Object Oriented Programming using Java

Prepared By:
Suyel, PhD
Assistant Professor
Dept. of CSE, NIT Patna

Outline

1. Inheritance
2. Examples of Inheritance
3. Types of Inheritance
4. Important Points about Inheritance

Inheritance

- **Inheritance** can be defined as the process, where one class acquires the properties (method and field) of another class.
- With the use of inheritance the information is made manageable in a hierarchical order.
- When we inherit from an existing class, we can reuse methods and fields of the parent class. Moreover, we can also add new methods and fields in our current class.
- Inheritance is mainly used for method overriding and code reusability.
- **extends** is the keyword used to inherit the properties of a class.

Inheritance (Cont…)

- Syntax

```
class Parent
{
   //Methods and fields
}

class Child extends Parent
{

   //Methods and fields of Parent
   //Methods and fields of Child
}
```

- **Key terms**
  - **Sub class:** Subclass is a class, which inherits the other class. It is also called a derived class, extended class, or child class.
  - **Super class:** Superclass is the class from where a subclass inherits the features. It is also called a base class or a parent class.
  - **Reusability:** It is a mechanism which facilitates us to reuse the fields and methods of the existing class, when we create a new class.

Inheritance (Cont…)

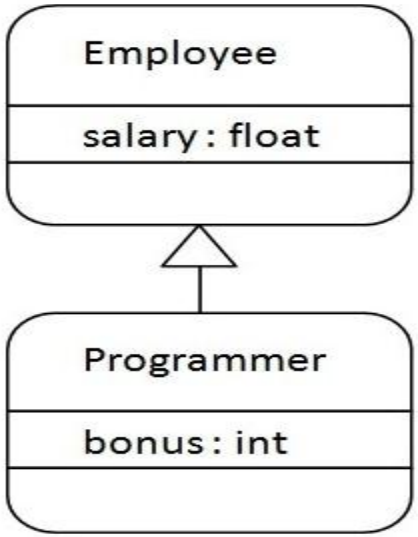

Fig. 1: Inheritance

- In Fig. 1, the relationship between the two classes is Programmer IS-A Employee. It means that Programmer is a type of Employee.

Examples of Inheritance

```
class Employee
{
   int salary = 10000;
}

class Programmer extends Employee
{
   int bonus = 2000;
   public static void main(String args[])
   {
      Programmer obj = new Programmer();
      System.out.println("Programmer salary is: " + obj.salary);
      System.out.println("Bonus of Programmer is: " + obj.bonus);
   }
}
```

Examples of Inheritance (Cont…)

- Output

```
Programmer salary is: 10000
Bonus of Programmer is: 2000
```

Examples of Inheritance (Cont…)

```
class Faculty
{
   String designation = "Asst. Prof.";
   String instituteName = "NIT Patna";
   void work()
   {
      System.out.println("Teaching and Administrative");
   }
}

class Cse extends Faculty
{
   String subject = "OOP";
   public static void main(String args[])
   {
      Cse obj = new Cse();
      System.out.println("Institute Name: " + obj.instituteName);
      System.out.println("Designation of Faculty: " + obj.designation);
      System.out.println("Subject teaching: " + obj.subject);
      obj.work();
   }
}
```

Examples of Inheritance (Cont…)

- Output

```
Institute Name: NIT Patna
Designation of Faculty: Asst. Prof.
Subject teaching: OOP
Teaching and Administrative
```

Examples of Inheritance (Cont…)

Animal.java

```
class Animal
{
   String name;
   public void eat()
   {
      System.out.println("It is cute");
   }
}
```

Dog.java

```
class Dog extends Animal
{
   public void disp()
   {
      System.out.println("Name of dog: " + name);
   }
}
```

Inheritance1.java

```
class Inheritance1
{
   public static void main(String[] args)
   {
      Dog obj = new Dog();

      obj.name = "Max";
      obj.disp();

      obj.eat();
   }
}
```

Examples of Inheritance (Cont…)

- Output

  Name of dog: Max

  It is cute

Examples of Inheritance (Cont…)

```
class Calculation
{
   int z;

   public void add(int x, int y)
   {
      z = x + y;
      System.out.println("The sum of the given numbers: " + z);
   }

   public void subt(int x, int y)
   {
      z = x - y;
      System.out.println("The difference between the given numbers: " + z);
   }
}

class Inheritance2 extends Calculation
{
   public void multi(int x, int y)
   {
      z = x * y;
      System.out.println("The multiplication of the given numbers: " + z);
   }

   public static void main(String args[])
   {
      int a = 5, b = 10;
      Inheritance2 obj = new Inheritance2();
      obj.add(a, b);
      obj.subt(a, b);
      obj.multi(a, b);
   }
}
```

Examples of Inheritance (Cont…)

- Output

```
The sum of the given numbers: 15
The difference between the given numbers: -5
The multiplication of the given numbers: 50
```

Types of Inheritance

- There are five types of inheritance in java:
  - **Single inheritance:** In single inheritance, subclasses inherit the features of one superclass.
  - **Multilevel inheritance:** In multilevel inheritance, a subclass inherits a superclass and as well as the subclass also acts as the base class to other subclass (child class).
  - **Hierarchical inheritance:** In hierarchical inheritance, one class serves as a superclass (base class) for more than one subclass.
  - **Multiple inheritance:** When one subclass inherits multiple superclasses, it is known as multiple inheritance.
  - **Hybrid inheritance:** When one subclass inherits multiple superclasses and these multiple superclasses also inherit one superclass, then it is known as hybrid inheritance.

