FLIP ROBO

FULL STACK DEVELOPMENT: WORKSHEET – 3

FLIP ROBO

Q1. Which one of the following is not a Java feature?

A. Object-oriented

B. Use of pointers

C. Portable

D. Dynamic and Extensible

Ans. B. Use of pointers

Java differs from languages like C and C++ in a significant way by not supporting the explicit usage of pointers. Java manages memory and manipulates objects using references rather than pointers.

Q2. Which of these cannot be used for a variable name in Java?

A. identifier & keyword

B. identifier

C. keyword

D. none of the mentioned

Ans. C. keyword

Java uses reserved words called keywords that have a particular function and meaning. Keywords cannot be used as variable names.

Q3. Which of the following is a superclass of every class in Java?

A. ArrayList

B. Abstract class

C. Object class

D. String

Ans. C. Object class

The Object class in Java serves as the base class for all other classes. Every class in Java derives in some way from the Object class.

Q4.Which one is a valid declaration of a boolean?

A. boolean b1 = 1;

B. boolean b2 = ‘false’;

C. boolean b3 = false;

D. boolean b4 = ‘true’

Ans. C. boolean b3 = false;

Boolean variables in Java can only be given the values true or false. Declaring a boolean variable and initialising it with false is the correct course of action in Option C.

Q5. Which is the modifier when there is none mentioned explicitly?

A. protected

B. private

C. public

D. default

Ans. D. default

Java uses the default access modifier when no access modifier is supplied. This indicates that the member—variable, method, or class—is only usable inside the confines of its own package.

Q6.All the variables of interface should be?

A. default and final

B. default and static

C. public, static and final

D. protect, static and final

Ans. C. public, static, and final

All variables in Java interfaces are by default public, static, and final. They are open to the public since interfaces are designed to specify a class's public API. The variables must be accessed at the class level, hence they are static since interfaces cannot be instantiated. Additionally, they are final because once they have been initialised in the interface, their values cannot be modified.

Q7.Which of these data types is used to store command line arguments?

A. Array

B. Stack

C. String

D. Integer

Ans. A. Array

Java stores command-line variables as strings in an array that is provided to the class's main function.

Q8.How many arguments can be passed to main()?

A. Infinite

B. Only 1

C. System Dependent

D. None of the mentioned

Ans. B. Only 1

An array of strings is a valid argument for Java's main function. When running a Java programme from the command line, you can pass a string array containing any number of variables, including zero.

Q9. What will be the output of the following Java program, Command line execution is done as – "java Output This is a command Line"?

```
class Output
{
   public static void main(String args[])
  {
      System.out.print(args[0]);
  }
}
```

A. java

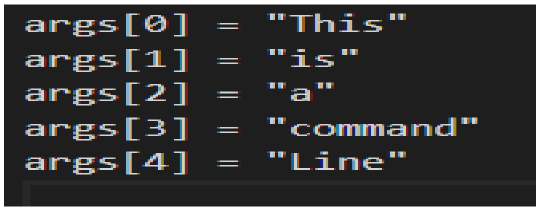
B. Output

C. This

D. is

Ans. A. java

When the Java program is executed with the command line argument "This is a command Line", the args array will contain the following elements:



```
args[0] = "This"
args[1] = "is"
args[2] = "a"
args[3] = "command"
args[4] = "Line"
```

Figure 1

The System.out.print(args[0]); statement prints the first element of the args array, which is "This".

Q10.What is the value of "d" in the following Java code snippet?

double d = Math.round (2.5 + Math.random());

A. 2

B. 3

C. 4

D. 2.5

Ans. B. 3

A random decimal value between 2.5 and 3.5 is generated using the equation Math.round(2.5 + Math.random()), which rounds it to the closest integer. Since there is an equal chance that the random number will fall between 2.5 and 3.5, the outcome of rounding will be 3. However, because the variable d is a double, it may contain decimal numbers.

So, the correct answer is: 3

Q11.Which of these methods is a rounding function of Math class?

A. max()

B. min()

C. abs()

D. None of the mentioned

Ans. None of the mentioned

None of the methods (max(), min(), or abs()) stated in the available choices are rounding operations of the Java Math class.

- The bigger of two numbers is returned by max().
- The lesser of two numbers is returned by min().
- The function abs() returns a number's absolute value.

Math.round() in the Math class may be used to implement a rounding function. The closest long or int value to the argument is returned by math.round(). Math.round(2.5), for instance, would yield 3, but Math.round(2.4) would return 2.

