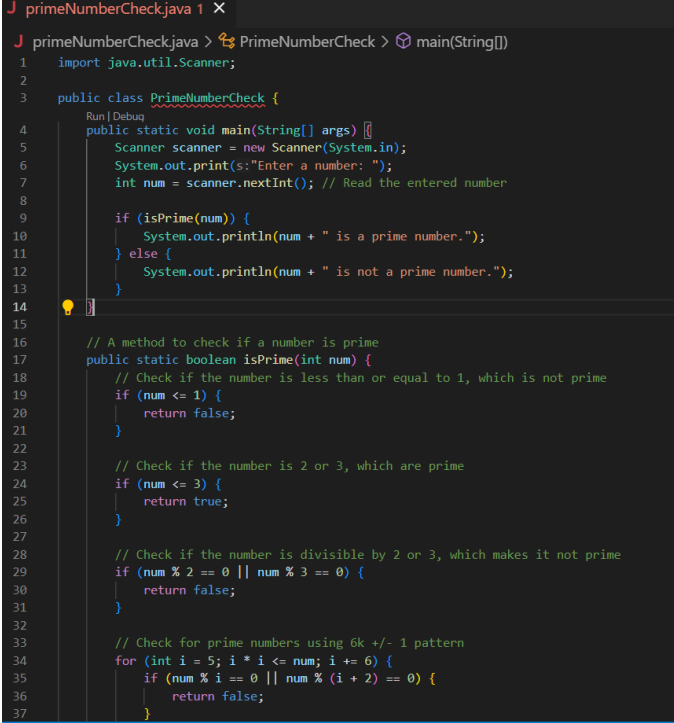
**Q20. Write a java program to check whether a number is prime or not.**

**Ans.** import java.util.Scanner;

```
public class PrimeNumberCheck {  
    public static void main(String[] args) {  
        Scanner scanner = new Scanner(System.in);  
        System.out.print("Enter a number: ");  
        int num = scanner.nextInt(); // Read the entered number  
        if (isPrime(num)) {  
            System.out.println(num + " is a prime number.");  
        } else {  
            System.out.println(num + " is not a prime number.");  
        }  
    }  
    // A method to check if a number is prime  
    public static boolean isPrime(int num) {  
        // Check if the number is less than or equal to 1, which is not prime  
        if (num <= 1) {  
            return false;  
        }  
        // Check if the number is 2 or 3, which are prime  
        if (num <= 3) {  
            return true;  
        }  
        // Check if the number is divisible by 2 or 3, which makes it not prime
```



```
    if (num % 2 == 0 || num % 3 == 0) {  
        return false;  
    }  
    // Check for prime numbers using 6k +/- 1 pattern  
    for (int i = 5; i * i <= num; i += 6) {  
        if (num % i == 0 || num % (i + 2) == 0) {  
            return false;  
        }  
    }  
    // If none of the above conditions are met, the number is prime  
    return true;  
}  
}
```



```
J primeNumberCheck.java 1 X  
J primeNumberCheck.java > PrimeNumberCheck > main(String[])  
1 import java.util.Scanner;  
2  
3 public class PrimeNumberCheck {  
4     public static void main(String[] args) {  
5         Scanner scanner = new Scanner(System.in);  
6         System.out.print(s:"Enter a number: ");  
7         int num = scanner.nextInt(); // Read the entered number  
8  
9         if (isPrime(num)) {  
10            System.out.println(num + " is a prime number.");  
11        } else {  
12            System.out.println(num + " is not a prime number.");  
13        }  
14    }  
15  
16    // A method to check if a number is prime  
17    public static boolean isPrime(int num) {  
18        // Check if the number is less than or equal to 1, which is not prime  
19        if (num <= 1) {  
20            return false;  
21        }  
22  
23        // Check if the number is 2 or 3, which are prime  
24        if (num <= 3) {  
25            return true;  
26        }  
27  
28        // Check if the number is divisible by 2 or 3, which makes it not prime  
29        if (num % 2 == 0 || num % 3 == 0) {  
30            return false;  
31        }  
32  
33        // Check for prime numbers using 6k +/- 1 pattern  
34        for (int i = 5; i * i <= num; i += 6) {  
35            if (num % i == 0 || num % (i + 2) == 0) {  
36                return false;  
37            }  
38        }  
39        return true;  
40    }  
41 }
```

