**OBJECT ORIENTED PROGRAMMING**

***Department of Software Engineering***

### Public Relations, MUET, SZAB Campus Khairpur Mir's - College & University | Facebook - 9,847 PhotosMentor:

#### Engr. Abdul Hafeez Babar [hafeez.babar@muetkhp.edu.pk](mailto:hafeez.babar@muetkhp.edu.pk)

***Mehran University of Engineering and Technology, SZAB Campus Khairpur Mir’s***



INHERITANCE IN JAVA



* Inheritance in Java is a mechanism in which one object acquires all the properties and behaviors of a parent object.
* The idea behind inheritance in Java is that you can create new classes that are built upon existing classes.

When you inherit from an existing class, you can reuse methods and fields of the parent class. Moreover, you can add new methods and fields in your current class also.

##### *Inheritance represents the IS-A relationship which is also known as a parent-child relationship.*



**Why use inheritance in java**

* + For Method Overriding (so runtime polymorphism can be achieved).
  + For Code Reusability.
  + We can reuse the code from the base class.
  + Using inheritance, we can increase features of class or method by overriding.
  + Inheritance is used to use the existing features of class.
  + It is used to achieve runtime polymorphism i.e method overriding.
  + Using inheritance, we can organize the information in a hierarchal form.



### Basic Terminologies

***Class:*** A class is a group of objects which have common properties. It is a template or blueprint from which

objects are created.

***Sub Class/Child Class:*** Subclass is a class which inherits the other class. It is also called a derived class, extended class, or child class.

***Super Class/Parent Class:*** Superclass is the class from where a subclass inherits the features. It is also called a base class or a parent class.

***Reusability:*** As the name specifies, reusability is a mechanism which facilitates you to reuse the fields and methods of the existing class when you create a new class. You can use the same fields and methods already defined in the previous class.



# How is Inheritance implemented/achieved in Java?

* Inheritance in Java can be implemented or achieved by using two keywords:

1. **extends:** extends is a keyword that is used for developing the inheritance between two classes and two interfaces.

Note that a class always extends another class. An interface always extends another interface and can extend more than one interface.

1. **implements:** implements keyword is used for developing the inheritance between a class and interface. A class always implements the interface.



# Role of Constructor in Java Inheritance

* + The role of constructor in inheritance is that the constructor in the superclass is responsible for building an

object of superclass and the constructor of subclass builds an object of subclass.

* + When the constructor of a subclass is called during the object creation, by default, it calls the default

constructor of superclass.

* + The superclass constructor can be called using keyword “super” from the subclass constructor. It must be the first statement in a constructor.
  + We can call other constructors of same class using this keyword but you cannot call a subclass constructor from the superclass constructor.



# Important Rules of Java Inheritance

1. We cannot assign a superclass to the subclass.
2. We cannot extend the final class.
3. A class cannot extend itself.
4. One class can extend only a single class.
5. We cannot extend a class having a private constructor but if we have private constructor as well as public constructor then we can extend superclass to subclass. In this case, the only public constructor will work.
6. If we assign subclass reference to superclass reference, it is called dynamic method dispatch in java.
7. Constructor, Static initialization block (SIB), and Instance initialization block (IIB) of the superclass cannot be inherited to its subclass but they are executed while creating an object of the subclass.
8. A static method of superclass is inherited to the subclass as a static member and non-static method is inherited as a non-static member only