Q12. Standard output variable 'out' is defined in which class?

A. Void

B. Process

C. Runtime

D. System

Ans. D. System

The System class of the Java language defines the standard output variable out.

Q13.What will be the output of the following Java program?

```
class main_class
{
   public static void main(String args[])
  {
      int x = 9;
      if (x == 9)
     {
        int x = 8;
        System.out.println(x);
      }
  }
}
```

A. 9

B. 8

C. Compilation error

D. Runtime error

Ans. B. 8

There are two variables called x in the sample code. Within the confines of the if block, the inner x variable that was declared inside the block shadows the outside x variable. The inner x variable with the value 8 is referred to when the word "x" is written inside the if block.

Q14.Which of these is the method which is executed first before execution of any other thing takes place in a program?

A. main method

B. static method

C. private method

D. finalize method

Ans. A. main method

The main method of a Java programme serves as its entry point. It is the method that is used as the program's beginning point since it is the one that runs initially when the programme is run.

Q15.Which of these can be used to differentiate two or more methods having the same name?

A. Parameters data type

B. Number of parameters

C. Return type of method

D. All of the mentioned

Ans. D. All of the mentioned

Method overloading in Java enables the definition of multiple methods with the same name in the same class, each of which may be distinguished by its argument list. The number of parameters and the parameter types are two examples of these variations. Additionally, only the method name and the data types and number of the arguments are taken into account when overloading methods; the return type of the method does not distinguish methods.

Q16. What will be the output of the following Java program?

```
 class Output
{
  public static void main(String[] args)
 {
     int x , y = 1;
     x = 10;
     if(x != 10 && x / 0 == 0)
       System.out.println(y);
     else
       System.out.println(++y);
 }
}
```

A. 1

B. 2

C. Runtime Error

D. Compilation Error

Ans. B. 2

The if condition in the sample Java programme uses the logical AND operator (&&), which requires that both of the criteria on each side of the operator be true in order for the code within the if block to be executed.

Let's analyze the conditions:

i) x!= 10 - This test determines whether or not x is equal to 10. Since x in this instance is 10, this condition is false.

ii) The formula x / 0 == 0 attempts to divide x by 0. Since division by zero is not specified in either mathematics nor Java, this circumstance will result in an ArithmeticException runtime error and an irregular programme termination.

The if block won't be performed since the first condition is false and the second one results in a runtime error. The else block, which increases the value of y (++y) and prints the new value, will be executed.

As a result, the following will be the output of the Java programme: B. 2

Q17.What will be the output of the following Java program?

```
 class area
{
  int width;
  int length;
  int height;
  area()
 {
    width = 5;
    length = 6;
    height = 1;
    }
    void volume()
   {
         volume = width * height * length;
   }
}
class cons_method
{
     public static void main(String args[])
    {
            area obj = new area();

            obj.volume();

            System.out.println(obj.volume);

        }

    }
```

A. 0

B. 1

C. 25

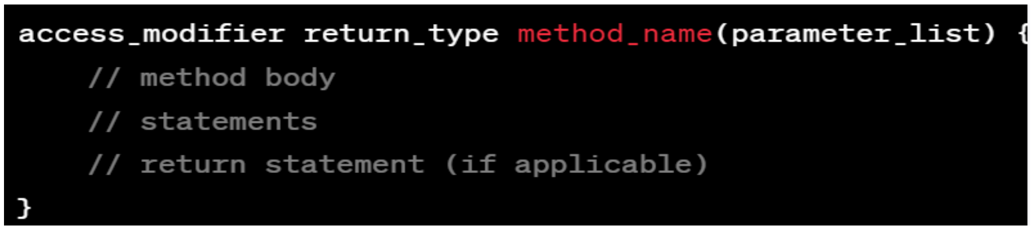
D. 30

Ans. D. 30

The volume of the item is computed as 5 * 1 * 6, which equals 30, hence the result will be 30 (option D).

Q18. Write Syntax to create/define java methods.