Figure 4: Prime Number

Explanation of the code:

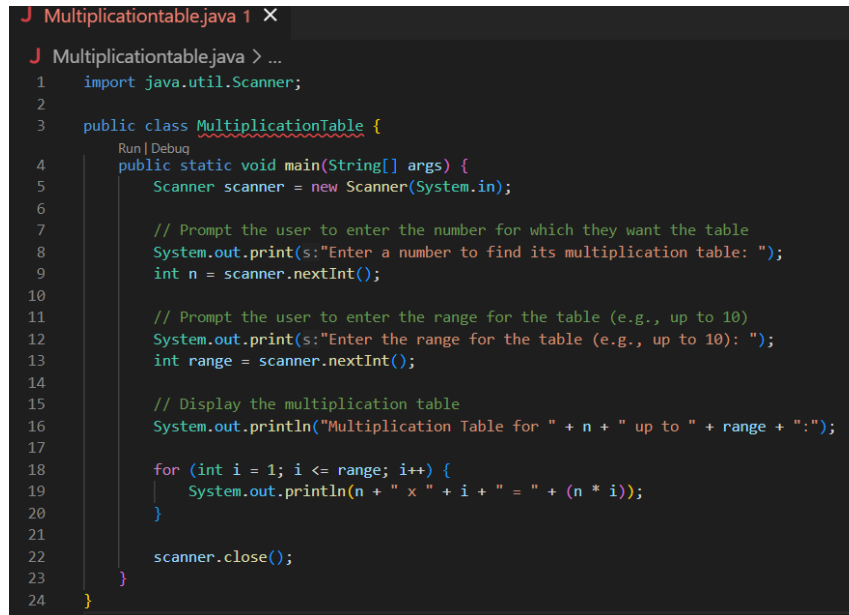
- **import java.util.Scanner:** This line imports the Java.util.Scanner class, which is used to receive user input.
- **primeNumberCheck Class:** This class has the functionality to determine if a given number is a prime number.
- **main Method:** The program's entry point is the main method. It performs the following:
  - to read user input from the console, creates the Scanner object with the name scanner.
  - requests a number from the user.
  - scans the entered number to read it and saves it in the variable num using nextInt().
  - determines whether num is a prime number by calling the isPrime function.
  - depending on whether num is a prime number, prints the outcome.
- **is Prime Method:** The return value of this method, which accepts an integer num as an input, is a boolean value (true if the number is a prime, false otherwise). To determine if a number is prime, the following procedures are taken:
  - Given that prime numbers are more than 1, the function returns false if num is less than or equal to 1.
  - As they are prime numbers, if num is 2 or 3, it returns true.
  - Since prime numbers have only two divisors: 1 and themselves, if num is divisible by 2 or 3, it returns false.
  - To optimise the algorithm, the loop from  $i = 5$  to  $i * i = \text{num}$  iterates through odd values (skipping even ones). It determines if num is divisible by  $i$  or  $i + 2$  within the loop. The procedure returns false if each of these circumstances holds true, indicating that num is not prime.
  - The method returns true, indicating that num is prime, if none of the following requirements are satisfied during the loop.
- Overall, the programme uses an algorithm that is optimised to eliminate unnecessary divisibility checks for some integers while determining if a given number is prime or not.

**Q21. Write a java program to find table of n.**

**Ans.** `import java.util.Scanner;`

```
public class MultiplicationTable {  
    public static void main(String[] args) {  
        Scanner scanner = new Scanner(System.in);  
  
        // Prompt the user to enter the number for which they want the table  
        System.out.print("Enter a number to find its multiplication table: ");  
        int n = scanner.nextInt();  
  
        // Prompt the user to enter the range for the table (e.g., up to 10)  
        System.out.print("Enter the range for the table (e.g., up to 10): ");  
        int range = scanner.nextInt();  
  
        // Display the multiplication table  
        System.out.println("Multiplication Table for " + n + " up to " + range + ":");  
  
        for (int i = 1; i <= range; i++) {  
            System.out.println(n + " x " + i + " = " + (n * i));  
        }  
  
        scanner.close();  
    }  
}
```

