Inheritance

Shristi Technology Labs

Contents

- What is inheritance?
- Use of super keyword
- Constructor calling in inheritance scenario
- Overriding
- DynamicMethod Dispatch

Inheritance - Basics

- One class using the properties of another class
- Reusability.
- Java uses simple & multilevel inheritance.
- Keyword - extends
- Sub class can use all the properties and methods of the super class except private members
- Subclass can also have its own methods, variables
- The top level(super) class for all the classes is Object
- The super class variables are initialized and then the sub class variables are initialized

```
class Employee{
  String name; int age;
      Employee(){
      name = "Ram";
      age = 25;
       }
      void getDetails(){
        System.out.println("In super class");
        System.out.println("Name : "+ name);
        System.out.println("Age : "+ age);
}
 }
```

```
class Manager extends Employee{
      int salary;
      Manager(){
              name="shyam";
              age=35;
              salary=1000;
 }
   void m2(){
         System.out.println("In sub class");
       System.out.println("salary "+ salary);
    }
}
class inherdemo{
      public static void main(String a[]){
       Employee e=new Employee();
      e.getDetails();
      Manager ob=new Manager();
      ob.getDetails();
      ob.m2();
}}
```

Use of super keyword

- Has to be in the first line of a constructor
- Always in the sub class only.
- Allows to call and instantiate the super class variables and then comes down to sub class variables.
- super() is added in the first line by default by the compiler - if it is not written by the developer
- The parameters within super must match atleast one constructor of the super class

Constructor without parameters

```
class Employee{
  String name;int age;
      Employee(){
      name="Ram";
      age=25;
      }
}
class Manager extends Employee{
      int salary;
      Manager(){
       super();
      name="shyam";
      age=35;
      salary=1000;
 }
```

super(); → super() written by the compiler goes to super class(Object)

super() written by the developer goes to super class (Employee)

super() written by the compiler goes to super class(Employee)

Constructor with parameters

```
class Employee{
  String name; int age;
     Employee(String n,int a){
     name= n;
     age= a;
     }
}
class Manager extends Employee{
     int salary;
     Manager(String s,int a){
      super();
     //super(s,a);
              salary=1000;
   } }
```

goes to constructor of super class(Object)

No def. constructor. So gives a compiler error

goes to default constructor of super class(Employee)

goes to para. constructor of super class(Employee)

Runtime Polymorphism

- also called as ***Overriding/DynamicMethodDispatch***
- Is done to get ***different implementation*** from the sub classes.
- If the subclass has the same method signature of the superclass, overriding happens.

Overriding

Syntax

access-specifier returntype **methodname(parameterlist)**

access-specifier → same or level above(less restrictive)

methodname, parameterlist → same

returntype →same except co-variant return types

Example

void calculatesalary(int x) → super class (Employee)

void calculateSalary(int x) → sub class (Manager) calculates bonus for manager

void calculateSalary(int x) → sub class (Clerk) calculates bonus for clerk

eg. Three classes Employee(super) ,Manager, Clerk(subclasses).

Method in Employee

```
    void calcSalary(int  x){
System.out.print( “salary “  + x);
    }
```

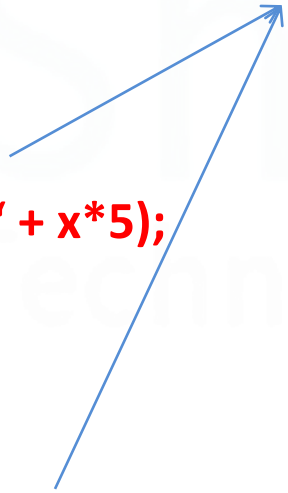
Method in Manager

```
    void calcSalary(int  x) {
    System.out.print( “salary “ + x*5);
    }
```

Method in Clerk

```
    void calcSalary(int x) {
System.out.print( “salary “  + x*3);
    }
```

Overriding super class method .
The subclass methods have their own logic for calculating salary



Dynamic Method Dispatch

- DMD happens during the runtime.
- The method gets dispatched dynamically during the runtime
- The super class decides which subclass method to call only during the runtime .
- How?

  from the clients input.
- using

  super-class ref **= sub-class object**

Example

```
Employee emp;
emp  = new Manager();
emp.calcSalary(100);

emp  = new Clerk();
emp.calcSalary(100);

emp  = new Employee();
emp.calcSalary(100);
```

Calls calcSalary() method of Manager

Calls calcSalary() method of Clerk

Calls calcSalary() method of Employee

Summary

- What is inheritance?
- Example
- Use of super keyword
- Example
- Calling constructor in inheritance scenario
- Overriding
- Example for overriding

Thank You