Java

Agenda

- Java Inheritance
- Method Overriding
- super Keyword
- final Keyword

Inheritance

- In OOP, computer programs are designed in such a way where everything is an object that interact with one another. Inheritance is one such concept where the properties of one class can be inherited by the other.

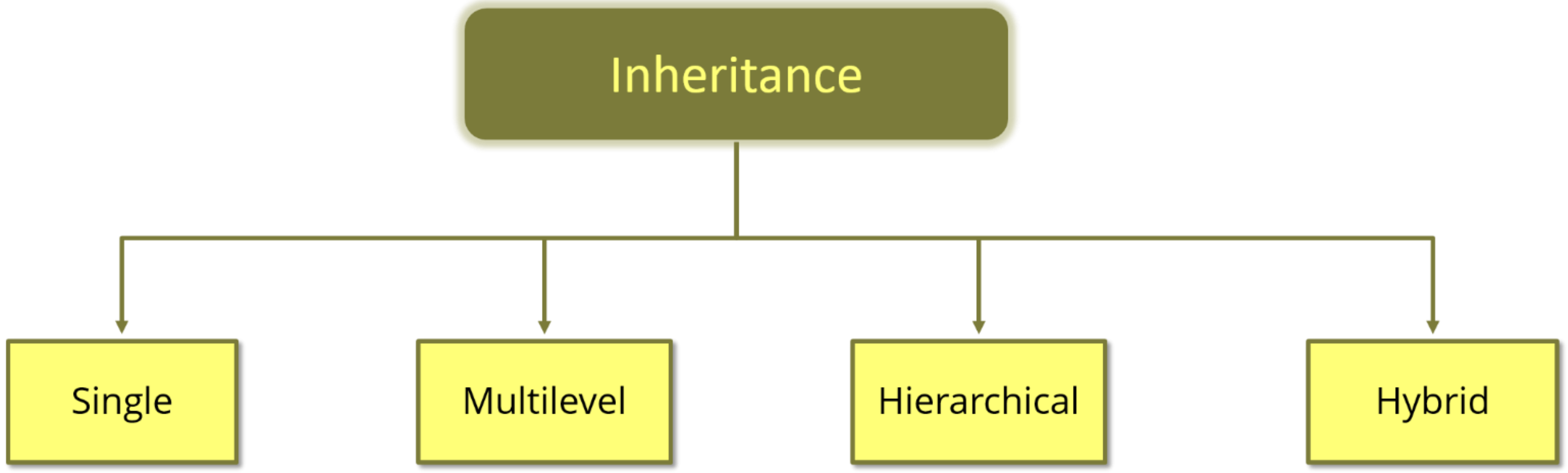
- It helps to reuse the code and establish a relationship between different classes.

- In Java, there are two classes:
  1. Parent class (Super or Base class)
  2. Child class (Subclass or Derived class)

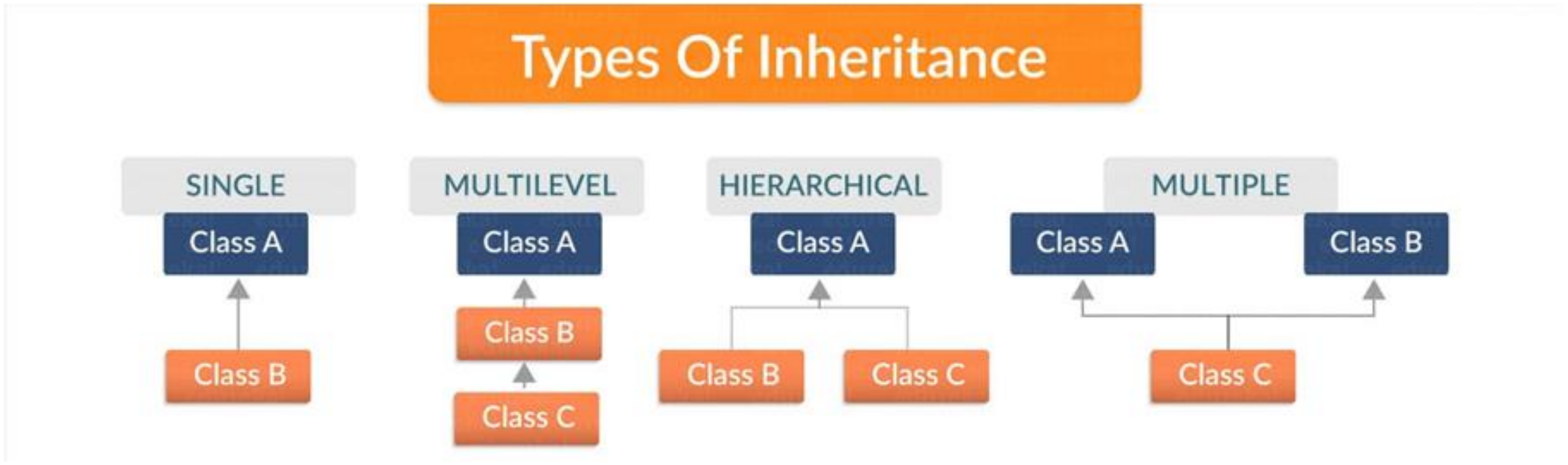
- A class which inherits the properties is known as Child Class whereas a class whose properties are inherited is known as Parent class.