}



```
J Multiplicationtable.java 1 X
J Multiplicationtable.java > ...
1  import java.util.Scanner;
2
3  public class MultiplicationTable {
4      Run | Debug
      public static void main(String[] args) {
5          Scanner scanner = new Scanner(System.in);
6
7          // Prompt the user to enter the number for which they want the table
8          System.out.print(s:"Enter a number to find its multiplication table: ");
9          int n = scanner.nextInt();
10
11         // Prompt the user to enter the range for the table (e.g., up to 10)
12         System.out.print(s:"Enter the range for the table (e.g., up to 10): ");
13         int range = scanner.nextInt();
14
15         // Display the multiplication table
16         System.out.println("Multiplication Table for " + n + " up to " + range + ":");
17
18         for (int i = 1; i <= range; i++) {
19             System.out.println(n + " x " + i + " = " + (n * i));
20         }
21
22         scanner.close();
23     }
24 }
25
```

Figure 5: Table of number (n)

Explanation of the code:

- **Import the Scanner Class :** For allowing user input, the programme begins by importing the java.util.Scanner class.
- **Declare the Main Class :** The main class, named MultiplicationTable, is declared.
- **Define the Main Method :** The main method, its programme entry point, is defined.
- **Create a Scanner Object :** To read user input, the object scanner is constructed.
- **Prompt for User Input – Number :** The user is prompted by the programme to input the number they want to utilise System.out to get the multiplication table for.print("Enter a number to find its multiplication table:");.
- **Read and Store the Entered Number :** Scanner reads the input number as an integer.and kept in the variable n after nextInt().
- **Prompt for User Input – Range :** System.out.print("Enter the range for the table (e.g., up to 10):"); is used by the programme to prompt the user to input the multiplication table's range (e.g., up to 10).
- **Read and Store the Entered Range :** Scanner reads the entered range as an integer.nextInt() and kept in the range of variables.
- **Display Table Header :** With the help of System.out.println("Multiplication Table for " + n + " up to " + range + ":");, the programme displays a header for the multiplication table.

- **Generate and Display the Multiplication Table :**
  - In order to create and show the multiplication table, the programme enters a for loop.
  - Set the value of the loop variable i to 1.
  - As long as i is below or equal to the specified range, the loop will continue.
  - Inside the loop :
    - Use  $(n * i)$  to calculate the product of n and i.
    - Use `System.out.println(n + " x " + i + " = " + (n * i));` to display the multiplication equation and its result.
    - Each time the loop is executed, i is increased.
- **Close the Scanner :** The programme ends the Scanner object using `scanner.close()` to release system resources utilised for user input after displaying the whole multiplication table.

The multiplication table for the entered number up to the range is generated and shown by this programme when it receives the user's input for both the number and the range. It iterates across the range using a for loop, computing the product of the current iteration value and the number, and showing each multiplication expression and result.

