Overloading

Shristi Technology Labs

Contents

- Polymorphism
  - Method Overloading
  - Constructor Overloading
- Varargs
- Overloading vs Varargs

Two types of polymorphism

Compiletime Polymorphism(Overloading)

Runtime Polymorphism(Overriding)

Compiletime Polymorphism

- Also called as ***overloading/Static Binding/Static method dispatch***
- Two types
  - **Constructor Overloading**
  - **Method Overloading**
- The method/constructor is overloaded with 0 to n number of parameters.
- The binding of the method and the calling happens during the compile time. (if not available gives compiler error)

Method Overloading

Syntax

access- specifier returntype **methodname(parameter list)**

access -specifier, return type – any

methodname – same

parameterlist – the no of parameters or type is different

Example

void calcArea(int x);

void calcArea(int x, int y); →(overloaded)

void calcArea(double y); →(overloaded)

int calcArea(int a); →error

```
class Methover{
    public static void main(String args[]){
        Shape s=new Shape();
        s.calcArea(10);
        s.calcArea(10.5f);
System.out.println("Rec"+s.calcArea(10,20));
    double tri=s.calcArea(10,5.0f);
    System.out.println("Tri:"+tri);
    }
}
```

```
class Shape{
    void calcArea(int a){
    System.out.println("Square:"+(a*a));
    }
    int calcArea(int a,int b){
        return(a*b);
    }
    void calcArea(float b){
System.out.println("Circle:"+(3.14*b*b));
    }
    double calcArea(int a,float b){
        return(0.5*a*b);
    }
}
```

Constructor Overloading

- The constructor gets overloaded with one to many parameters

Syntax

access-specifier **classname (parameterlist)**

Example

```
public Demo(int x)
public Demo(int x, int y)
```

```
class Conover{
      public static void main(String args[]){
      Sample s=new Sample();
      Sample s1=new Sample(10);
      Sample s2=new Sample(10,20);
      Sample s3=new Sample(100,200);
                s1.calcSum();
                s2.calcSum();
                s3.calcSum();
      }
}
```

```
class  Sample{
     int x,y;
     Sample(){
  System.out.println("default”);
     }
     Sample(int x){
                this.x=x;

       }
     Sample(int a, int b){
                x=a;
                y=b;
     }
     void calcSum(){
        System.out.print(x+y);
     } }
```

Var-Args

- substitute for overloading
- called as variable argument list
- takes arg from 0 to many

Syntax

access-specifier returntype methodname(datatype… varargs list)

Rules

- Can have only one vararg
- var args can be the last argument only.
- datatype must be followed by 3 dots(…)

Example

public void area(int… x) → takes 0 to many values of x

```
class VarArDemo{
    public static void main(String args[]){
    VarArDemo s=new VarArDemo();
            s.calcSum("Ram");
            s.calcSum("Tom",90);
            s.calcSum("John",20,30,40);
        }
    void  calcSum(String name,int... b){
    System.out.print("Welcome"+name); int sum = 0;
            for(int v:b)
               sum=sum+v;
             System.out.println("sum "+sum);
    }
/*
   void calcSum(String name){
     System.out.print("Have a goodday  "+name);
}
*/
}
```

Welcome Ram
Sum 0

Welcome Tom
Sum 90

Welcome John
Sum 100

Have a goodday Ram

To use Overloading or Varargs

Var-args

- If the logic(functionality) is same but the parameter list is different use varargs.

eg. double calcAvg(int … x)

can be used for

- **double calcAvg(90,80)**
- **double calcAvg(90,80,100)**
- **double calcAvg(90)**

Overloading

- If the logic and the parameter list is diff, use overloading

eg. double calcArea(int x) →(sq)
double calcArea(double x) →(circle)

Summary

- Overloading
  - Method Overloading
  - Constructor Overloading
- Varargs
- Overloading vs Varargs

Thank You