COJ :: Inheritance

TalentSprint

Licensed To Skill

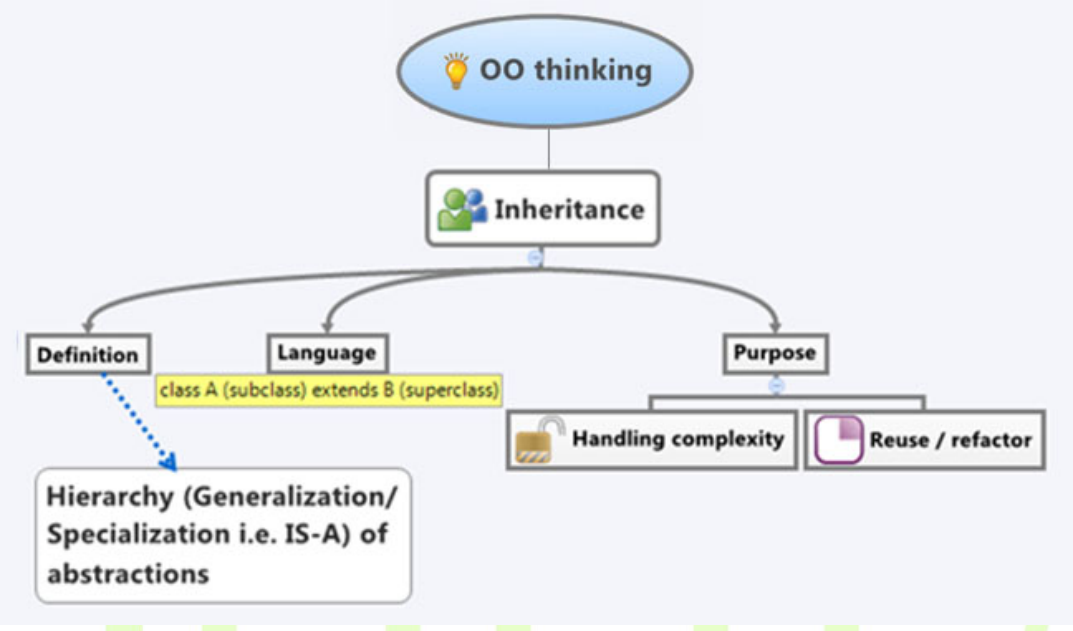
Version 1.0.4

Learning Objectives

By the end of this session, you will be able to:

- Define inheritance
- Explain the need for Inheritance
- Types of Inheritance
- Write Java code to create classes and subclasses in an inheritance hierarchy
- Use “super” keyword
- Explain constructor chaining in inheritance

Inheritance



Inheritance is the concept of a child class (sub class) automatically inheriting the variables and methods defined in its parent class (super class).

Inheritance

`Inheritance:`

- Inheritance can be defined as the process where one object acquires the properties of another.
- The keyword used for inheritance in Java is “extends”
- The relationship between two classes participating in Inheritance is “is-a”

Example

class Mammal **extends** Animal

Inheritance

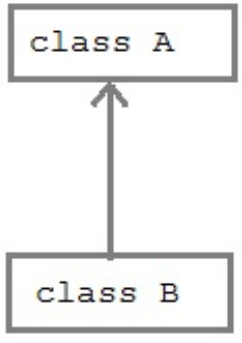
Use of Inheritance:

- Code Reusability
- Change Management

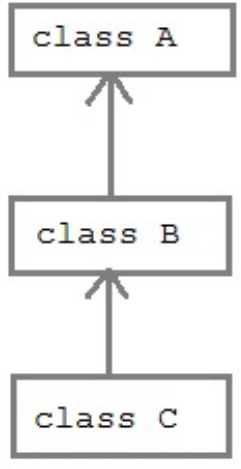
Types of Inheritance:

- Single or Simple Inheritance
- Multi-level Inheritance
- Hierarchial Inheritance

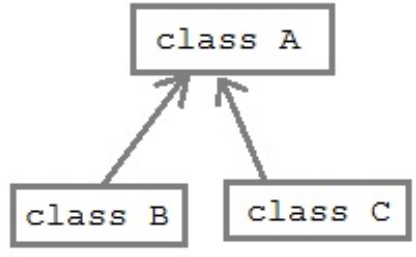
Inheritance



Simple Inheritance



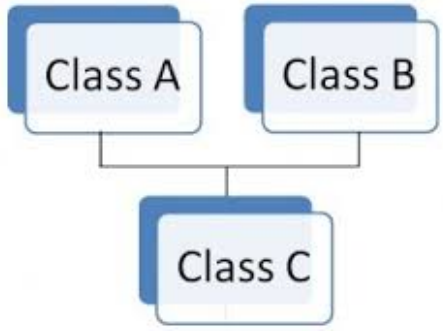
Multilevel inheritance



Heirarchical inheritance

Inheritance

Multiple Inheritance:



Note

Java doesn't support Multiple Inheritance. In Java, a class cannot inherit more than one class.

