D.S. COLLEGE, ALIGARH

BCA IVth Semester

Session : 2022 - 2023

JAVA ASSIGNMENT

TOPIC : Overloading And Overriding

"Java Programming (C-401)"

Submitted By :
NAME: GAGAN DIXIT
ROLL NO: 55
UNIVERSITY ROLL NO :
2100085011045

Submitted To :
DR. MONIKA VARSHNEY

Method Overloading in Java

If a class has multiple methods having same name but different in parameters, it is known as **Method Overloading.**

Method overloading is also known as Compile-time Polymorphism, Static Polymorphism, or Early binding in Java. In Method overloading compared to parent argument, child argument will get the highest priority.

Different Ways of Method Overloading in Java

- Changing the Number of Parameters.
- Changing Data Types of the Arguments.
- Changing the Order of the Parameters of Methods.

1. Changing the Number of Parameters

Method overloading can be achieved by changing the number of parameters while passing to different methods.

Example with a program:

```
import java.io.*;
// Class 1 (Helper class)
class Product {
 public int multiply(int a, int b)
   {
      int prod = a * b;
      return prod;
   }
   public int multiply(int a, int b, int c)
   {
      int prod = a * b * c;
      return prod;
   }
}

// Class 2 (Main class)
class GFG {
   public static void main(String[] args)
   {

      Product ob = new Product();
      int prod1 = ob.multiply(1, 2);
       System.out.println(
         "Product of the two integer value :" + prod1);
         int prod2 = ob.multiply(1, 2, 3);
          System.out.println("Product of the three integer value :" + prod2);
    }
}
```

Output:

Product of the two integer value :2

Product of the three integer value :6

2. Changing Data Types of the Arguments

In many cases, methods can be considered Overloaded if they have the same name but have different parameter types, methods are considered to be overloaded.

Example with a program:

```
import java.io.*;

// Class 1 (Helper class)

class Product {

     public int Prod(int a, int b, int c){

       int prod1 = a * b * c;

       return prod1;

    }

    public double Prod(double a, double b, double c){

       double prod2 = a * b * c;

       return prod2;

    }

}

// Class 2 (Main class)

  class GFG {

    public static void main(String[] args){

       Product obj = new Product();

       int prod1 = obj.Prod(1, 2, 3);

       System.out.println("Product of the three integer value :" + prod1);

       double prod2 = obj.Prod(1.0, 2.0, 3.0);

       System.out.println("Product of the three double value :" + prod2);

    }

}
```

Output:

Product of the three integer value : 6

Product of the three double value : 6.0

3. Changing the Order of the Parameters of Methods

Method overloading can also be implemented by rearranging the parameters of two or more overloaded methods. For example, if the parameters of method 1 are (String name, int roll_no) and the other method is (int roll_no, String name) but both have the same name, then these 2 methods are considered to be overloaded with different sequences of parameters.

Example with a program:

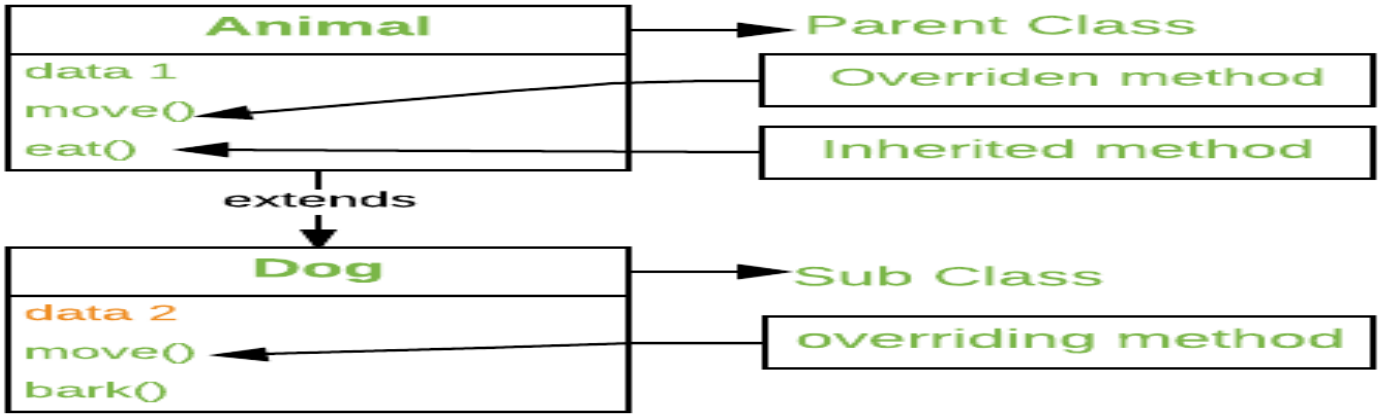
```
import java.io.*;

//Class 1 (Helper class)

class Student {

   public void StudentId(String name, int roll_no){

    System.out.println("Name :" + name + " " + "Roll-No :" + roll_no);

   }

   public void StudentId(int roll_no, String name)

   {

     System.out.println("Roll-No :" + roll_no + " " + "Name :" + name);

   }

}

//Class 2 (Main class)

class GFG {

    public static void main(String[] args){

   Student obj = new Student();

       obj.StudentId("Gagan", 55);

      obj.StudentId(2, "Lucky");

   }

}
```