**Q22. Write a java program to find the largest of three numbers.**

**Ans.** import java.util.Scanner;

```
public class LargestOfThreeNumbers {  
    public static void main(String[] args) {  
        Scanner scanner = new Scanner(System.in);  
  
        // Prompt the user to enter the three numbers  
        System.out.print("Enter the first number: ");  
        double num1 = scanner.nextDouble();  
  
        System.out.print("Enter the second number: ");  
        double num2 = scanner.nextDouble();  
  
        System.out.print("Enter the third number: ");  
        double num3 = scanner.nextDouble();  
  
        // Find the largest number among the three  
        double largest = findLargest(num1, num2, num3);  
  
        // Display the largest number  
        System.out.println("The largest number is: " + largest);  
  
        scanner.close();  
    }  
}
```

```
}

// A method to find the largest number among three

public static double findLargest(double num1, double num2, double num3) {

    if (num1 >= num2 && num1 >= num3) {

        return num1;

    } else if (num2 >= num1 && num2 >= num3) {

        return num2;

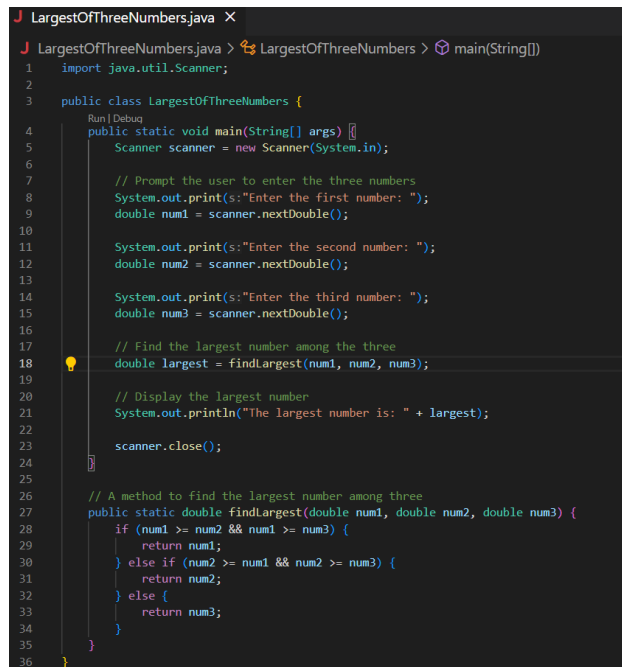
    } else {

        return num3;

    }

}

}
```



```
LargestOfThreeNumbers.java X
J LargestOfThreeNumbers.java > LargestOfThreeNumbers > main(String[])
1 import java.util.Scanner;
2
3 public class LargestOfThreeNumbers {
4     public static void main(String[] args) {
5         Scanner scanner = new Scanner(System.in);
6
7         // Prompt the user to enter the three numbers
8         System.out.print(s:"Enter the first number: ");
9         double num1 = scanner.nextDouble();
10
11         System.out.print(s:"Enter the second number: ");
12         double num2 = scanner.nextDouble();
13
14         System.out.print(s:"Enter the third number: ");
15         double num3 = scanner.nextDouble();
16
17         // Find the largest number among the three
18         double largest = findLargest(num1, num2, num3);
19
20         // Display the largest number
21         System.out.println("The largest number is: " + largest);
22
23         scanner.close();
24     }
25
26     // A method to find the largest number among three
27     public static double findLargest(double num1, double num2, double num3) {
28         if (num1 >= num2 && num1 >= num3) {
29             return num1;
30         } else if (num2 >= num1 && num2 >= num3) {
31             return num2;
32         } else {
33             return num3;
34         }
35     }
36 }
```

Figure 6: Largest of three numbers

Explanation of the code:

- **Import the Scanner Class :** To enable user input, the programme begins by importing the `java.util.Scanner` class.
- **Declare the Main Class :** The main class, named `LargestOfThreeNumbers`, is declared.
- **Define the Main Method :** The principal method, or programme entry point, is specified.
- **Create a Scanner Object :** To read user input, the object scanner is created.
- **Prompt for User Input - Three Numbers :**
  - The programme prompts the user to input three numbers sequentially:
  - `System.out.print("Enter the first number: ");` prompts the user to enter the first number.
  - `System.out.print("Enter the second number: ");` prompts the user to enter the second number.
  - `System.out.print("Enter the third number: ");` prompts the user to enter the third number.
- **Read and Store the Entered Numbers :** Using a scanner, the entered numbers are read as double values using the functions `nextDouble()` and `num1`, `num2`, and `num3` to store data.
- **Call the findLargest Method :** To determine which of the three entered numbers is the largest, the programme calls the `findLargest` function.
- **The findLargest Method :**
  - Below the main method is the definition of the `findLargest` method.
  - It requires the triple values of `num1`, `num2`, and `num3`.
  - Inside the process:
  - To discover which of the three numbers is the biggest conditional statements are used.
  - It returns `num1` as the largest if `num1` is greater than or equal to `num2` and `num1` is greater than or equal to `num3`.
  - It returns `num2` as the largest if `num2` is greater than or equal to `num1` and `num2` is greater than or equal to `num3`.
  - It provides `num3` as the greatest if neither of the above criteria is true.
- **Display the Largest Number :** `System.out.println("The largest number is: " + largest);` is used to display the greatest number found by the `findLargest` method.
- **Close the Scanner :** The programme uses `scanner.close()` to shut down the Scanner object after showing the biggest number in order to free up system resources used for user input.

In conclusion, this programme asks the user to input three integers, finds the largest one using the `findLargest` method, and then outputs the result. The greatest number is provided by the `findLargest` function after comparing the three.



