METHOD OVERLOADING & ARRAY INTRODUCTION (PART1)

27/10/2022

SWARUP MANDAL
Full Stack Java Developer
swarupmandal07

Ⓐ **method overloading:-**
Method overloading in Java is a feature that allows a class to have more than one method with same name, but with different Parameters.

?why→ To develop a Java application many programmer collaborately developing same application. For a developer it is defecult to remember so many method name of particular task in a class. To avoid this java introduce M.O.

⊛ ⟹ In C Programming method overloading does not allowed. Because compiler of C does not support it.

E.g# Class Calculator {

```
        int add(int a, int b){
            return a+b;
        }
        int add (int a, int b, int c){
            return a+b+c;
        }
        float add(int a float b){
            return a+b;
        }
        float add( float a, float b){
            return a+b;
        }
        float add (int a, float b, float c){
            return a+b+c;
        }
        double add( int a, int b, double c){
            return a+b+c;
        }
        double add (double a, double b, double c){
            return a+b+c;
        }
```

P.T.O

```
Public class MethodOverLoading {
   P.S.v.m (S[] args) {
   Calculator calc = new Calculator(); // got class life
 { int a=10, b=20, c=30; // integer value manually given
   float m=10.32f, n=5.3f, o=2.3f; // float
   double x=2.323, y=7.894, z=5.235; // double

   S.o.Pln (calc.add(a, b));
   S.o.Pln (calc.add(a, m));
   S.o.Pln(calc.add(a, n, o)); // i,f,f
   S.o.Pln(calc.add(m, n); // f,f
   S.o.Pln (calc.add(a, b, x)); // int, int, double
   S.o.Pln (calc.add(x, y, o)); // d, d, f
   }
}
```

(*) Parameter variable ~~name~~ can be changes. it does not matter to give exact name

(*) ⇒ $\boxed{\text{one : many = Polymorphism}}$

= 1 : M = Polymorphism ⇒ add

= add is a 1 method : performing many activities = Polymorphism

But, This polymorphism is a false Pol....

[Because, Generally one method of add performing only one activity internally.]

(*) **Rules of resolving the issue for M.O by the compiler:-**

HOW?

1. number of parameters.
2. Data type of the Parameters.
3. Order of the datatype of the parameters

(*) example: Println is follows method overloading.

POLYMORPHISM

(*)(*) ① Compiletime Polymorphism / Static Polymorphism / Early binding / False Polymorphism (Method overloading) —

→ Java polymorphism allows the incorporation of multiple methods with in a class. The methods use the same name but the parameter varies. This represents the static polymorphism.

→ This polymorphism is resolved during the compile time and is achieved through the method overloading. (C-T. P) S.P in java decides which method to execute during compile time.

② Run-Time Polymorphism / Dynamic Polymorphism / Real Polymorphism :- (method overriding)
→ In this form of polymorphism in java, the compiler doesn't determine the method to be execute, It's the JVM that performs the process at the run time. D.P in java refers to the process when a call to an overridden process resolved at the run time.

⇒ **Some Snippets**

Q)

```
class Calculator{
    int add (int a, int b) {
            return a+b;
    }
    void add(int a, int b) {          // it gives CE for void
        int ~~result~~=a+b;              and int add method.
        S.O. Pln(result);             because both will store
} }                                   int a and int b.
Public class LaunchMocs {             compiler will
    P.S.v m (S[] a) {                 confuse and give
  Calculator calc = new Calculator(); (C. E)
  Calc.add (10, 20); // whom to give?
}}
```

Ⓐ return type has no role to play.

Ans: Compile-Time Error

ex:=

```
class Calcu{
   float add (float a, int b)
   { return a+b; }
   float add (float a, float b, int c)
   { return    a+b+c; }}
Public class Example2{
P.S.V.m (S[] args) {
Calcu calc = new Calcu();
S.O.Pln (calc(10 ,20)); // 10 will store in float.
                        (implicit type casting)
```

Ⓐ method overloading with numeric Type-Promotion or Implicit Type-casting.

classmate
Date
Page

Ex:3

```
Class Display{                 // this also a M.O Example
  void disp(){
  S.o.pl("ineuron");
  }
  void disp (String name){
     S.o.pl(name);
  }
  void disp (int age){
     S.o.pl(age);
} }
public class LaunchMoh{
    p.s.v.m(S[] args){
   Display d=new Display();
   d.disp();
   d.disp(28);
   d.disp("Swarup");
 }
```

I.Q) Can we overload main() method?

Ans:- Yes, we can overload main method however JVM will call such a main method which accept (String[] args) as parameters.

→ JVM will search and start from where String[] args present and indicate as starting point of code.

C.S#

```
public class LaunchMomain{             → starting point
   • public s.v.main(String[] args)
     { S.o.pln("it is actual main method");}
   • p.s.v.m(int[] args){
        S.o.pln("accepting int args");}
   • p.s.v.m(double d){
        S.o.pln("double value");
   }}
```

Method overloading ~~will~~ come back again in inharitance (method overridden also) ⟶

SWARUP MANDAL
Full Stack Java Developer
swarupmandal07