Inheritance

- One of the pillars of object-orientation.
- A new class is derived from an existing class:
    1) existing class is called super-class
    2) derived class is called sub-class
- A sub-class is a specialized version of its super-class:
    1) has all non-private members of its super-class
    2) may provide its own implementation of super class methods
- Objects of a sub-class are a special kind of objects of a super-class.

Inheritance

- **Inheritance** is the mechanism in java by which one class is allow to inherit the features(fields and methods) of another class.
- Important terminology:
- Super Class**:** The class whose features are inherited is known as super class(or a base class or a parent class).
- Sub Class: The class that inherits the other class is known as sub class(or a derived class, extended class, or child class).
- The subclass can add its own fields and methods in addition to the superclass fields and methods.
- Reusability: Inheritance supports the concept of “reusability”, i.e. when we want to create a new class and there is already a class that includes some of the code that we want, we can derive our new class from the existing class.
- By doing this, we are reusing the fields and methods of the existing class.

Inheritance Syntax

- Syntax:
- class Super-class {
- .....
- .....
- }
- class Sub-class extends Super-class {
- …
- }
- Each class has at most one super-class; no multi-inheritance in Java.
- No class is a sub-class of itself.

Example: Super-Class

- class A {

```
int i;
void showi() {
        System.out.println("i: " + i);
        }
}
```

Example: Sub-Class

```
class B extends A {
        int j;
        void showj() {
                System.out.println(“j: " + j);
        }
        void sum() {
                System.out.println("i+j: " + (i+j));
        }
}
```

Example: Testing Class

- class SimpleInheritance {

```
public static void main(String args[]) {
        A  ob1= new A();
        B ob2 = new B();
         ob1.i = 10;
        System.out.println("Contents of a: ");
         ob1.showi();
         ob2.i = 7; ob2.j = 8;
        System.out.println("Contents of b: ");
         ob2.showi(); ob2.showj();
        System.out.println("Sum of I and j in b:");
         ob2.sum();
        }
```

- }

Inheritance and Private Members

- A class may declare some of its members private.
- A sub-class has no access to the private members of its super-class:
- ```
  class A {
      int i;
      private int j;
      void setij(int x, int y) {
          i = x; j = y;
      }
  }
  ```
  ```

Inheritance and Private Members

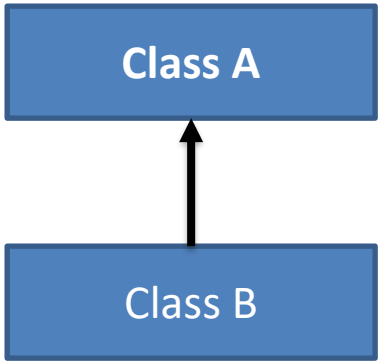
- Class B has no access to the A's private variable j.
- This program will not compile:
- class B extends A {

```
    int total;
    void sum() {
        total = i + j;
    }
```

- }

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.
- Class A is your parent class and
  Class B is your child class
  which inherits the properties
  and behavior of the parent class.



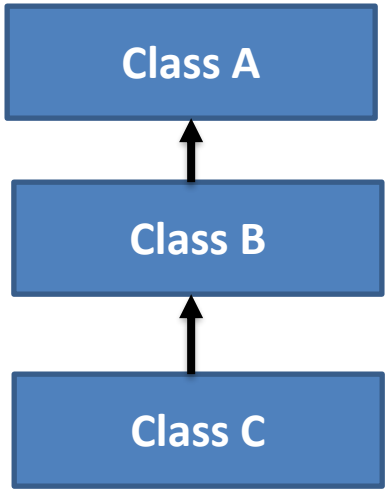
Inheritance

- **Single Inheritance**
- class Animal{

