CMPS 4143 - Topics in Contemporary Programming Languages

L6- Java Inheritance

Images and text taken from textbook unless otherwise noted.

Inheritance

- Main features of OOP
- Form of software reusability
- (Derived) classes are created by absorbing the methods and variables of an existing (base) class
- It then adds its own methods to enhance its capabilities
- **"Is a"** relationship: derived class object can be treated as base class object - Inheritance
- **"Has a**" relationship: class object has object references as members – composition
- A derived class can only access non-private base class members unless it inherits accessor functions
- Constructors are not inherited

Inheritance (…)

- Inheritance does not have to stop at one layer of classes.
- Could have an `Executive` class that extends `Manager`.
- Path from a particular class to its ancestor is the *inheritance chain*.
- May be more than one chain of descent. For example, a subclass `Programmer` and `Secretary` that extends `Employee`.
- Java does not support **multiple** inheritance. (Has interface classes, though.)

Inheritance (…)

Source: Sitesbay.com

Base and Derived Classes

Base class	Derived classes
`Student`	`GraduateStudent` `UndergraduateStudent`

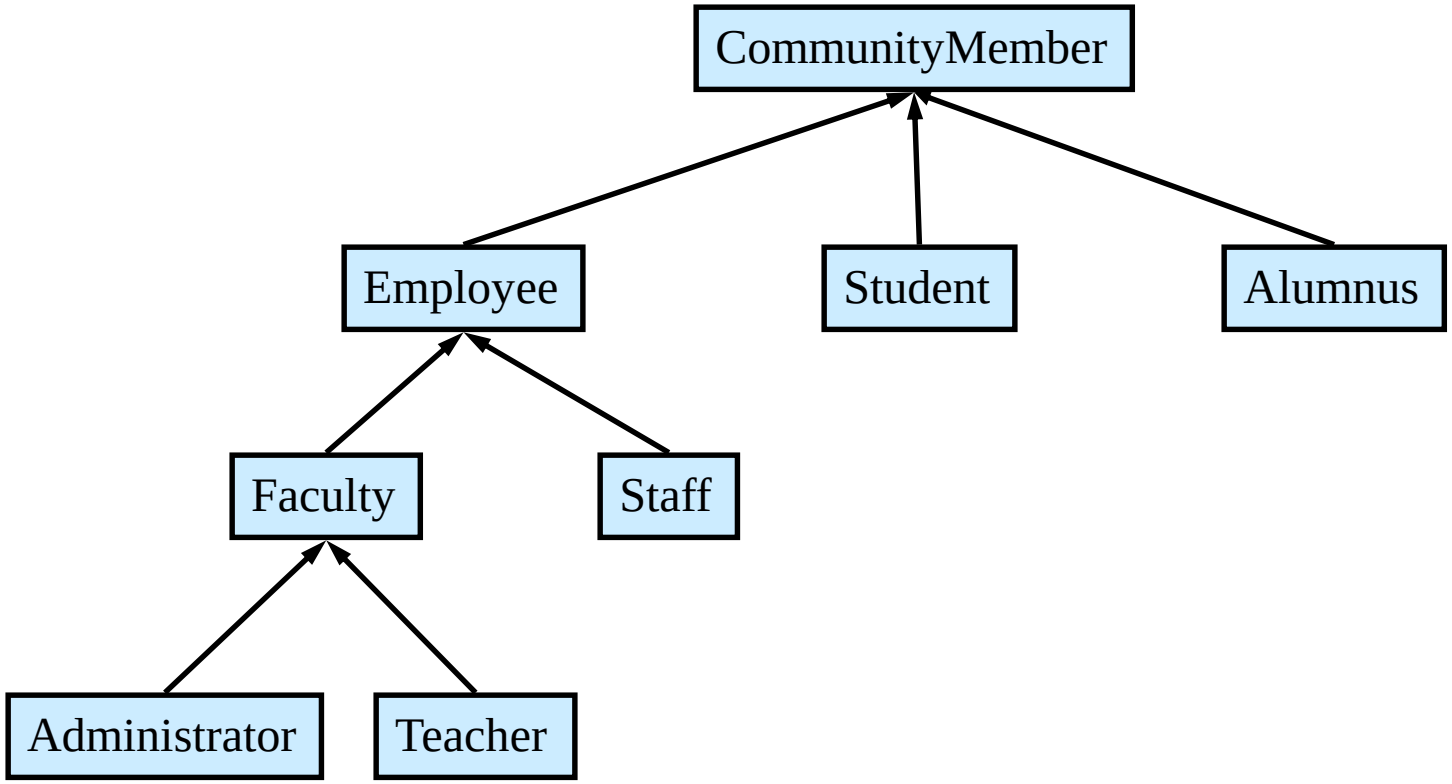
`Shape`	`Circle`
	`Triangle`
	`Rectangle`