Types of Inheritance (Cont…)

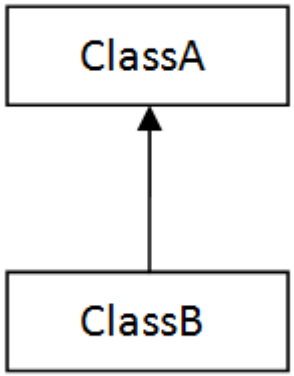

Fig. 2: Single inheritance

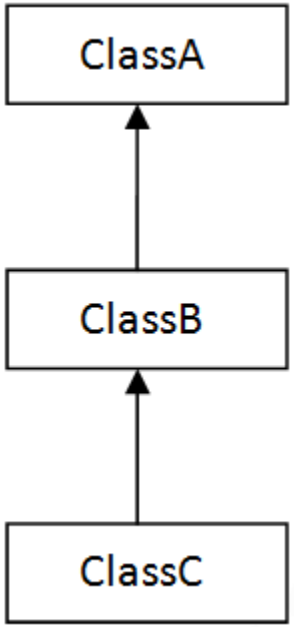

Fig. 3: Multilevel inheritance

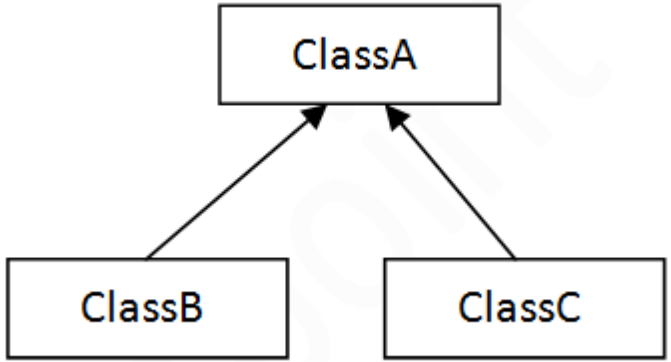

Fig. 4: Hierarchical inheritance

Types of Inheritance (Cont…)

- Java does not support multiple inheritance.
  - It is mainly because to reduce the complexity and simplify the language.
  - Consider a scenario where A, B and C are three classes. The C class inherits A and B classes. If A and B classes have the same method and we call it from the child class's object, there is an ambiguity to call the method of A or B class.
  - As compile-time errors are better than runtime errors, Java renders compile-time error, if we inherit 2 classes. So, whether we have same method or different, it will give compile time error.
- In Java, multiple and hybrid inheritance are supported through **interface** only.

Types of Inheritance (Cont…)