```
    void eat(){
        System.out.println("eating");
    }
}
class Dog extends Animal{
    void bark(){
        System.out.println("barking");
    }
}
class TestInheritance{
    public static void main(String args[]){
        Dog d=new Dog();
        d.bark();
        d.eat();
}
```

Inheritance

- **Multi-level 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.
- 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.



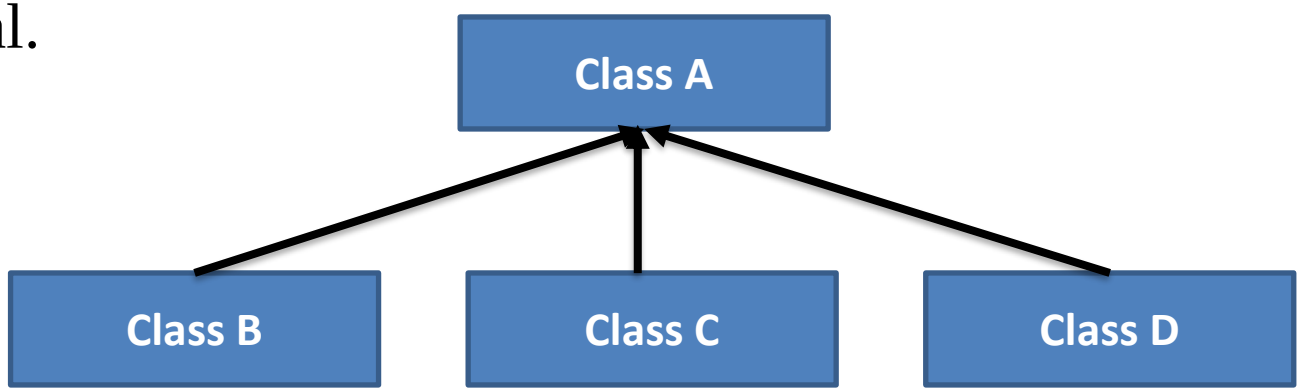
Inheritance

Multi-level Inheritance

```
classAnimal{
 voideat(){
   System.out.println(“eating…”);
 }
}
class Dog extends Animal{
 void bark(){
    System.out.println(“barking…”);
 }
}
class Puppy extends Dog{
  void weep() {
   System.out.println(“weeping…”);
  }
}
class TestInheritance2{
  public static void main(String args[]){
            Puppy d=new Puppy();
            d.weep();
            d.bark();
            d.eat();
   }
}
```

Inheritance

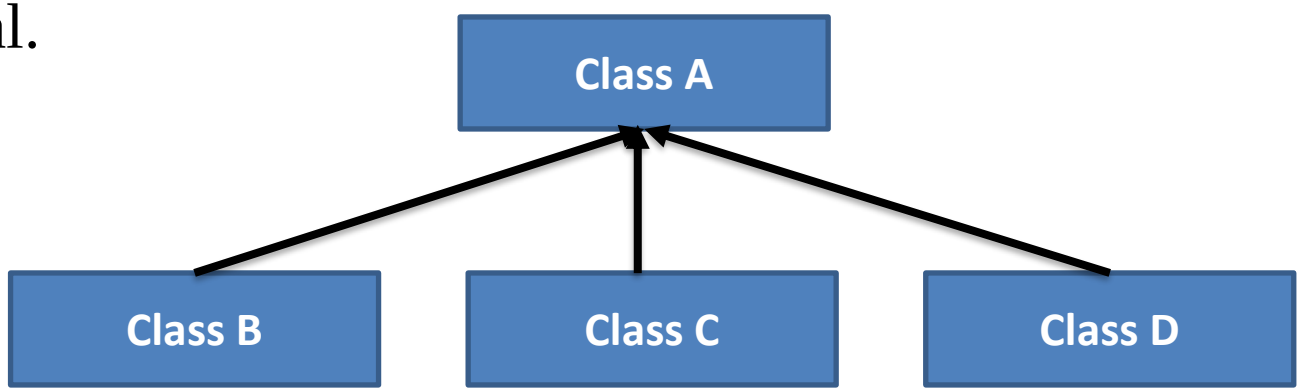
- Hierarchical Inheritance
- When a class has more than one child classes (subclasses) or in other words, more than one child classes have the same parent class, then such kind of inheritance is known as hierarchical.



- In the above flowchart, Class B ,class C and class D are the child classes which are inheriting from the parent class i.e Class A.

Inheritance

- Hierarchical Inheritance
- When a class has more than one child classes (subclasses) or in other words, more than one child classes have the same parent class, then such kind of inheritance is known as hierarchical.



- In the above flowchart, Class B ,class C and class D are the child classes which are inheriting from the parent class i.e Class A.

Inheritance

- Hierarchical Inheritance

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

```
class Cat extends Animal{
    void meow(){
        System.out.println("meowing..");
    }
}
class TestInheritance3{
  public static void main(String args[]){
        Cat c=new Cat();
        c.meow();
        c.eat();
    }
}
```