Ans. In Java, classes include methods that may be defined. Here is the fundamental syntax for defining and creating Java methods:



```
access_modifier return_type method_name(parameter_list) {
    // method body
    // statements
    // return statement (if applicable)
}
```

Figure 2

Let me break down the syntax for you:

i) **Access Modifier :** It describes the method's visibility. Java has four different categories of access modifiers:
- **public :** Any other class is able to access the method.
- **protected :** Within its own class, its subclasses, and classes belonging to the same package, the method can be accessed.
- **default (no modifier) :** Only classes in the same package and its own class can access the function.
- **private :** Only other classes in the same class can access the method.

ii) **Return Type :** It details the value's data type, which is what the method returns. The void keyword can be used if the method returns nothing.

iii) **Method Name :** It serves as the method's identification. The method name has to conform to the Java naming rules.

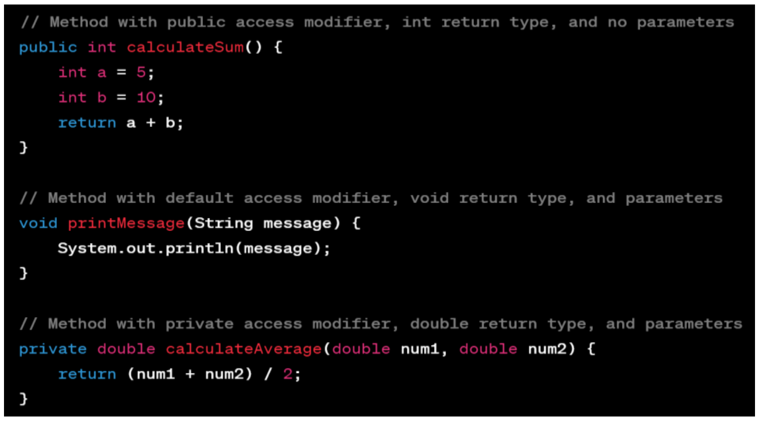
iv) **Parameter List :** It consists of a list of input parameters separated by commas and contained in brackets. Leave the brackets empty if the method doesn't take any parameters.

v) **Method Body :** The sentences that describe what the method performs are contained there. Curly braces {} surround the method body in code.

vi) **Return Statement (if applicable):** The method must return a value of the given type if it has a return type other than void. The value is returned using the return statement. You can skip the return statement or use it without a value to leave the method if it is of type void.

Here are a few examples of Java methods that use various return types and access modifiers:

```
// Method with public access modifier, int return type, and no parameters
public int calculateSum() {
    int a = 5;
    int b = 10;
    return a + b;
}

// Method with default access modifier, void return type, and parameters
void printMessage(String message) {
    System.out.println(message);
}

// Method with private access modifier, double return type, and parameters
private double calculateAverage(double num1, double num2) {
    return (num1 + num2) / 2;
}
```



```
// Method with public access modifier, int return type, and no parameters
public int calculateSum() {
    int a = 5;
    int b = 10;
    return a + b;
}

// Method with default access modifier, void return type, and parameters
void printMessage(String message) {
    System.out.println(message);
}

// Method with private access modifier, double return type, and parameters
private double calculateAverage(double num1, double num2) {
    return (num1 + num2) / 2;
}
```

Figure 3

Remember in mind that in Java, methods are defined within classes. When calling the methods, class objects or, if they are static methods, the class name itself can be used.

Q19. Write a java program following instructions

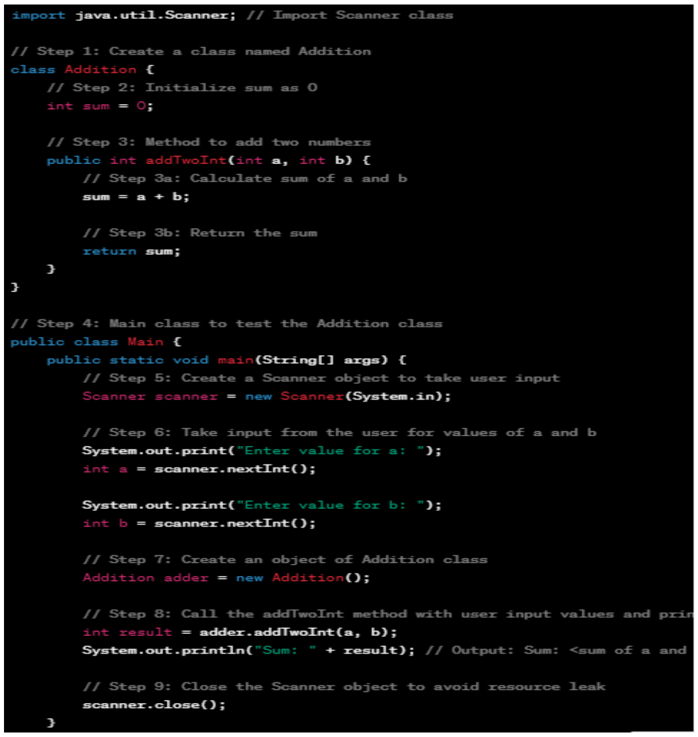
A. Make a class Addition

a. initialize sum as 0

b. make addTwoInt method taking two int parameters a,b. make sum = a+b. Return Sum