Types of Inheritance

- Inheritance is further classified into 4 types.

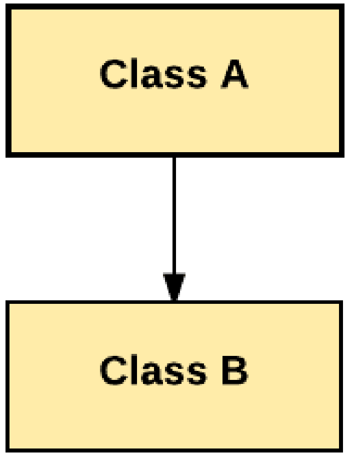


Types of Inheritance..



Single Inheritance

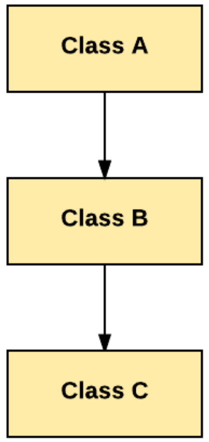
- In single inheritance, one class inherits the properties of another. It enables a derived class to inherit the properties and behavior from a single parent class.
- This will in turn enable code reusability as well as add new features to the existing code.
- Here, Class A is your parent class and Class B is your child class which inherits the properties and behavior of the parent class.

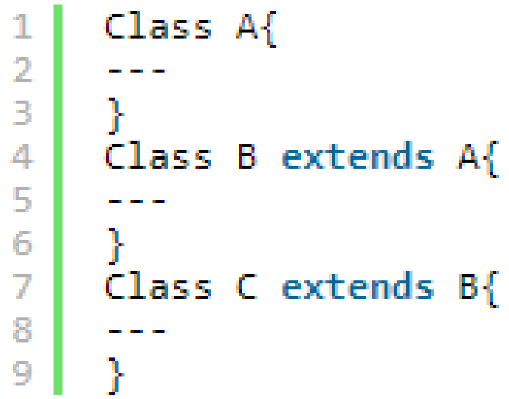


```
Class A
{
---
}
Class B extends A {
---
}
```

Multilevel Inheritance

- When a class is derived from a class which is also derived from another class, i.e. a class having more than one parent class but at different levels, such type of inheritance is called Multilevel Inheritance.
- If we talk about the flowchart, class B inherits the properties and behavior of class A and class C inherits the properties of class B. Here A is the parent class for B and class B is the parent class for C. So in this case class C implicitly inherits the properties and methods of class A along with Class B. That's what is multilevel inheritance.