`Account`	`CheckingAccount` `SavingsAccount`

To specify that class Two is derived from class One

```
class Two extends One or    class Circle extends Shape
{                           {
```

Base and Derived Classes (…)



Inheritance forms a tree-like hierarchy

Inheritance hierarchy for university **`CommunityMember`**s.

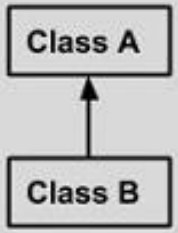
Members

- All inheritance in Java is public inheritance
  - Can be accessed by base class or any class derived from that base class
  - No analog to C++ features of private and protected

- Definitions
  - super, base, parent class
  - child, sub-, derived class

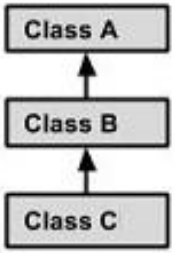
Types of Inheritance

Single Inheritance



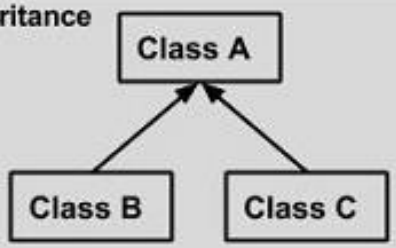

```
public class A {
   .......
}
public class B extends A {
   .........
}
```

Multi Level Inheritance



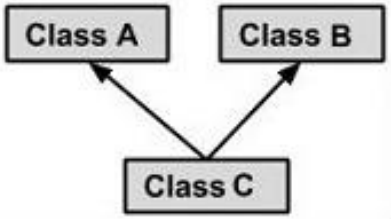

```
public class A { ..................}
public class B extends A {.....................}
public class C extends  B {..................... }
```

Hierarchical Inheritance




```
public class A { ..................}
public class B extends A {.....................}
public class C extends A {..................... }
```

Multiple Inheritance




```
public class A { ..................}
public class B {.....................}
public class C extends  A,B {
   .....................
} // Java does not support mutiple Inheritance
```

Case Study

- Bicycle is a base class
  - Attributes: Gear, Speed
  - Behaviors: Brake, Speedup, Status
- Mountain Bike is a derived class that extends Bicycle class
  - Attributes: Gear, Speed
  - Behaviors: Brake, Speedup, Status
  - New Attribute: SeatHeight
- Test is a driver class to run program

Case Study (base class…)

```
// Java program to illustrate the
// concept of inheritance

// base class
class Bicycle {
        // the Bicycle class has two fields
        public int gear;
        public int speed;

        // the Bicycle class has one constructor
        public Bicycle(int gear, int speed)
        {
                this.gear = gear;
                this.speed = speed;
        }
        // the Bicycle class has three methods
        public void applyBrake(int decrement)
        {
                speed -= decrement;
        }
```

Case Study (base class…)

```
        public void speedUp(int increment)
        {
                speed += increment;
        }

        // toString() method to print info of Bicycle
        public String toString()
        {
                return ("No of gears are " + gear + "\n"
                                + "speed of bicycle is " + speed);
        }
}
```