Ans. Here's a Java program that follows the given instructions:

```
import java.util.Scanner; // Import Scanner class


// Step 1: Create a class named Addition
class Addition {
    // Step 2: Initialize sum as 0
    int sum = 0;


    // Step 3: Method to add two numbers
    public int addTwoInt(int a, int b) {
        // Step 3a: Calculate sum of a and b
        sum = a + b;

        // Step 3b: Return the sum
        return sum;
    }
}

// Step 4: Main class to test the Addition class
public class Main {
    public static void main(String[] args) {
        // Step 5: Create a Scanner object to take user input
        Scanner scanner = new Scanner(System.in);

        // Step 6: Take input from the user for values of a and b
        System.out.print("Enter value for a: ");
        int a = scanner.nextInt();

        System.out.print("Enter value for b: ");
        int b = scanner.nextInt();

        // Step 7: Create an object of Addition class
        Addition adder = new Addition();
        // Step 8: Call the addTwoInt method with user input values and print the result

        int result = adder.addTwoInt(a, b);

        System.out.println("Sum: " + result); // Output: Sum: <sum of a and b>


        // Step 9: Close the Scanner object to avoid resource leak

        scanner.close();

    }

}
```



```
import java.util.Scanner; // Import Scanner class

// Step 1: Create a class named Addition
class Addition {
    // Step 2: Initialize sum as 0
    int sum = 0;

    // Step 3: Method to add two numbers
    public int addTwoInt(int a, int b) {
        // Step 3a: Calculate sum of a and b
        sum = a + b;

        // Step 3b: Return the sum
        return sum;
    }
}

// Step 4: Main class to test the Addition class
public class Main {
    public static void main(String[] args) {
        // Step 5: Create a Scanner object to take user input
        Scanner scanner = new Scanner(System.in);

        // Step 6: Take input from the user for values of a and b
        System.out.print("Enter value for a: ");
        int a = scanner.nextInt();

        System.out.print("Enter value for b: ");
        int b = scanner.nextInt();

        // Step 7: Create an object of Addition class
        Addition adder = new Addition();

        // Step 8: Call the addTwoInt method with user input values and prin
        int result = adder.addTwoInt(a, b);
        System.out.println("Sum: " + result); // Output: Sum: <sum of a and

        // Step 9: Close the Scanner object to avoid resource leak
        scanner.close();
    }
```

Figure 4

The Scanner class is used in this version of the programme to collect user-provided input for the values of a and b. Integers submitted by the user are read using the nextInt() function of Scanner. The programme then computes and outputs the sum of variables a and b. To stop resource leakage, the Scanner object is then shut off.