Output:

Name :Gagan Roll-No :55
Roll-No :2 Name :Lucky

Overriding in Java

In any object-oriented programming language, Overriding is a feature that allows a subclass or child class to provide a specific implementation of a method that is already provided by one of its super-classes or parent classes. When a method in a subclass has the same name, same parameters or signature, and same return type(or sub-type) as a method in its super-class, then the method in the subclass is said to override the method in the super-class.



Example:

```
class Parent {
   void show() {
      System.out.println("Parent's show()");
   }
}
class Child extends Parent {
   @Override
   void show() {
      System.out.println("Child's show()");
   }
}
class Main {
   public static void main(String[] args){
      Parent obj1 = new Parent();
      obj1.show();
       Parent obj2 = new Child();
      obj2.show();
   }
}
```

Output:

```
Parent's show()
Child's show()
```

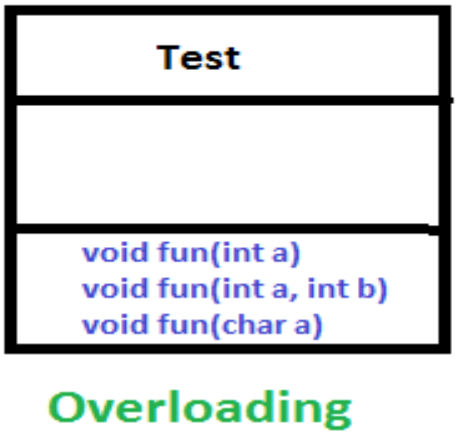
Usage of Java Method Overriding

- Method overriding is used to provide the specific implementation of a method which is already provided by its superclass.
- Method overriding is used for runtime polymorphism.

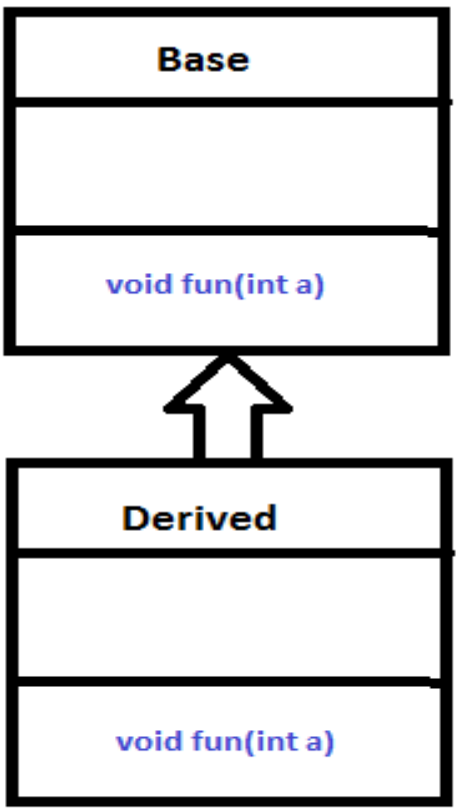
Rules for Java Method Overriding

1. The method must have the same name as in the parent class
2. The method must have the same parameter as in the parent class
3. There must be an IS-A relationship (inheritance)

Over loading vs Over riding



Overloading



Overriding

- Overloading is about same method have different signatures. Overriding is about same method, same signature but different classes connected through inheritance.
- Overloading is an example of compiler-time polymorphism and overriding is an example of run time polymorphism.