Inheritance

`Deriving a Subclass:`

To derive a child class, we use the extends keyword

Example

Suppose we have a parent class called Person. Then a subclass Student can be created as . . .

Inheritance

```
public class Person {
    protected String name;
    protected String address;
    /**
     * Default constructor
    */
    public Person() {
        System.out.println("Inside Person : Constructor");
        name = "";
        address = "";
    }
}
```

```
public class Student extends Person {
    public Student(){
        System.out.println("Inside Student : Constructor");
    }
. . . .
}
```

Inheritance

What one can do in a Sub-class regarding Attributes

- The inherited attributes can be used directly, just like any other attributes
- You can declare new attributes in the subclass that are not in the super class
- You can declare an attribute in the subclass with the same name as the one in the super class, thus hiding it (not recommended)

Inheritance

What one can do in a Sub-class regarding Methods

- The inherited methods can be used directly as they are
- You can declare new methods in the subclass that are not in the super class
- You can write a new instance method in the subclass that has the same signature as the one in the super class, thus overriding it

Inheritance

Need for “super”

Let us consider the following code snippet

```
class Person {
  String name;
  int age;
public Person(String name,int
age)
{
      this.name = name;
      this.age = age;
}
}
class Student extends Person {
  int rollNo;
  int rank;
 public Student(String
name,int age,int rollNo, int
rank)
{
      this.name = name;
      this.age = age;
      this.rollNo = rollNo;
      this.rank = rank;
}
}
```

```
Class Student extends Person{
  int rollNo;
  int rank;

 public Student(String name,int age,int
rollNo, int rank)
{
      super(name,age);
  this.rollNo = rollNo;
  this.rank = rank;
}
}
```

Calls the super class constructor

Can I avoid this repetition...?? Let me Try..!

Inheritance

What is “super” keyword?

- A subclass can explicitly call a constructor of its super class using the super constructor call e.g.:**super**()
- A super constructor call in the constructor of a subclass will result in the execution of relevant constructor from the super class, based on the arguments passed.
  Ex: **super**(10,20);//it will invoke two-parameterized super class constructor

Inheritance

```
class Person{
     String name;
     public Person(String name){
        this.name=name;
     }
}

class Student extends Person{
    int rollNo;
    public Student(int rollNo){
       super();

       this.rollNo = rollNo;
    }
}
```

```
class Student extends Person{
    int rollNo;
    public Student(int rollNo){
      super("Grady Booch");

      this.rollNo = rollNo;
    }
}
class Test{
    public static void
      main(String args[]){
     Student st;
     st = new Student(131);
   }
}
```

1. By default, the compiler will insert a super() call in the sub-class constructor which calls the default constructor of the super-class.
2. If the super-class doesnâĂŹt have default constructor, we have to explicitly write a super() call in sub-class which matches with the constructor in super-class

Inheritance

Few things to remember when using the super constructor call:

- The **super**() call must occur as the first statement in a constructor
- The **super**() call can only be used in a constructor (not in ordinary methods)
- The Java compiler inserts **super**() call as the first statement of sub class constructor if we don't provide it

Another use of super is to refer to members of the super class (just like the keyword "**this**").