B. define class as Method Call. Define main method

a. Create object of class Addition

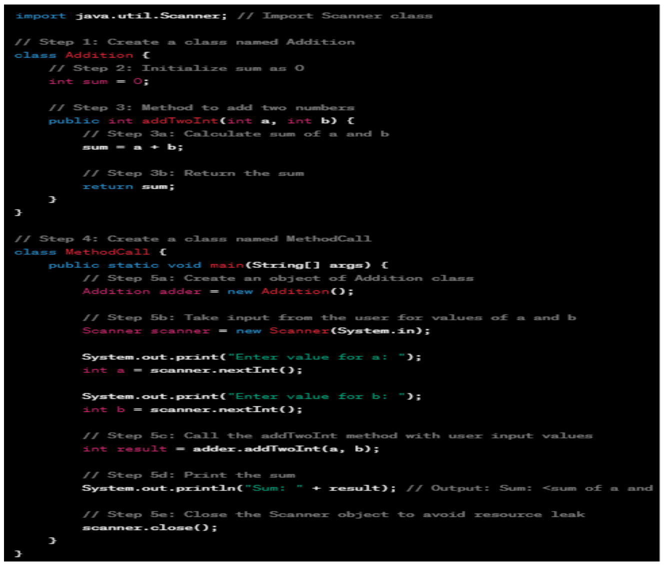
b. call method using instance of object

c. Print sum

Ans. Here's the updated version of the program following the new instructions:

```
import java.util.Scanner; // Import Scanner class


// Step 1: Create a class named Addition
class Addition {
    // Step 2: Initialize sum as 0
    int sum = 0;


    // Step 3: Method to add two numbers
    public int addTwoInt(int a, int b) {
        // Step 3a: Calculate sum of a and b
        sum = a + b;


        // Step 3b: Return the sum
        return sum;
    }
}


// Step 4: Create a class named MethodCall
```

```
class MethodCall {
    public static void main(String[] args) {
        // Step 5a: Create an object of Addition class
        Addition adder = new Addition();

        // Step 5b: Take input from the user for values of a and b
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter value for a: ");
        int a = scanner.nextInt();

        System.out.print("Enter value for b: ");
        int b = scanner.nextInt();

        // Step 5c: Call the addTwoInt method with user input values
        int result = adder.addTwoInt(a, b);

        // Step 5d: Print the sum
        System.out.println("Sum: " + result); // Output: Sum: <sum of a and b>

        // Step 5e: Close the Scanner object to avoid resource leak
        scanner.close();
    }
}
```



```
import java.util.Scanner; // Import Scanner class

// Step 1: Create a class named Addition
class Addition {
    // Step 2: Initialize sum as 0
    int sum = 0;

    // Step 3: Method to add two numbers
    public int addTwoInt(int a, int b) {
        // Step 3a: Calculate sum of a and b
        sum = a + b;

        // Step 3b: Return the sum
        return sum;
    }
}

// Step 4: Create a class named MethodCall
class MethodCall {
    public static void main(String[] args) {
        // Step 5a: Create an object of Addition class
        Addition adder = new Addition();

        // Step 5b: Take input from the user for values of a and b
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter value for a: ");
        int a = scanner.nextInt();

        System.out.print("Enter value for b: ");
        int b = scanner.nextInt();

        // Step 5c: Call the addTwoInt method with user input values
        int result = adder.addTwoInt(a, b);

        // Step 5d: Print the sum
        System.out.println("Sum: " + result); // Output: Sum: <sum of a and

        // Step 5e: Close the Scanner object to avoid resource leak
        scanner.close();
    }
}
```

Figure 5

In this program:

- The primary method, from which the programme launches its execution, is included in the MethodCall class.
- (Addition adder = new Addition();) creates an object of the Addition class.
- The Scanner class is used to collect user input for the variables a and b.
- With the user-inputted numbers, the addTwoInt method of the Addition class is invoked, and the sum is computed and saved in the result variable.
- The addTwoInt method's result, the total, is printed by the programme.