Case Study (derived class)

```
// derived class
class MountainBike extends Bicycle {

        // the MountainBike subclass adds one more field
        public int seatHeight;

        // the MountainBike subclass has one constructor
        public MountainBike(int gear, int speed,
                                                int startHeight)
        {
                // invoking base-class(Bicycle) constructor
                super(gear, speed);
                seatHeight = startHeight;
        }
```

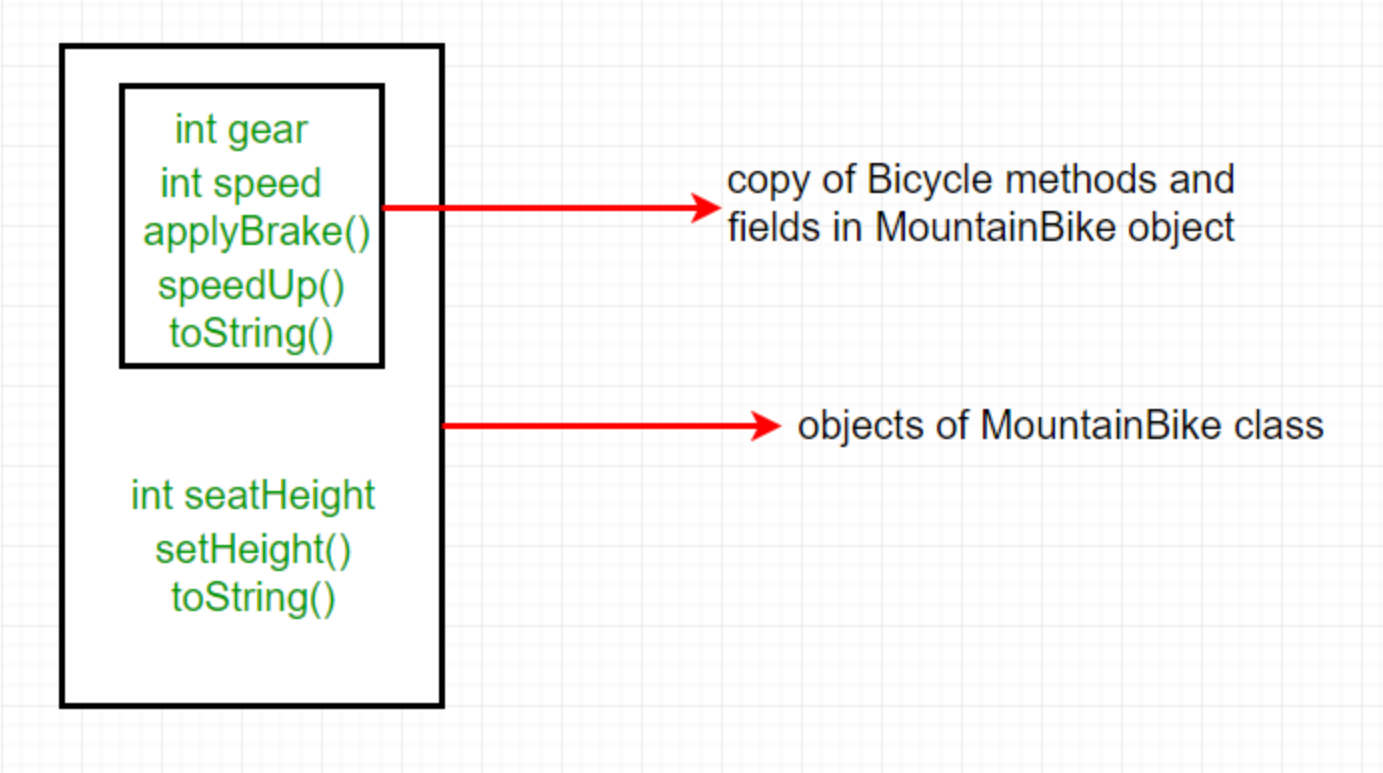
Case Study (derived class…)

```
        // the MountainBike subclass adds one more method
        public void setHeight(int newValue)
        {
                seatHeight = newValue;
        }

        // overriding toString() method
        // of Bicycle to print more info
        @Override public String toString()
        {
                return (super.toString() + "\nseat height is "
                                + seatHeight);
        }
}
```