**Q23. Write a java program to calculate Simple Interest.**

**Ans.** import java.util.Scanner; // Import the Scanner class from java.util package for user input

```
public class SimpleInterestCalculator { // Declare a class named SimpleInterestCalculator
```

```
    public static void main(String[] args) { // Define the main method
```

```
        Scanner scanner = new Scanner(System.in); // Create a Scanner object for input
```

```
        // Prompt the user to enter the principal amount
```

```
        System.out.print("Enter the principal amount: ");
```

```
        double principal = scanner.nextDouble(); // Read the entered principal amount as a double
```

```
        // Prompt the user to enter the annual interest rate (as a percentage)
```

```
        System.out.print("Enter the annual interest rate (as a percentage): ");
```

```
        double rate = scanner.nextDouble(); // Read the entered rate as a double
```

```
        // Prompt the user to enter the time in years
```

```
        System.out.print("Enter the time (in years): ");
```

```
        double time = scanner.nextDouble(); // Read the entered time as a double
```

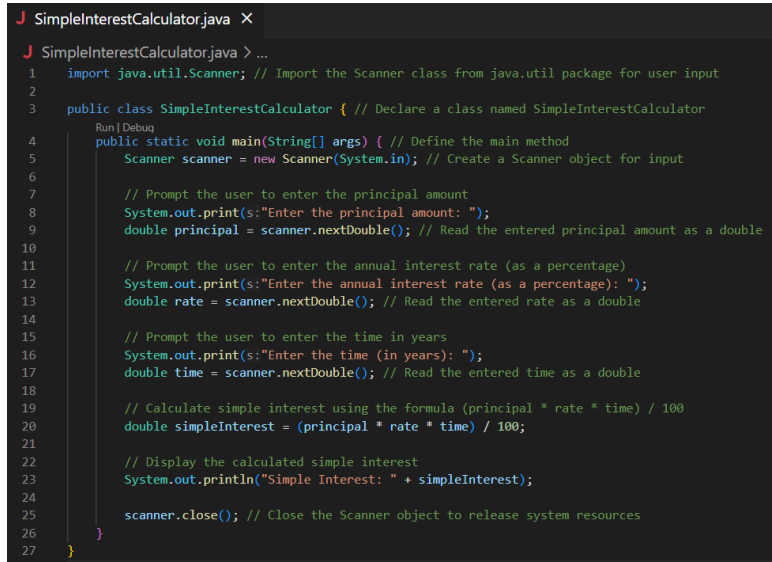
```
        // Calculate simple interest using the formula (principal * rate * time) / 100
```

```
        double simpleInterest = (principal * rate * time) / 100;
```

```
        // Display the calculated simple interest
```

```
        System.out.println("Simple Interest: " + simpleInterest);
```

```
scanner.close(); // Close the Scanner object to release system resources  
}  
}
```



```
J SimpleInterestCalculator.java X  
J SimpleInterestCalculator.java > ...  
1 import java.util.Scanner; // Import the Scanner class from java.util package for user input  
2  
3 public class SimpleInterestCalculator { // Declare a class named SimpleInterestCalculator  
4     public static void main(String[] args) { // Define the main method  
5         Scanner scanner = new Scanner(System.in); // Create a Scanner object for input  
6  
7         // Prompt the user to enter the principal amount  
8         System.out.print(s:"Enter the principal amount: ");  
9         double principal = scanner.nextDouble(); // Read the entered principal amount as a double  
10  
11         // Prompt the user to enter the annual interest rate (as a percentage)  
12         System.out.print(s:"Enter the annual interest rate (as a percentage): ");  
13         double rate = scanner.nextDouble(); // Read the entered rate as a double  
14  
15         // Prompt the user to enter the time in years  
16         System.out.print(s:"Enter the time (in years): ");  
17         double time = scanner.nextDouble(); // Read the entered time as a double  
18  
19         // Calculate simple interest using the formula (principal * rate * time) / 100  
20         double simpleInterest = (principal * rate * time) / 100;  
21  
22         // Display the calculated simple interest  
23         System.out.println("Simple Interest: " + simpleInterest);  
24  
25         scanner.close(); // Close the Scanner object to release system resources  
26     }  
27 }
```

Figure 7: Simple Interest

Explanation of the code:

- **Import the Scanner Class:** To enable user input, the programme begins by importing the `java.util.Scanner` class.
- **Declare the Main Class:** The main class, named `SimpleInterestCalculator`, is declared.
- **Define the Main Method:** The principal method, or programme entry point, is provided.
- **Create a Scanner Object:** To read user input, the object `scanner` is created.
- **Prompt for User Input - Principal Amount, Rate, and Time:**
  - The user is prompted by the programme to enter the following information:
  - `System.out.print("Enter the principal amount: ");` prompts the user to enter the principal amount.
  - `System.out.print("Enter the annual interest rate (as a percentage): ");` prompts the user to enter the annual interest rate (as a percentage).
  - `System.out.print("Enter the time (in years): ");` prompts the user to enter the time in years.