Q20. Write a java program following instructions

A. Define a class Example

a. Define two instance variables number and name

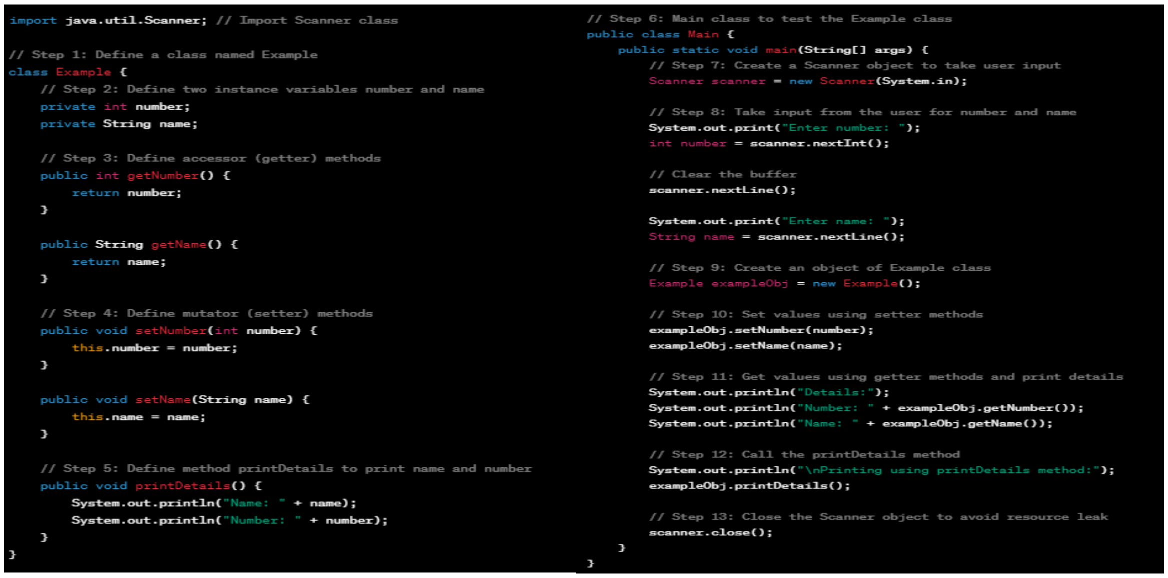
b. Define accessor (getter) methods

c. Define mutator (setter) methods

d. define method printDetails —-> print name and number

Ans. import java.util.Scanner; // Import Scanner class

```java
// Step 1: Define a class named Example
class Example {
    // Step 2: Define two instance variables number and name
    private int number;
    private String name;

    // Step 3: Define accessor (getter) methods
    public int getNumber() {
        return number;
    }

    public String getName() {
        return name;
    }

    // Step 4: Define mutator (setter) methods
    public void setNumber(int number) {
        this.number = number;
    }

    public void setName(String name) {
        this.name = name;
    }
    // Step 5: Define method printDetails to print name and number
    public void printDetails() {
        System.out.println("Name: " + name);
        System.out.println("Number: " + number);
    }
}

// Step 6: Main class to test the Example class
public class Main {
    public static void main(String[] args) {
        // Step 7: Create a Scanner object to take user input
        Scanner scanner = new Scanner(System.in);

        // Step 8: Take input from the user for number and name
        System.out.print("Enter number: ");
        int number = scanner.nextInt();

        // Clear the buffer
        scanner.nextLine();

        System.out.print("Enter name: ");
        String name = scanner.nextLine();

        // Step 9: Create an object of Example class
        Example exampleObj = new Example();
        // Step 10: Set values using setter methods
        exampleObj.setNumber(number);
        exampleObj.setName(name);

        // Step 11: Get values using getter methods and print details
        System.out.println("Details:");
        System.out.println("Number: " + exampleObj.getNumber());
        System.out.println("Name: " + exampleObj.getName());

        // Step 12: Call the printDetails method
        System.out.println("\nPrinting using printDetails method:");
        exampleObj.printDetails();

        // Step 13: Close the Scanner object to avoid resource leak
        scanner.close();
    }
}
```



```java
import java.util.Scanner; // Import Scanner class

// Step 1: Define a class named Example
class Example {
    // Step 2: Define two instance variables number and name
    private int number;
    private String name;

    // Step 3: Define accessor (getter) methods
    public int getNumber() {
        return number;
    }

    public String getName() {
        return name;
    }

    // Step 4: Define mutator (setter) methods
    public void setNumber(int number) {
        this.number = number;
    }

    public void setName(String name) {
        this.name = name;
    }

    // Step 5: Define method printDetails to print name and number
    public void printDetails() {
        System.out.println("Name: " + name);
        System.out.println("Number: " + number);
    }
}
```

```java
// Step 6: Main class to test the Example class
public class Main {
    public static void main(String[] args) {
        // Step 7: Create a Scanner object to take user input
        Scanner scanner = new Scanner(System.in);

        // Step 8: Take input from the user for number and name
        System.out.print("Enter number: ");
        int number = scanner.nextInt();

        // Clear the buffer
        scanner.nextLine();

        System.out.print("Enter name: ");
        String name = scanner.nextLine();

        // Step 9: Create an object of Example class
        Example exampleObj = new Example();

        // Step 10: Set values using setter methods
        exampleObj.setNumber(number);
        exampleObj.setName(name);

        // Step 11: Get values using getter methods and print details
        System.out.println("Details:");
        System.out.println("Number: " + exampleObj.getNumber());
        System.out.println("Name: " + exampleObj.getName());

        // Step 12: Call the printDetails method
        System.out.println("\nPrinting using printDetails method:");
        exampleObj.printDetails();

        // Step 13: Close the Scanner object to avoid resource leak
        scanner.close();
    }
}
```

Figure 6

For the number and name variables in this revised version, the programme makes use of a Scanner object. The string input is read using the nextLine() function, while the integer input is read using the nextInt() method. The programme prints the information using both getter methods and the printDetails() function after setting the values using the setter methods. To prevent resource leaks, the Scanner object is finally shut off.