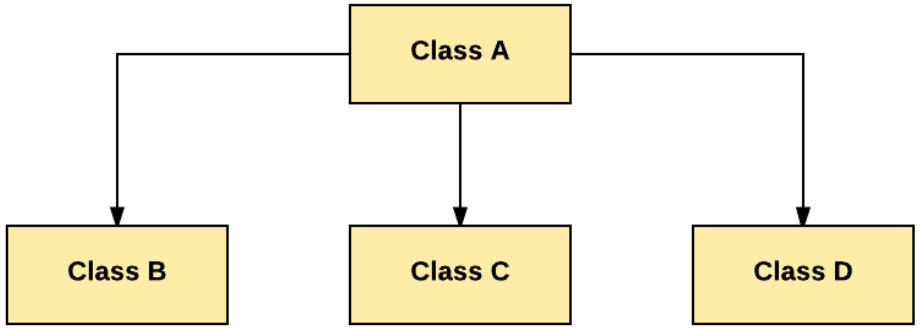


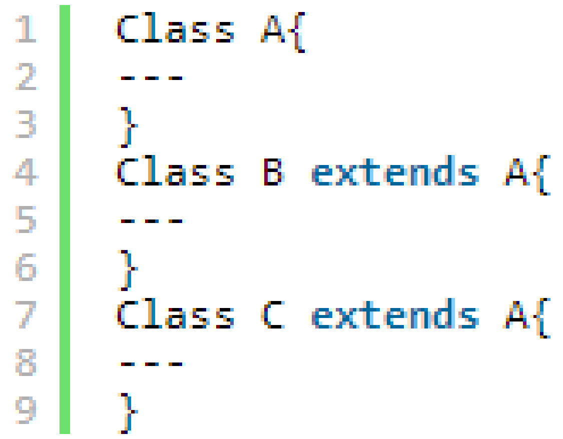


```
Class A{
---
}
Class B extends A{
---
}
Class C extends B{
---
}
```

Hierarchical Inheritance

- When a class has more than one child classes (sub classes) or in other words, more than one child classes have the same parent class, then such kind of inheritance is known as **hierarchical**.
- If we talk about the flowchart, Class B and C are the child classes which are inheriting from the parent class i.e Class A.

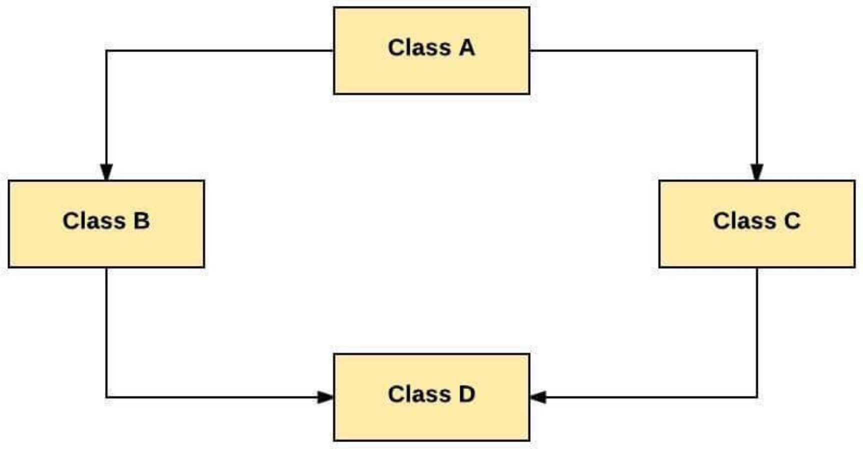




```
Class A{
---
}
Class B extends A{
---
}
Class C extends A{
---
}
```