- **Read and Store the Entered Values:** Principal, rate, and time values are read as double values using a scanner. and the variables primary, rate, and time were saved by using nextDouble().
- **Calculate Simple Interest:** Simple interest is calculated and stored in the variable simpleInterest using the formula  $(\text{principal} * \text{rate} * \text{time}) / 100$ .
- **Display the Calculated Simple Interest:** System.out.println("Simple Interest: " + simpleInterest); is used in the programme to display the calculated simple interest.
- **Close the Scanner:** The programme closes the Scanner object using scanner after presenting the results. Release system resources used for user input by using close().

Based on principal, rate, and time inputs from the user, this programme calculates the simple interest and shows the result. The formula for calculating simple interest is  $(\text{principal} * \text{rate} * \text{time}) / 100$ .

**Q24. Write a java program to calculate Area and perimeter of Rectangle.**

**Ans.** import java.util.Scanner; // Import the Scanner class from java.util package for user input

```
public class RectangleCalculator { // Declare a class named RectangleCalculator

    public static void main(String[] args) { // Define the main method

        Scanner scanner = new Scanner(System.in); // Create a Scanner object for input


        // Prompt the user to enter the length of the rectangle
        System.out.print("Enter the length of the rectangle: ");

        double length = scanner.nextDouble(); // Read the entered length as a double


        // Prompt the user to enter the width of the rectangle
        System.out.print("Enter the width of the rectangle: ");

        double width = scanner.nextDouble(); // Read the entered width as a double


        // Calculate the area of the rectangle using the calculateArea method
        double area = calculateArea(length, width);

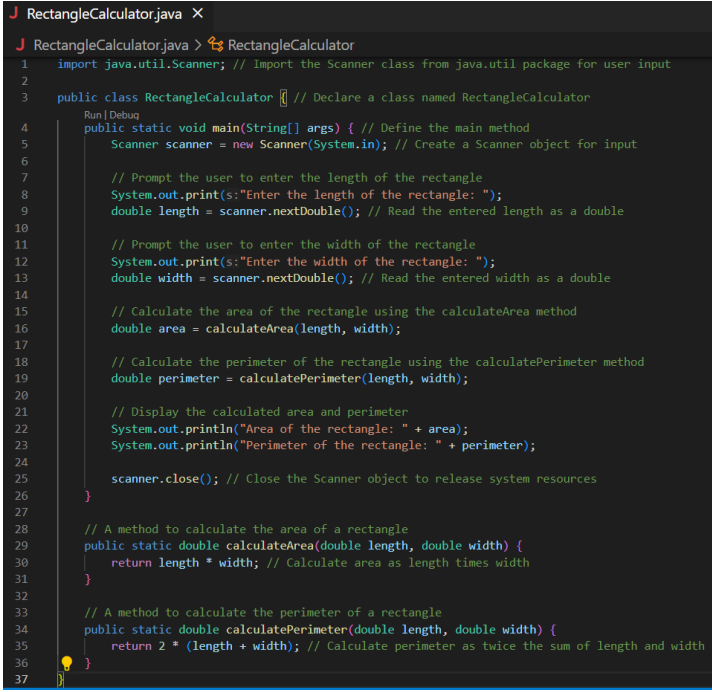

        // Calculate the perimeter of the rectangle using the calculatePerimeter method
        double perimeter = calculatePerimeter(length, width);


        // Display the calculated area and perimeter
        System.out.println("Area of the rectangle: " + area);
        System.out.println("Perimeter of the rectangle: " + perimeter);
    }
}
```

```
scanner.close(); // Close the Scanner object to release system resources
}

// A method to calculate the area of a rectangle
public static double calculateArea(double length, double width) {
    return length * width; // Calculate area as length times width
}

// A method to calculate the perimeter of a rectangle
public static double calculatePerimeter(double length, double width) {
    return 2 * (length + width); // Calculate perimeter as twice the sum of length and width
}
}
```



```
J RectangleCalculator.java X
J RectangleCalculator.java > RectangleCalculator
1 import java.util.Scanner; // Import the Scanner class from java.util package for user input
2
3 public class RectangleCalculator { // Declare a class named RectangleCalculator
4     public static void main(String[] args) { // Define the main method
5         Scanner scanner = new Scanner(System.in); // Create a Scanner object for input
6
7         // Prompt the user to enter the length of the rectangle
8         System.out.print("Enter the length of the rectangle: ");
9         double length = scanner.nextDouble(); // Read the entered length as a double
10
11        // Prompt the user to enter the width of the rectangle
12        System.out.print("Enter the width of the rectangle: ");
13        double width = scanner.nextDouble(); // Read the entered width as a double
14
15        // Calculate the area of the rectangle using the calculateArea method
16        double area = calculateArea(length, width);
17
18        // Calculate the perimeter of the rectangle using the calculatePerimeter method
19        double perimeter = calculatePerimeter(length, width);
20
21        // Display the calculated area and perimeter
22        System.out.println("Area of the rectangle: " + area);
23        System.out.println("Perimeter of the rectangle: " + perimeter);
24
25        scanner.close(); // Close the Scanner object to release system resources
26    }
27
28    // A method to calculate the area of a rectangle
29    public static double calculateArea(double length, double width) {
30        return length * width; // Calculate area as length times width
31    }
32
33    // A method to calculate the perimeter of a rectangle
34    public static double calculatePerimeter(double length, double width) {
35        return 2 * (length + width); // Calculate perimeter as twice the sum of length and width
36    }
37 }
```