B. Define public class Demo (Main Class)

a. Define main method

b. Make Instance/object of example class

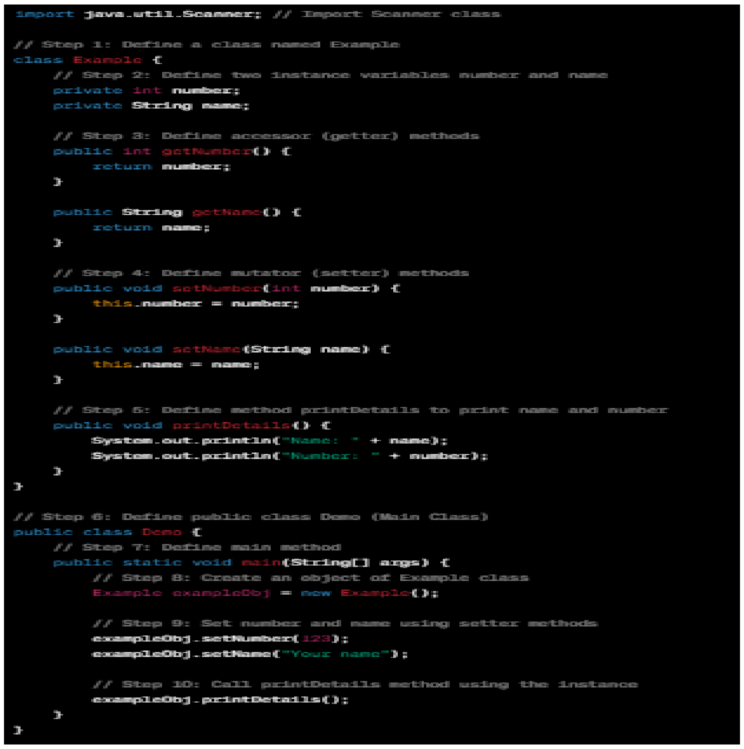
c. set number and name using instance created as 123 and Your name.

d. call printDetails method using instance

Ans. Here is the revised programme that uses the Demo class as its primary class and complies with the directives:

import java.util.Scanner; // Import Scanner class

```
// Step 1: Define a class named Example
class Example {
    // Step 2: Define two instance variables number and name
    private int number;
    private String name;

    // Step 3: Define accessor (getter) methods
    public int getNumber() {
        return number;
    }

    public String getName() {
        return name;
    }

    // Step 4: Define mutator (setter) methods
    public void setNumber(int number) {
        this.number = number;
    }

    public void setName(String name) {
        this.name = name;
    }
    // Step 5: Define method printDetails to print name and number
    public void printDetails() {
        System.out.println("Name: " + name);
        System.out.println("Number: " + number);
    }
}

// Step 6: Define public class Demo (Main Class)
public class Demo {
    // Step 7: Define main method
    public static void main(String[] args) {
        // Step 8: Create an object of Example class
        Example exampleObj = new Example();

        // Step 9: Set number and name using setter methods
        exampleObj.setNumber(123);
        exampleObj.setName("Your name");

        // Step 10: Call printDetails method using the instance
        exampleObj.printDetails();
    }
}
```



```
import java.util.Scanner; // Import Scanner class

// Step 1: Define a class named Example
class Example {
    // Step 2: Define two instance variables number and name
    private int number;
    private String name;

    // Step 3: Define accessor (getter) methods
    public int getNumber() {
        return number;
    }

    public String getName() {
        return name;
    }

    // Step 4: Define mutator (setter) methods
    public void setNumber(int number) {
        this.number = number;
    }

    public void setName(String name) {
        this.name = name;
    }

    // Step 5: Define method printDetails to print name and number
    public void printDetails() {
        System.out.println("Name: " + name);
        System.out.println("Number: " + number);
    }
}

// Step 6: Define public class Demo (Main Class)
public class Demo {
    // Step 7: Define main method
    public static void main(String[] args) {
        // Step 8: Create an object of Example class
        Example exampleObj = new Example();

        // Step 9: Set number and name using setter methods
        exampleObj.setNumber(123);
        exampleObj.setName("Your name");

        // Step 10: Call printDetails method using the instance
        exampleObj.printDetails();
    }
}
```

Figure 7

In this program:

- The main class containing the main function is referred to as the demo class.
- It creates an object of the Example class.
- The values of number and name are set using the setNumber(int number) and setName(String name) methods, respectively.
- To print the name and number, the printDetails() function is used using the instance of the Example class.