Case Study (driver class)

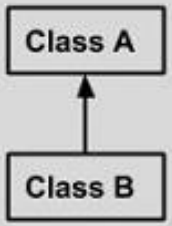
```
// driver class
public class Test {
        public static void main(String args[])
        {

                MountainBike mb = new MountainBike(3, 100, 25);
                System.out.println(mb.toString());
        }
}
```

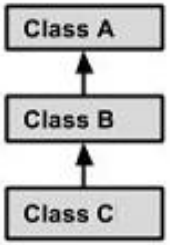
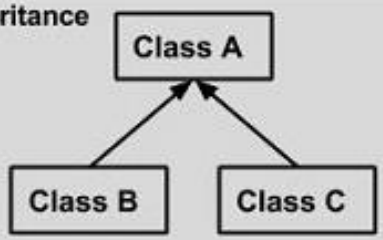
Case Study

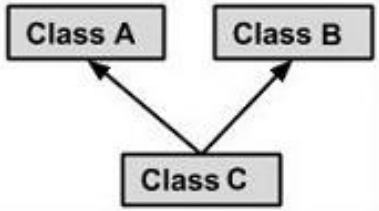


Source: https://www.geeksforgeeks.org/inheritance-in-java/

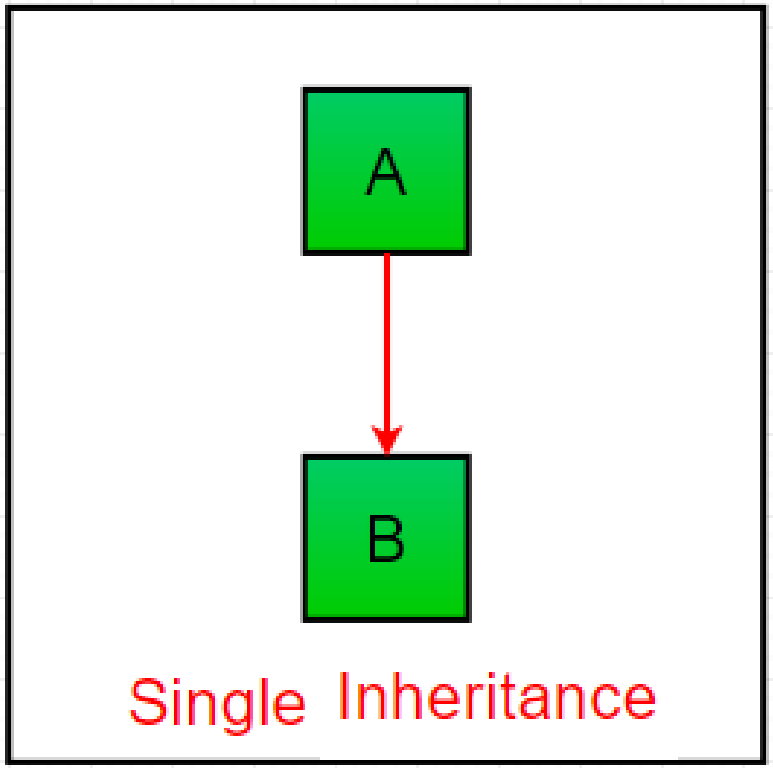
Types of Inheritance

Single Inheritance
 
	public class A {
	.....
	}
	public class B **extends** A {
	.....
	}

Multi Level Inheritance 	public class A { .....} public class B **extends** A {.....} public class C **extends** B {..... }
Hierarchical Inheritance 	public class A { .....} public class B **extends** A {.....} public class C **extends** A {..... }

Multiple Inheritance
 
	public class A { .....}
	public class B {.....}
	public class C **extends** A,B {
	.....
	} // Java does not support mutiple Inheritance