Figure 8: Area and Perimeter of Rectangle

Explanation of the code:

- **Import the Scanner Class:** To enable user input, the programme begins by importing the `java.util.Scanner` class.
- **Declare the Main Class:** The main class, named `RectangleCalculator`, is declared.
- **Define the Main Method:** The principal method, or programme entry point, is defined.
- **Create a Scanner Object:** To read user input, the object scanner is created.
- **Prompt for User Input - Length and Width:**
  - The programme prompts the user to input the rectangle's length and width:
  - `System.out.print("Enter the length of the rectangle: ");` prompts the user for the length.
  - `System.out.print("Enter the width of the rectangle: ");` prompts the user for the width.
- **Read and Store the Entered Values:** The length and width input data are read as double numbers using a scanner. length and width were saved in the variables `nextDouble()`.
- **Calculate Area and Perimeter:**
  - To make calculations, we use the separate methods `calculateArea` and `calculatePerimeter`.
  - The parameters length and width are sent to `calculateArea`, which returns the area (`length * width`).
  - The parameters length and width are also used by `calculatePerimeter`, which returns the perimeter as `2 * (length + width)`.
- **Display the Calculated Area and Perimeter:** Using `System.out.println`, the programme displays the calculated area and perimeter.
- **Close the Scanner:** The programme uses `scanner.close()` to close down the Scanner object after showing the findings in order to free up system resources used for user input.

Based on length and width inputs from the user, this programme determines the area and perimeter of a rectangle. To increase the readability and organisation of the code, it uses different methods for area and perimeter calculations.

**Q25. Write a java program to check whether character is vowel or consonant.**

**Ans.**     import java.util.Scanner; // Import the Scanner class from java.util package for user input

```
public class VowelConsonantChecker { // Declare a class named VowelConsonantChecker
```

```
    public static void main(String[] args) { // Define the main method
```

```
        Scanner scanner = new Scanner(System.in); // Create a Scanner object for input
```

```
        // Prompt the user to enter a character
```

```
        System.out.print("Enter a character: ");
```

```
        char ch = scanner.next().charAt(0); // Read the entered character
```

```
        // Check if the character is a vowel or a consonant using the isVowel method
```

```
        if (isVowel(ch)) {
```

```
            System.out.println(ch + " is a vowel.");
```

```
        } else {
```

```
            System.out.println(ch + " is a consonant.");
```

```
        }
```

```
        scanner.close(); // Close the Scanner object to release system resources
```

```
    }
```

```
    // A method to check if a character is a vowel
```

```
    public static boolean isVowel(char ch) {
```

```
        // Convert the character to lowercase for case-insensitive comparison
```

