Why Java is different TO c++?

JAVA is not directly supporting to input the number values by the users. to perform this has same indirect methods

1. Object and Class

What is a Class?

A class in Java is a blueprint for creating objects. It defines the properties (fields) and behaviors (methods) that the objects created from the class will have.

What is an Object?

An object is an instance of a class. It represents a specific implementation of the class, with actual values assigned to the properties defined by the class.

Example:

```
java
CopyEdit
// Define a class
class Car {
    String color;
    int year;

    void drive() {
        System.out.println("The car is driving.");
    }
}

// Create an object
public class Main {
    public static void main(String[] args) {
```

```
        Car myCar = new Car(); // Create a Car object
        myCar.color = "Red";
        myCar.year = 2020;
        myCar.drive(); // Call the drive method
    }
}
```

In this example, `Car` is a class with properties `color` and `year`, and a

Example:

java Copy

```java
// Define a class
class Car {
    String color;
    int year;

    void drive() {
        System.out.println("The car is driving.");
    }
}

// Create an object
public class Main {
    public static void main(String[] args) {
        Car myCar = new Car(); // Create a Car object
        myCar.color = "Red";
        myCar.year = 2020;
        myCar.drive(); // Call the drive method
    }
}
```

In this example, `Car` is a class with properties `color` and `year`, and a method `drive()`. `myCar` is an of the `Car` class.

---

2. Using Methods (Functions)

What is a Method?

A method in Java is a block of code that performs a specific task. Methods are used to define the behaviors of objects and to organize code into reusable blocks.[Programiz+1GeeksforGeeks+1]

Example:

java
CopyEdit

```
public class Calculator {
    // Method to add two numbers
    int add(int a, int b) {
        return a + b;
    }

    // Method to subtract two numbers
    int subtract(int a, int b) {
        return a - b;
    }

    public static void main(String[] args) {
        Calculator calc = new Calculator();
        int sum = calc.add(5, 3);
        int difference = calc.subtract(5, 3);
        System.out.println("Sum: " + sum);
        System.out.println("Difference: " + difference);
    }
}
```

Here, add and subtract are methods that perform arithmetic

Example:

java

Copy Edit

```java
public class Calculator {
    // Method to add two numbers
    int add(int a, int b) {
        return a + b;
    }

    // Method to subtract two numbers
    int subtract(int a, int b) {
        return a - b;
    }

    public static void main(String[] args) {
        Calculator calc = new Calculator();
        int sum = calc.add(5, 3);
        int difference = calc.subtract(5, 3);
        System.out.println("Sum: " + sum);
        System.out.println("Difference: " + difference);
    }
}
```

---

3. Method Overloading

What is Method Overloading?

Method overloading allows multiple methods in the same class to have the same name but different parameters (number, type, or

order). It enables methods to handle different types or numbers of inputs.Great Learning+2GeeksforGeeks+2Programiz+2

Example:

```
java
CopyEdit
public class Display {
    void show(int a) {
        System.out.println("Integer: " + a);
    }

    void show(String b) {
        System.out.println("String: " + b);
    }

    public static void main(String[] args) {
        Display d = new Display();
        d.show(10);       // Calls the method with integer parameter
        d.show("Hello");  // Calls the method with string parameter
    }
}
```

In this example, the show method is overloaded to handle both

Example:

java

Copy Edit

```java
public class Display {
    void show(int a) {
        System.out.println("Integer: " + a);
    }

    void show(String b) {
        System.out.println("String: " + b);
    }

    public static void main(String[] args) {
        Display d = new Display();
        d.show(10);       // Calls the method with integer parameter
        d.show("Hello");  // Calls the method with string parameter
    }
}
```

In this example, the `show` method is overloaded to handle both integer and string inputs.

---

4. Constructors

What is a Constructor?