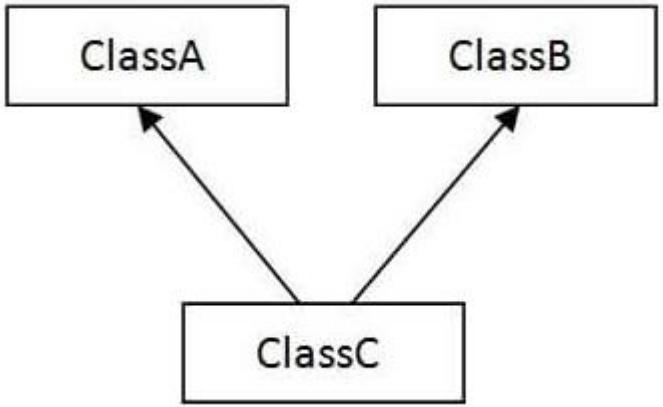

Fig. 5: Multiple inheritance

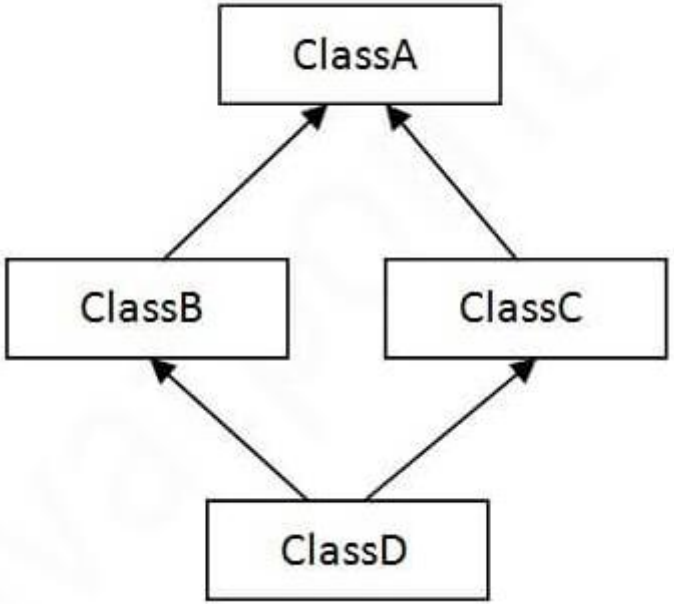

Fig. 6: Hybrid inheritance

Types of Inheritance (Cont…)

```
class Parents
{
    public void color()
    {
        System.out.println("Combination of fair and medium fair");
    }
}

class Child extends Parents
{
    public void colorChild()
    {
        System.out.println("Very fair");
    }
}

public class Inheritance3
{
    public static void main(String args[])
    {
        Child obj = new Child();
        obj.color();
        obj.colorChild();
    }
}
```

Types of Inheritance (Cont…)

- Output

  Combination of fair and medium fair

  Very fair

Types of Inheritance (Cont…)

```
class Animal
{
   String name;
   public void look()
   {
      System.out.println("It is cute");
   }
}

class Dog extends Animal
{
   public void bark()
   {
      System.out.println("It can bark");
   }
}

class BabyDog extends Dog
{
   public void disp()
   {
      System.out.println("Name of new born dog: " + name);
   }
}

class Inheritance4
{
   public static void main(String args[])
   {
      BabyDog obj = new BabyDog();

      obj.name = "Max";
      obj.disp();
      obj.bark();
      obj.look();
   }
}
```

Types of Inheritance (Cont…)

- Output

```
Name of new born dog: Max
It can bark
It is cute
```

Types of Inheritance (Cont…)

```
class Animal
{
   public void look()
   {
      System.out.println("It is cute");
   }
}

class Dog extends Animal
{
   public void bark()
   {
      System.out.println("It can bark");
   }
}

class Cat extends Animal
{
   public void eat()
   {
      System.out.println("It can eat");
   }
}

class Inheritance5
{
   public static void main(String args[])
   {
      Cat obj = new Cat();
      obj.eat();
      obj.bark();
      obj.look();
   }
}
```

Types of Inheritance (Cont…)

- Output

```
Inheritance5.java:32: error: cannot find symbol
      obj.bark();
          ^
  symbol:   method bark()
  location: variable obj of type Cat
1 error
```

Important Points about Inheritance

- **Default superclass**
  - Except Object class, which has no superclass, every class has one and only one direct superclass (single inheritance).
  - In the absence of any other explicit superclass, every class is implicitly a subclass of the Object class.
- **Superclass can only be one**
  - A superclass can have any number of subclasses.
  - However, a subclass can have only one superclass. This is because Java does not support multiple inheritances with classes.
  - Although with interfaces, multiple inheritances are supported by Java.

Important Points about Inheritance (Cont…)

- **Inheriting constructors**
  - A subclass inherits all the members (fields, methods and nested classes) from its superclass.
  - Constructors are not members, so they are not inherited by subclasses, but the constructor of the superclass can be invoked from the subclass.
- **Private member inheritance**
  - A subclass does not inherit the private members of its parent class.
  - However, if the superclass has public or protected methods (like getters and setters) for accessing its private fields, these can also be used by the subclass.

Important Points about Inheritance (Cont…)

- **Private member inheritance**

```
class Animal
{
   protected String name;
   protected void look()
   {
      System.out.println("It is cute");
   }
}

class Dog extends Animal
{
   public void info()
   {
      System.out.println("Name of the Dog: " + name);
   }
}

class Inheritance6
{
   public static void main(String args[])
   {
      Dog obj = new Dog();
      obj.name = "Max";
      obj.look();
      obj.info();
   }
}
```

Important Points about Inheritance (Cont…)

- **Private member inheritance (Cont…)**
  - **Output**

    It is cute

    Name of Dog: Max

Important Points about Inheritance (Cont…)

- **Private member inheritance (Cont…)**

```
class Animal
{
   private String name;
   private int age;
   void setData(String str1, int x1)
   {
      name = str1;
      age = x1;
   }
   String getData1()
   {
      return name;
   }
   int getData2()
   {
      return age;
   }
}

class Dog extends Animal
{
   String originalName;
   int originalAge;
   void originalSetData(String str1, int x1, String str2, int x2)
   {
      setData(str1, x1);
      originalName = str2;
      originalAge = x2;
   }
   void showData()
   {
      System.out.println("String String1: " + getData1());
      System.out.println("Integer x1: " + getData2());
      System.out.println("Original Name of Dog: " + originalName);
      System.out.println("Original Age of Dog: " + originalAge);
   }
}

class Inheritance7
{
   public static void main(String args[])
   {
      Dog obj = new Dog();
      obj.originalSetData("Tiger", 10, "Max", 5);
      obj.showData();
   }
}
```

Important Points about Inheritance (Cont…)

- **Private member inheritance (Cont…)**
  - **Output**

```
String String1: Tiger
Integer x1: 10
Original Name of Dog: Max
Original Age of Dog: 5
```


thank
you . . .

Slides are prepared from various sources, such as Book, Internet Links and many more.