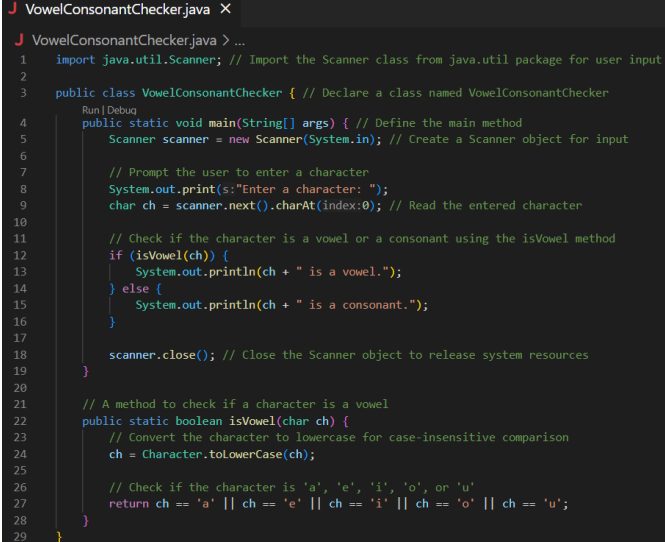
```
ch = Character.toLowerCase(ch);

// Check if the character is 'a', 'e', 'i', 'o', or 'u'

return ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u';

}

}
```



```
J VowelConsonantChecker.java X
J VowelConsonantChecker.java > ...
1 import java.util.Scanner; // Import the Scanner class from java.util package for user input
2
3 public class VowelConsonantChecker { // Declare a class named VowelConsonantChecker
4     Run/Debug
5     public static void main(String[] args) { // Define the main method
6         Scanner scanner = new Scanner(System.in); // Create a Scanner object for input
7
8         // Prompt the user to enter a character
9         System.out.print("Enter a character: ");
10        char ch = scanner.next().charAt(index:0); // Read the entered character
11
12        // Check if the character is a vowel or a consonant using the isVowel method
13        if (isVowel(ch)) {
14            System.out.println(ch + " is a vowel.");
15        } else {
16            System.out.println(ch + " is a consonant.");
17        }
18
19        scanner.close(); // Close the Scanner object to release system resources
20    }
21
22    // A method to check if a character is a vowel
23    public static boolean isVowel(char ch) {
24        // Convert the character to lowercase for case-insensitive comparison
25        ch = Character.toLowerCase(ch);
26
27        // Check if the character is 'a', 'e', 'i', 'o', or 'u'
28        return ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u';
29    }
30 }
```

Figure 9: Vowel or Consonant

Explanation of the code:

- **Import the Scanner Class:** To enable user input, the programme begins by importing the `java.util.Scanner` class.
- **Declare the Main Class:** The main class, named `VowelConsonantChecker`, is declared.
- **Define the Main Method:** The principal method, or programme entry point, is defined.
- **Create a Scanner Object:** To read user input, the object `scanner` is created.
- **Prompt for User Input - Character:** When using `System`, the programme prompts the user to input a character. `out."Enter a character:"` is printed.
- **Read and Store the Entered Character:** `Scanner` reads the input character as a string. `next()`, followed by `charAt(0)` to turn it into a character. It is kept in the `ch` variable.
- **Check if the Character is a Vowel:**
  - To determine whether the letter is a vowel, we call the `isVowel` method.
  - For case-insensitive comparison, the `isVowel` method accepts the character as an input and lowercases it using `Character.toLowerCase(ch)`.



- The character must be a 'a', 'e', 'i', 'o', or 'u' in order for the function to return true, indicating that the character is a vowel.
- **Display the Result:** Based on the result of the isVowel method, the programme indicates whether the entered letter is a vowel or a consonant.
- **Close the Scanner:** The programme uses scanner.close() to close down the Scanner object after showing the results in order to free up system resources used for user input.

This programme determines whether the letter input is a vowel (a, e, i, o, or u) and informs the user if it is. In order to handle both uppercase and lowercase letters, it also does comparisons that are not case-sensitive.