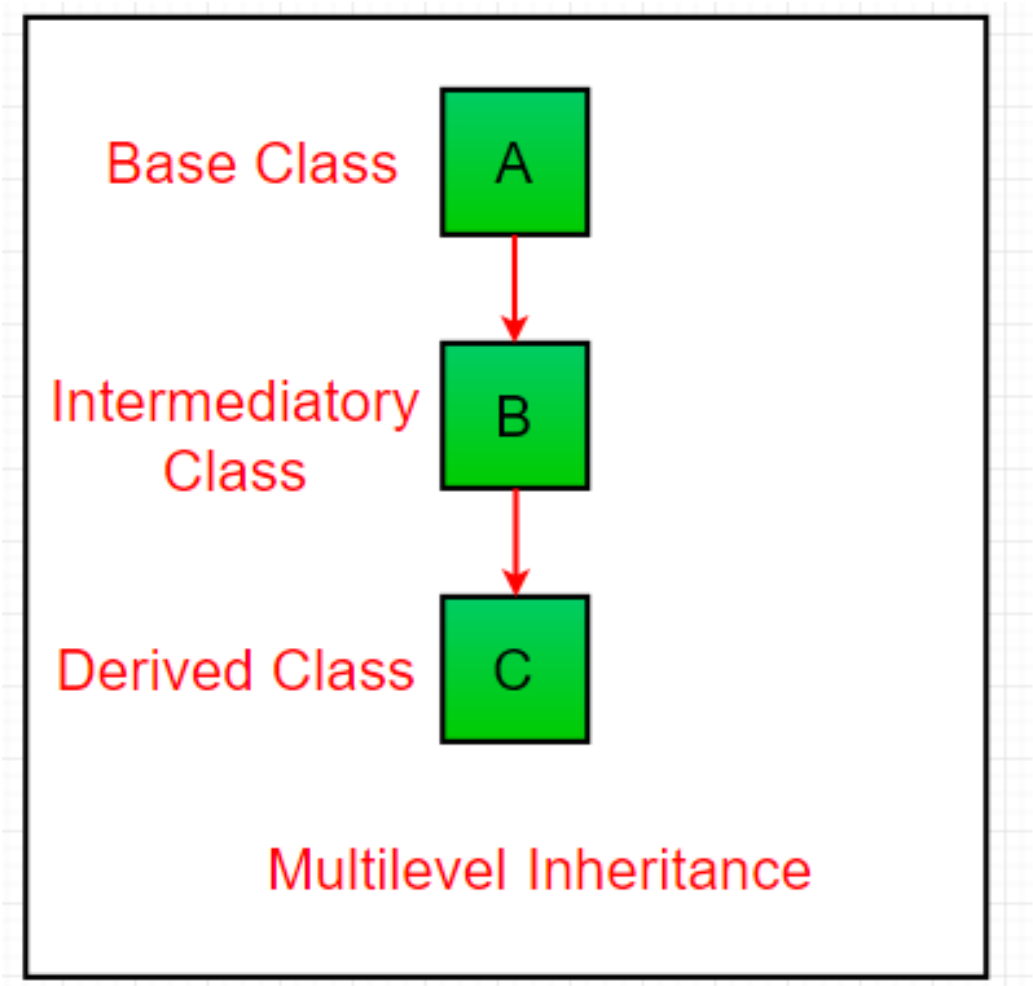
Single Inheritance



Single Inheritance Example

```
class one{
    public void print_base() {
        System.out.println(”Printing base“);
    }
}
class two extends one {
    public void print_derived() { System.out.println(“Printing derived"); }
}
// Driver class
public class Main {
    public static void main(String[] args) {
        two obj = new two();
        obj.print_base();
        obj.print_derived();
    }
}
```