Inheritance

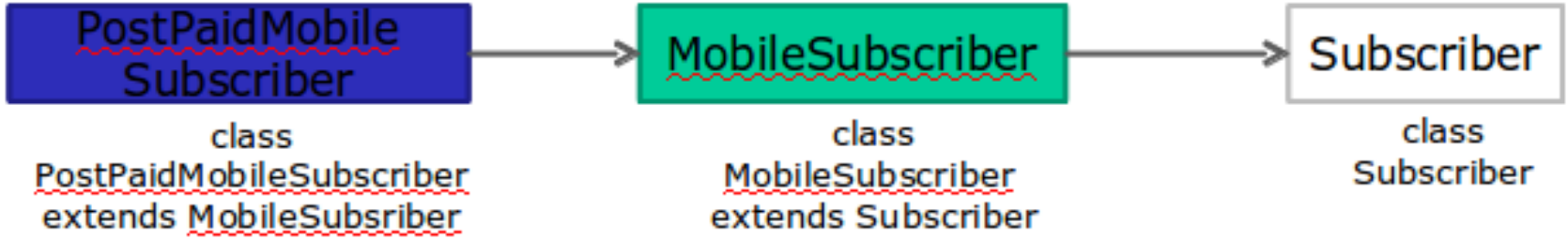
When does super class constructor gets called ?
A subclass constructor invokes the constructor of the immediate superclass implicitly.
In the below example the sub-class (Student) constuctor calls super-class (Person) constructor implictly.

```
class Student extends Person{
        Student(){
           super();
        }
}
```

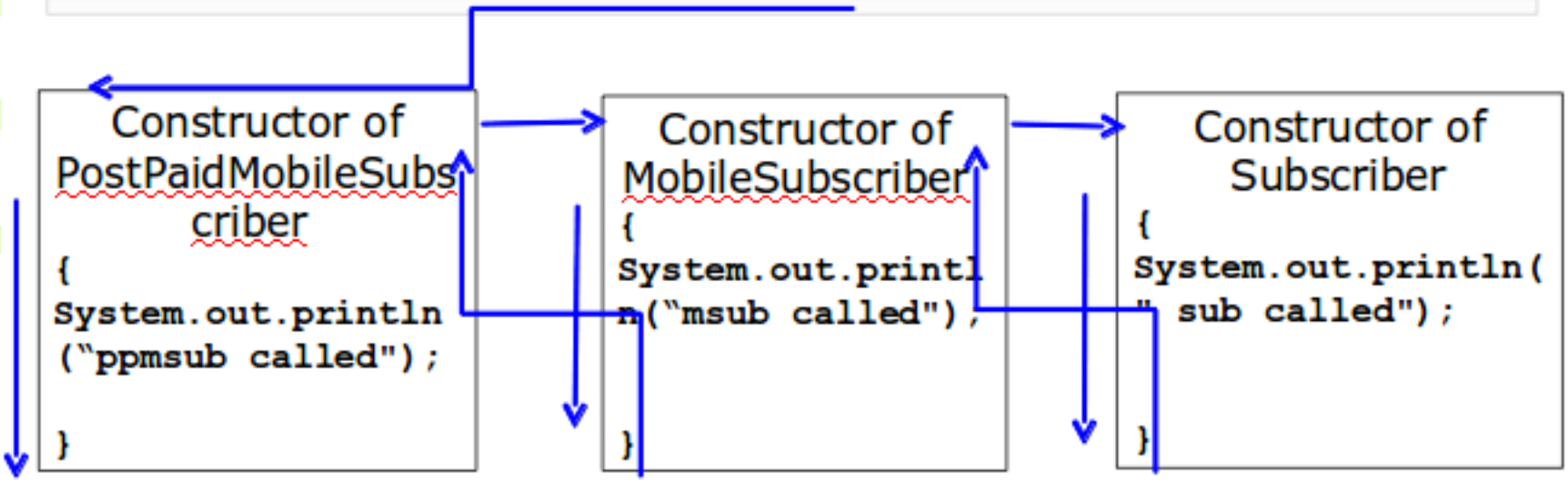
```
class  Person{
        Person(){
        }
}
```

Inheritance

Constructor Chaining:



If an object of PostPaidMobileSubscriber is created as:
PostPaidMobileSubscriber ppms = new PostPaidMobileSubscriber ();



Inheritance

Constructor Chaining in Inheritance

```
public class Subscriber {
    public Subscriber() {
        System.out.println("Subscriber  is called");
    }
}
public class MobileSubscriber extends Subscriber {
    public MobileSubscriber() {
        System.out.println("Mobile Subscriber is called");
    }
}
public class PostPaidMobileSubscriber extends MobileSubscriber {
    public PostPaidMobileSubscriber() {
        System.out.println("Postpaid mobile Subscriber is called");
    }
}
```

Inheritance

```
public class RunProgram {
    public static void main(String args[]) {
        PostPaidSubscriber ppsc = new
    PostPaidSubscriber();
    }
}
```

Output

```
Subscriber is called
Mobile Subscriber is called
Postpaid mobile Subscriber is called
```

Inheritance