No.	Method Overloading	Method Overriding
1)	Method overloading is used *to increase the readability* of the program.	Method overriding is used *to provide the specific implementation* of the method that is already provided by its super class.
2)	Method overloading is performed *within class*.	Method overriding occurs *in two classes* that have IS-A (inheritance) relationship.
3)	In case of method overloading, *parameter must be different*.	In case of method overriding, *parameter must be same*.
4)	Method overloading is the example of *compile time polymorphism*.	Method overriding is the example of *run time polymorphism*.
5)	In java, method overloading can't be performed by changing return type of the method only. *Return type can be same or different* in method overloading. But you must have to change the parameter.	*Return type must be same or covariant* in method overriding.

MCQs

1. Which of this keyword can be used in a subclass to call the constructor of superclass?

a) super

b) this

c) extent

d) extends

2. What is the process of defining a method in a subclass having same name & type signature as a method in its superclass?

a) Method overloading

b) Method overriding

c) Method hiding

d) None of the mentioned

3. Which of these keywords can be used to prevent Method overriding?

a) static

b) constant

c) protected

d) final

4. Which of these is correct way of calling a constructor having no parameters, of superclass A by subclass B?

a) super(void);

b) superclass.();

c) super.A();

d) super();

5. At line number 2 in the following code, choose 3 valid data-type attributes/qualifiers among “final, static, native, public, private, abstract, protected”

```
public interface Status
  {
      /* insert qualifier here */ int MY_VALUE = 10;
  }
```

a) final, native, private

b) final, static, protected

c) final, private, abstract

d) final, static, public

6. Which of these is supported by method overriding in Java?

a) Abstraction

b) Encapsulation

c) Polymorphism

d) None of the mentioned

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

```
 class Alligator
{
  public static void main(String[] args)
   {
   int []x[] = {{1,2}, {3,4,5}, {6,7,8,9}};
   int [][]y = x;
   System.out.println(y[2][1]);
   }
}
```

a) 2

b) 3

c) 7

d) Compilation Error

8. 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

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

```
class Abc
{
    public static void main(String[]args)
    {
        String[] elements = { "for", "tea", "too" };
        String first = (elements.length > 0) ? elements[0]: null;
    }
}
```

a) Compilation error

b) An exception is thrown at run time

c) The variable first is set to null

d) The variable first is set to elements[0]

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

```
class A
{
    int i;
    public void display()
    {
        System.out.println(i);
    }
}
class B extends A
{
    int j;
```

```
    public void display()
    {
        System.out.println(j);
    }
}
class Dynamic_dispatch
{
    public static void main(String args[])
    {
        B obj2 = new B();
        obj2.i = 1;
        obj2.j = 2;
        A r;
        r = obj2;
        r.display();
    }
}
```

a) 1

b) 2

c) 3

d) 4

11. Which of this keyword can be used in a subclass to call the constructor of superclass?

a) super

b) this

c) extent

d) extends

12. What is the process of defining a method in a subclass having same name & type signature as a method in its superclass?

a) Method overloading

b) Method overriding

c) Method hiding

d) None of the mentioned

13. Which of these keywords can be used to prevent Method overriding?

a) static

b) constant

c) protected

d) final

14. Which of these is correct way of calling a constructor having no parameters, of superclass A by subclass B?

a) super(void);

b) superclass.();

c) super.A();

d) super();

15. To successfully overload a method in Java, the method names must be ___.

A) Same

B) Different

C) Same or different

D) None

16. Java method overloading implements the OOPS concept ___.

A) Inheritance

B) Polymorphism

C) Encapsulation

D) None

17. Which is the overloaded static method of Math class to get absolute value in Java?

A) Math.abs(int)

B) Math.abs(float)

C) Math.abs(double)

D) All the above

18. What is the process of defining two or more methods within same class that have same name but different parameters declaration?

a) method overloading

b) method overriding

c) method hiding

d) none of the mentioned

19. Which of these can be overloaded?

a) Methods

b) Constructors

c) All of the mentioned

d) None of the mentioned

20. What is the process of defining a method in terms of itself, that is a method that calls itself?

a) Polymorphism

b) Abstraction

c) Encapsulation

d) Recursion