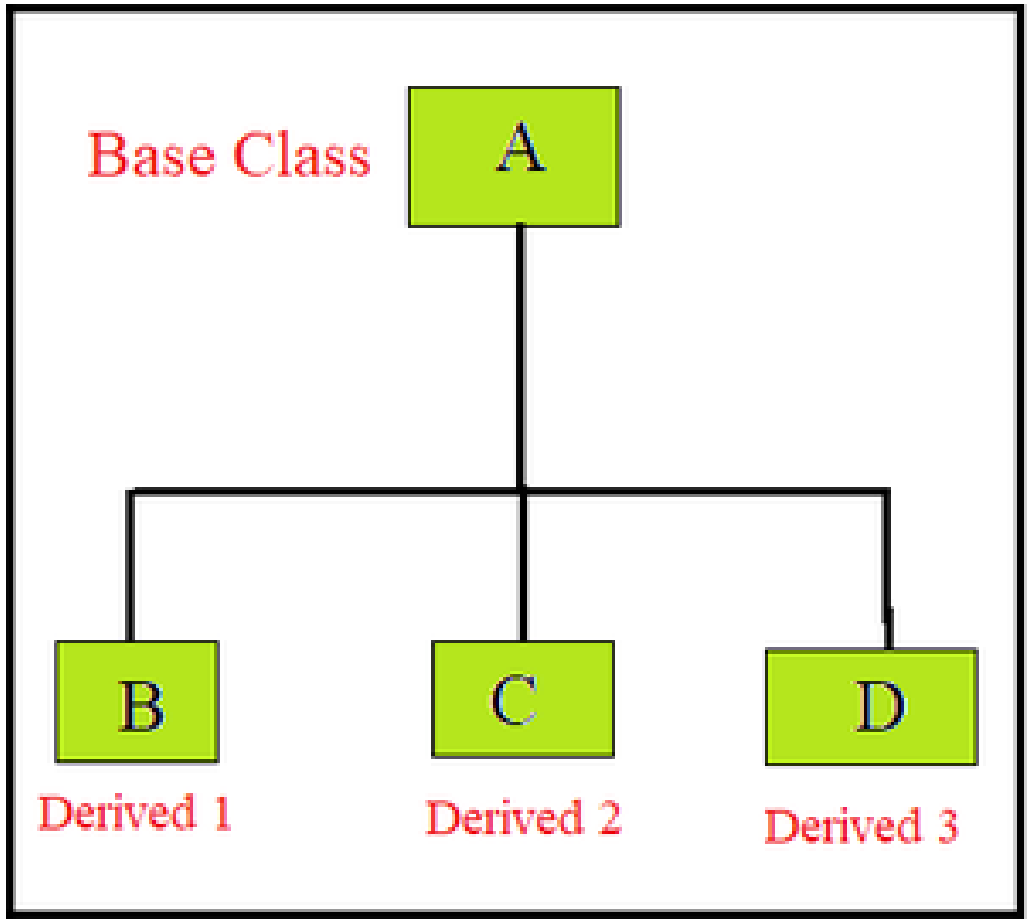
Multilevel Inheritance



Multilevel Inheritance Example

```
class one {
    public void print_one(){ System.out.println("Printing 1"); }
}

class two extends one {
    public void print_two() { System.out.println("Printing 2"); }
}

class three extends two {
    public void print_three(){ System.out.println("Printing 3"); }
}

public class Main {        // Drived class
    public static void main(String[] args){
        three obj = new three();
        Obj.print_one();     obj.print_two();     obj.print_three();
}}
```

Hierarchical Inheritance



Hierarchical Inheritance Example

```
class A {
    public void print_A() { System.out.println("Class A"); }
}

class B extends A {
    public void print_B() { System.out.println("Class B"); }
}

class C extends A {
    public void print_C() { System.out.println("Class C"); }
}

class D extends A {
    public void print_D() { System.out.println("Class D"); }
}
```

Hierarchical Inheritance Example (…)

```
// Driver Class
public class Test {
    public static void main(String[] args)
    {
        B obj_B = new B();
        obj_B.print_A();
        obj_B.print_B();

        C obj_C = new C();
        obj_C.print_A();
        obj_C.print_C();

        D obj_D = new D();
        obj_D.print_A();
        obj_D.print_D();
    }
}
```

Output

```
Class A
Class B
Class A
Class C
Class A
Class D
```

Questions?