A constructor is a special method that is called when an object is instantiated. It initializes the object's properties. Constructors have the same name as the class and do not have a return type.[Programiz+1TutorialsPoint+1]

Example:

java
CopyEdit

```
public class Person {
    String name;
    int age;

    // Constructor
    Person(String personName, int personAge) {
        name = personName;
        age = personAge;
    }

    void displayInfo() {
        System.out.println("Name: " + name + ", Age: " + age);
    }

    public static void main(String[] args) {
        Person p = new Person("Alice", 30);
        p.displayInfo();
    }
}
```

Example:

java

```java
public class Person {
    String name;
    int age;

    // Constructor
    Person(String personName, int personAge) {
        name = personName;
        age = personAge;
    }

    void displayInfo() {
        System.out.println("Name: " + name + ", Age: " + age);
    }

    public static void main(String[] args) {
        Person p = new Person("Alice", 30);
        p.displayInfo();
    }
}
```

Here, the constructor `Person` initializes the `name` and `age` properties when a new `Person` object is created.

Example: A Method with Parameters

java Copy Edit

```java
public class Greeting {

    // Method with parameters
    void sayHello(String name, int age) {
        System.out.println("Hello, " + name + "! You are " + age + " years old.");
    }

    public static void main(String[] args) {
        Greeting g = new Greeting();

        // Passing parameters to the method
        g.sayHello("Fique", 22);
        g.sayHello("Lydivine", 20);
    }
}
```

Explanation:

- sayHello(String name, int age) is a method that takes **two parameters**: a String and an int.
- We **call the method** with actual values (called arguments), like "Fique" and 22.

You can modify this to accept different types or more parameters. If you want a version using **return values**, just let me know!

Great! Here's a full Java program that defines a **method with parameters** and calls it from the `main` meth

---

Example: Method and Parameters Together

java

Copy Edit

```java
public class Calculator {

    // Method with parameters that adds two numbers
    int add(int a, int b) {
        int sum = a + b;
        return sum;
    }

    public static void main(String[] args) {
        Calculator calc = new Calculator(); // Create object

        // Call the method with arguments
        int result = calc.add(10, 5);

        // Display the result
        System.out.println("The sum is: " + result);
    }
}
```

IF AND ELSE

Example 3: Check if a Number is Even or Odd

java

Copy Edit

```java
public class Practice3 {

    void checkEvenOdd(int number) {
        if (number % 2 == 0) {
            System.out.println(number + " is even.");
        } else {
            System.out.println(number + " is odd.");
        }
    }

    public static void main(String[] args) {
        Practice3 obj = new Practice3();
        obj.checkEvenOdd(7);
        obj.checkEvenOdd(12);
    }
}
```

IF AND ELSE and switch

java

```java
public class ConditionExample {
    public static void main(String[] args) {
        int number = -5;

        // if-else if-else
        if (number > 0) {
            System.out.println("Positive number");
        } else if (number < 0) {
            System.out.println("Negative number");
        } else {
            System.out.println("Zero");
        }

        // switch-case
        int day = 3;
        switch (day) {
            case 1:
                System.out.println("Monday");
                break;
            case 2:
                System.out.println("Tuesday");
                break;
            case 3:
                System.out.println("Wednesday");
                break;
            default:
                System.out.println("Invalid day");
        }
    }
}
```

2. Loops (for, while, do-while)

◆ **What it is:**

Used to repeat a block of code multiple times.

✅ Example:

```java
public class LoopExample {
    public static void main(String[] args) {
        // for loop
        System.out.println("For Loop:");
        for (int i = 1; i <= 3; i++) {
            System.out.println("i = " + i);
        }

        // while loop
        System.out.println("While Loop:");
        int j = 1;
        while (j <= 3) {
            System.out.println("j = " + j);
            j++;
        }

        // do-while loop
        System.out.println("Do-While Loop:");
        int k = 1;
        do {
            System.out.println("k = " + k);
            k++;
        } while (k <= 3);
    }
}
```

✓ 3. Arrays & Strings

◆ What it is:

- **Arrays** hold multiple values of the same type.
- **Strings** are text-based data.

✅ Example:

java

```java
public class ArrayStringExample {
    public static void main(String[] args) {
        // Array
        int[] scores = {85, 90, 78};
        System.out.println("First score: " + scores[0]); // Output: 85

        // String
        String name = "Java Programming";
        System.out.println("Length: " + name.length());
        System.out.println("Uppercase: " + name.toUpperCase());
        System.out.println("First letter: " + name.charAt(0));
    }
}
```