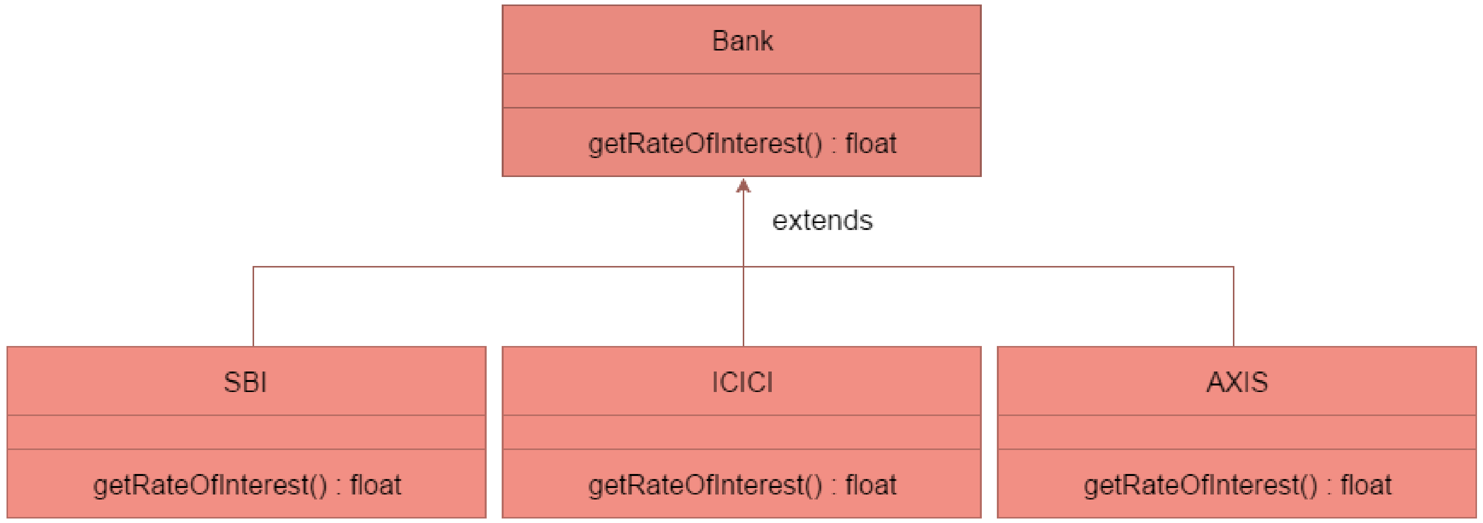
Hybrid Inheritance

- Hybrid inheritance is a combination of *multiple* inheritance and *multilevel* inheritance. Since multiple inheritance is not supported in Java as it leads to ambiguity, so this type of inheritance can only be achieved through the use of the interfaces.
- If we talk about the flowchart, class A is a parent class for class B and C, whereas Class B and C are the parent class of D which is the only child class of B and C.



Method Overriding

- If subclass (child class) has the same method as declared in the parent class, it is known as **method overriding in java**.
- **Rules for Java Method Overriding**
  - method must have same name as in the parent class
  - method must have same parameter as in the parent class.


Bank
getRateOfInterest() : float
extends
SBI
getRateOfInterest() : float
ICICI
getRateOfInterest() : float
AXIS
getRateOfInterest() : float

super keyword

- The **super** keyword in java is a reference variable which is used to refer immediate parent class object.
- **Usage of java super Keyword**
  - ➢ super can be used to refer immediate parent class instance variable.
  - ➢ super can be used to invoke immediate parent class method.
  - ➢ super() can be used to invoke immediate parent class constructor.

Super Keyword

- super is used to refer immediate parent class instance variable.

```
class Animal
{
 String color="white";
}
class Dog extends Animal
{
 String color="black";
 void printColor()
  {
   System.out.println(color);//prints color of Dog class
   System.out.println(super.color);//prints color of Animal class
  }
}
```

```
class TestSuper1
{
public static void main(String args[])
{
Dog d=new Dog();
d.printColor();
}
}
```

Super Keyword

- super can be used to invoke parent class method.

```
class Animal
{
void eat()
{
System.out.println("eating...");
}
}
class Dog extends Animal
{
void eat()
{
System.out.println("eating bread...");
}

void work()
{
super.eat();
eat();
}
}
```

```
class TestSuper2
{
public static void main(String args[])
{
Dog d=new Dog();
d.work();
}
}
```

Super Keyword

- super is used to invoke parent class constructor

```
class Animal
{
 Animal()
 {
 System.out.println("animal is created");
 }
}

class Dog extends Animal
{
 Dog()
 {
 super();
 System.out.println("dog is created");
 }
}
```

```
class TestSuper3
{
public static void main(String args[])
{
Dog d=new Dog();
}
}
```

final Keyword

- The **final keyword** in java is used to restrict the user. The java final keyword can be used for variables, methods and classes.
  - variable
  - method
  - class

Java final variable

- If you make any variable as final, you cannot change the value of final variable(It will be constant).

```
class Bike9
{
 final int speedlimit=90;//final variable
 void run()
 {
  speedlimit=400;
 }
 public static void main(String args[])
 {
 Bike9 obj=new  Bike9();
 obj.run();
 }
}//end of class
```

Output:Compile Time Error

Java final method

- If you make any method as final, you cannot override it.

```
class Bike
{
  final void run()
  {
  System.out.println("running");
  }
}
class Honda extends Bike
{
   void run()
   {
   System.out.println("running safely with 100kmph");
   }

   public static void main(String args[])
   {
   Honda honda= new Honda();
   honda.run();
   }
}
```

Output:Compile Time Error

Java final class

- If you make any class as final, you cannot extend it.

```
final class Bike
{

}

class Honda1 extends Bike
{
  void run()
  {
  System.out.println("running safely with 100kmph");
  }

  public static void main(String args[])
  {
  Honda1 honda= new Honda();
  honda.run();
  }
}
```

Output:Compile Time Error

Assignment

- **Assignment-1**
- Create a class 'Teacher' which contains following variables and methods
  - designation = "Teacher";
  - collegeName = "BusyQA";
  - does() → Teaching
- Create another class 'ComputerTeacher' which extends 'Teacher' class then create objects then call methods.