Java Method Overloading

In Java, two or more methods may have the same name if they differ in parameters (different number of parameters, different types of parameters, or both). These methods are called overloaded methods and this feature is called method overloading. For example:

void func() { ... }

void func(int a) { ... }

float func(double a) { ... }

float func(int a, float b) { ... }

Here, the func() method is overloaded. These methods have the same name but accept different arguments.

How to perform method overloading in Java?

Here are different ways to perform method overloading:

1. Overloading by changing the number of parameters

```
class MethodOverloading {
    private static void display(int a){
        System.out.println("Arguments: " + a);
    }
    private static void display(int a, int b){
        System.out.println("Arguments: " + a + " and " + b);
    }
    public static void main(String[] args) {
        display(1);
```

```
        display(1, 4);
    }
}
```

Output:

Arguments: 1

Arguments: 1 and 4

Method Overloading by changing the data type of parameters

```
class MethodOverloading {
    // this method accepts int
    private static void display(int a){
        System.out.println("Got Integer data.");
    }
    // this method  accepts String object
    private static void display(String a){
        System.out.println("Got String object.");
    }
    public static void main(String[] args) {
        display(1);
        display("Hello");
    }
}
```

Output:

Got Integer data.

Got String object.

Java Method Overriding

Inheritance is an OOP property that allows us to derive a new class (subclass) from an existing class (superclass). The subclass inherits the attributes and methods of the superclass.

Now, if the same method is defined in both the superclass and the subclass, then the method of the subclass class overrides the method of the superclass. This is known as method overriding.

Example 1: Method Overriding

```
class Animal {
  public void displayInfo() {
    System.out.println("I am an animal.");
  }
}
class Dog extends Animal {
  @Override
  public void displayInfo() {
    System.out.println("I am a dog.");
  }
}
class Main {
  public static void main(String[] args) {
    Dog d1 = new Dog();
    d1.displayInfo();
```

```
  }}
```

Output:

```
I am a dog.
```

In the above program, the displayInfo() method is present in both the Animal superclass and the Dog subclass.

When we call displayInfo() using the d1 object (object of the subclass), the method inside the subclass Dog is called. The displayInfo() method of the subclass overrides the same method of the superclass.

super Keyword in Java Overriding

A common question that arises while performing overriding in Java is:

Can we access the method of the superclass after overriding?

Well, the answer is Yes. To access the method of the superclass from the subclass, we use the super keyword.

Use of super Keyword

```
class Animal {
  public void displayInfo() {
    System.out.println("I am an animal.");
  }
}

class Dog extends Animal {
  public void displayInfo() {
    super.displayInfo();
    System.out.println("I am a dog.");
  }
}
```

```
class Main {
  public static void main(String[] args) {
    Dog d1 = new Dog();
    d1.displayInfo();
  }
}
```

Run Code

Output:

I am an animal.

I am a dog.

Exercise:

1. Write a Java program to demonstrate method overloading with two methods having different data types.
2. Create a Java program to demonstrate method overloading with two methods having different numbers of parameters
3. Write a Java program to demonstrate method overloading with two methods having different data types and a common parameter
4. Create a Java program to demonstrate method overloading with a mixture of data types and parameter counts
5. Write a Java program to demonstrate method overloading with changing the number of parameters and data types
6. Create a Java program to demonstrate method overloading with overloaded methods that use different parameter types and return types, including type casting
7. Write a Java program to demonstrate method overloading with overloaded methods that use different parameter types, including booleans and boolean arrays
8. Create a Java program to demonstrate method overriding with a subclass that overrides a method with a different return type
9. Write a Java program to demonstrate method overriding with a subclass that overrides a method and calls the superclass method using the super keyword

10. Write a Java program to demonstrate method overriding with a subclass that overrides a method and changes the access modifier from public to private
11. Create a Java program to demonstrate method overriding with a subclass that overrides a method and adds a new parameter with a default value

Abstract Class in Java

An abstract class is a class marked with the abstract keyword.

Abstraction can be achieved with either abstract classes or interfaces .

- Abstract class must have one abstract method.
- We can’ts create object using abstract class.
- Abstract class can have abstract and non abstract methods.

Source Code

```
//Abstract Class in Java Programming
abstract class Shape
{
   abstract void draw();
   void message()
   {
      System.out.println("Message From Shape");
   }
}
class rectangleShape extends Shape
```

```
{
    @Override
    void draw() {
            System.out.println("Draw Rectangle Using Length & Breadth..");
    }
}
public class abstractDemo {
    public static void main(String args[]) {
        rectangleShape o =new rectangleShape();
        o.draw();
        o.message();
    }
}
```

Output

```
Draw Rectangle Using Length & Breadth..
Message From Shape
```

Method With varargs in Java

Varargs is a short name for variable arguments. In Java, an argument of a method can accept arbitrary number of values. This argument that can accept variable number of values is called varargs.

The syntax for implementing varargs is as follows:

```
accessModifier methodName(datatype… arg) {
   // method body
}
```

In order to define vararg, ... (three dots) is used in the formal parameter of a method.

A method that takes variable number of arguments is called a variable-arity method, or simply a varargs method.

Example: Working of varargs

```
class VarargExample {
   public int sumNumber(int ... args){
      System.out.println("argument length: " + args.length);
      int sum = 0;
      for(int x: args){
         sum += x;
      }
      return sum;
   }
   public static void main( String[] args ) {
```

```
        VarargExample ex = new VarargExample();
        int sum2 = ex.sumNumber(2, 4);
        System.out.println("sum2 = " + sum2);
        int sum3 = ex.sumNumber(1, 3, 5);
        System.out.println("sum3 = " + sum3);
        int sum4 = ex.sumNumber(1, 3, 5, 7);
        System.out.println("sum4 = " + sum4);
    }
}
```

Output:

```
argument length: 2
sum2 = 6
argument length: 3
sum3 = 9
argument length: 4
sum4 = 16
```

Here, the sumNumber() method returns the sum of int parameters passed to it (doesn't matter the number of arguments passed).

Example 2

```
public class MethodArgs {
```

```
    //Method With Varargs in Java
    public static void getNames(String... names) {
        for (String name : names)
            System.out.println(name);
    }
    public static void main(String args[]) {
        getNames("Ram", "Sam", "Ravi", "Kumar", "sara");
    }
}
```

Output

Ram

Sam

Ravi

Kumar

sara

This Java program defines a public class called MethodArgs which contains two methods: getNames and main. The getNames method is declared with a variable-length argument list using the ellipsis (...) notation, which allows the method to accept any number of arguments of type String.

In the implementation of the getNames method, a for-each loop is used to iterate through the names array and print each element to the console. The main method serves as the entry point for the program. In this method, the getNames method is called with five string arguments: "Ram", "Sam", "Ravi", "Kumar", and "sara".

https://www.tutorjoes.in/java_programming_tutorial/method_overloading_exercise_programs_in_java
https://www.tutorjoes.in/java_programming_tutorial/method_overriding_exercise_programs_in_java

https://javaconceptoftheday.com/java-practice-questions-on-method-overloading-and-overriding/