Polymorphism in JAVA

Dr. Susovan Jana

Associate Professor & Programme In-Charge

Department of Computer Science & Engineering (IoT)

Institute of Engineering & Management, Kolkata, INDIA

Email (O): susovan.jana@iem.edu.in

Email (P): jana.susovan2@gmail.com

What is Polymorphism?

- The word “poly” means many and “morphs” means forms, So it means many forms.
- It allows multiple implementation with same name

Types of Java Polymorphism

- Compile-time Polymorphism
  - Method Overloading
- Runtime Polymorphism
  - Method Overriding

What is Method Overloading?

- Multiple function with same name but with different implementation details
- Do not deals with the return type because it creates ambiguity
- Increases the readability of the program

How to Method Overloading?

- By changing number of arguments
- By changing the data type

By changing the number of arguments

```
class Calculation{
        void sum(int a, int b){
        System.out.println(a+b);}
        void sum(int a, int b, int c){
        System.out.println(a+b+c);}
        public static void main(String args[]){
        Calculation obj=new Calculation();
        obj.sum(10,10,10);
        obj.sum(20,20);
        } }
```

Output:

```
30
40
```

By changing the data type

```
class Calculation{
        void sum(int a,int b){
        System.out.println(a+b);}
        void sum(double a,double b){
        System.out.println(a+b);}
        public static void main(String args[]){
        Calculation obj=new Calculation();
        obj.sum(10.5,10.5);
        obj.sum(20,20);
} }
```

- **Output:**

```
21.0
40
```

Constructor

- Constructor in java is a special type of method that is used to initialize the object.
- Java constructor is invoked at the time of object creation. It constructs the values i.e. provides data for the object that is why it is known as constructor.
- The java compiler provides a default constructor if you don't have any constructor.

Rules for Creating Constructor

- Constructor name must be same as its class name
- Constructor must have no explicit return type

Types of Java Constructors

- ❑ Default constructor (no-arg constructor)
- ❑ Parameterized constructor

Default Constructor

```
class Student{
int id;
String name;
void display(){
System.out.println(id+" "+name);}
public static void main(String args[]){
Student s1=new Student();
Student s2=new Student();
s1.display();
s2.display();
}}
```

Output:

```
0 null
0 null
```

Parameterized Constructor

```
class Student{
            int id;
            String name;
            Student(int i, String n){
                        id = i;
                        name = n;
            }
            void display(){
                        System.out.println(id+" "+name);
                        }
            public static void main(String args[]){
                        Student s1 = new Student(101,“Rohit");
                        Student s2 = new Student(201,“Virat");
                        s1.display();
                        s2.display();
                        }
            }
```

Output: 101 Rohit
102 Virat

Constructor Overloading

```
class Student{
        int id;
        String name;
        Student(int i, String n){
                id = i;
                name = n;
        }
        Student(int i, String n, int a){
                id = i;
                name = n;
                age=a; }
        void display(){
                System.out.println(id+" "+name+" "+age);
                }
```

```
public static void main(String args[]){
        Student s1 = new Student(101,"Rohit");
        Student s2 = new Student(201,"Virat",34);
        s1.display();
        s2.display();
        }
}
```

Output:

```
101 Rohit 0
102 Virat 34
```

Java Copy Constructor

```
class Student{
int id;
String name;
Student(int i, String n){
id = i;
name = n;
}
Student(Student s){
id = s.id;
name =s.name;
}
void display(){
System.out.println(id+" "+name);
}
public static void main(String args[]){
Student s1 = new Student(101,“Rohan");
Student s2 = new Student(s1);
s1.display();
s2.display();
}
}
```

Output:

```
101 Rohan
101 Rohan
```

Static Keyword

- variable (also known as class variable)
- method (also known as class method)
- block

Static Variable

- The static variable can be used to refer the common property of all objects (that is not unique for each object) e.g. company name of employees, college name of students etc.
- The static variable gets memory only once in class area at the time of class loading.

Counter using Static Variable

```
class Counter
{
static int count=0;//will get memory only once and retain its value
Counter(){
count++;
System.out.print(count);
}
public static void main(String args[]){
Counter2 c1=new Counter2();
Counter2 c2=new Counter2();
Counter2 c3=new Counter2();
}
}
Output: 1 2  3
```

Method Overriding

should be discussed with inheritance