#### Syntax:

class Subclass-name extends Superclass-name

{

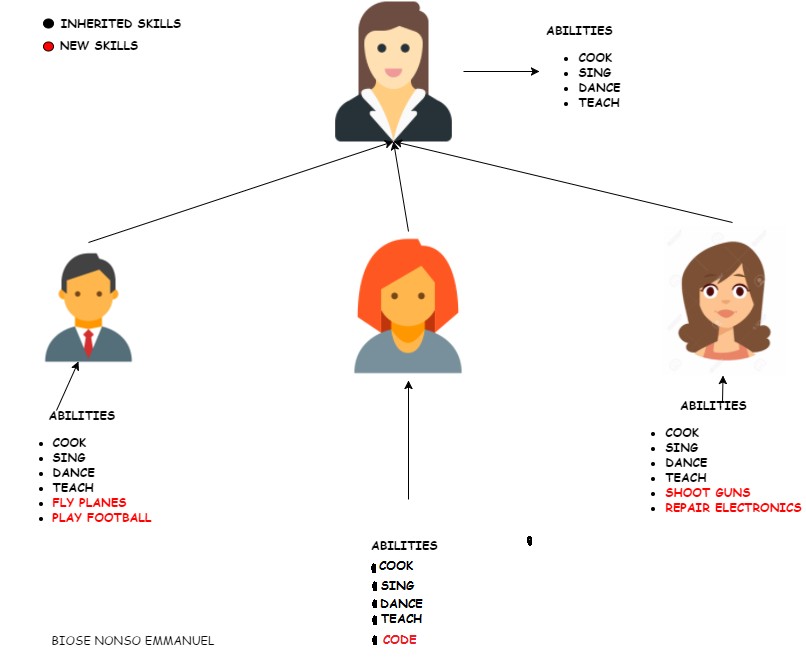
//methods and fields

}

* The extends keyword indicates that you are making a new class that derives from an existing class. The meaning of "extends" is to increase the functionality.
  + In the terminology of Java, a class which is inherited is called a parent or superclass, and the new class is called child or subclass.



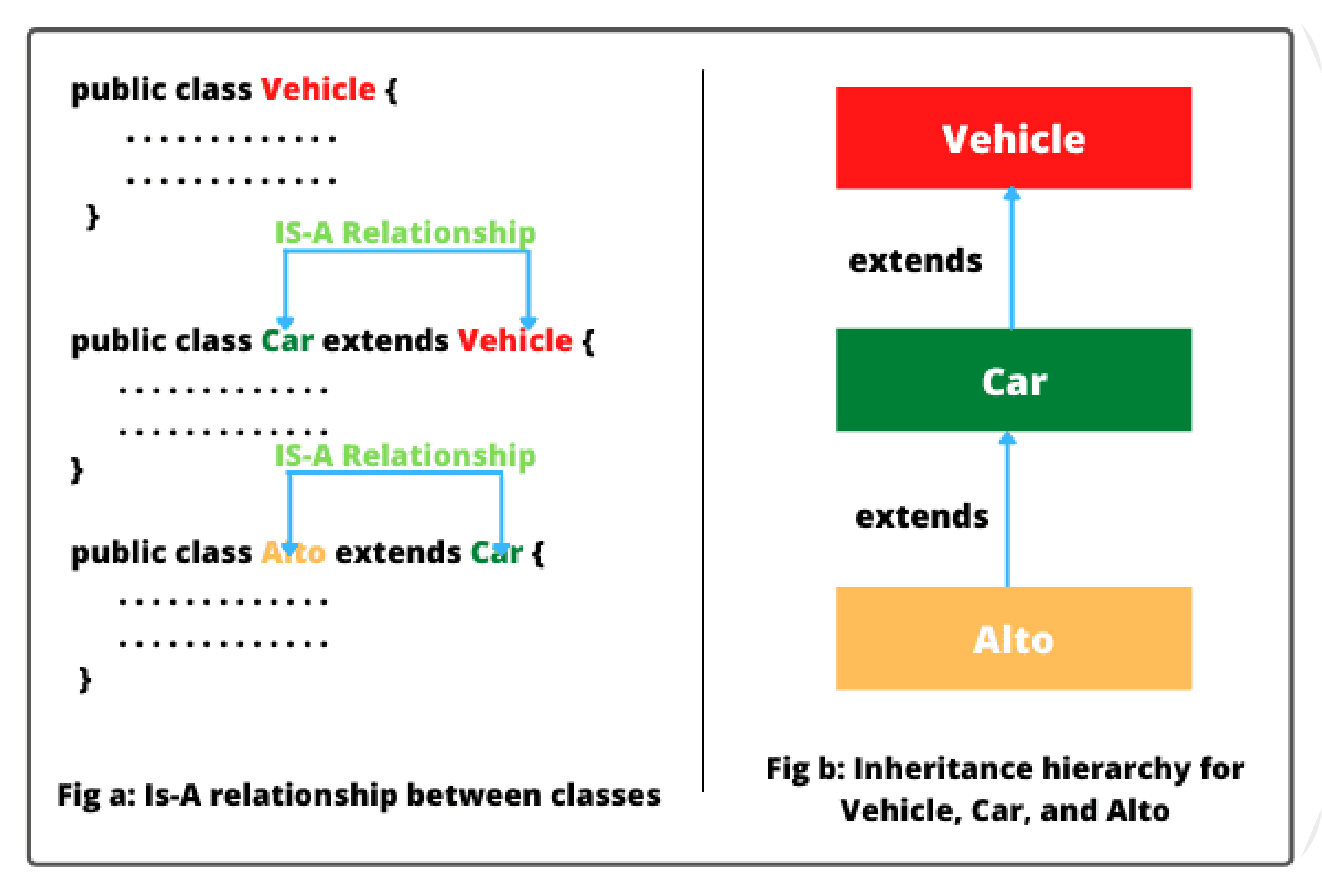
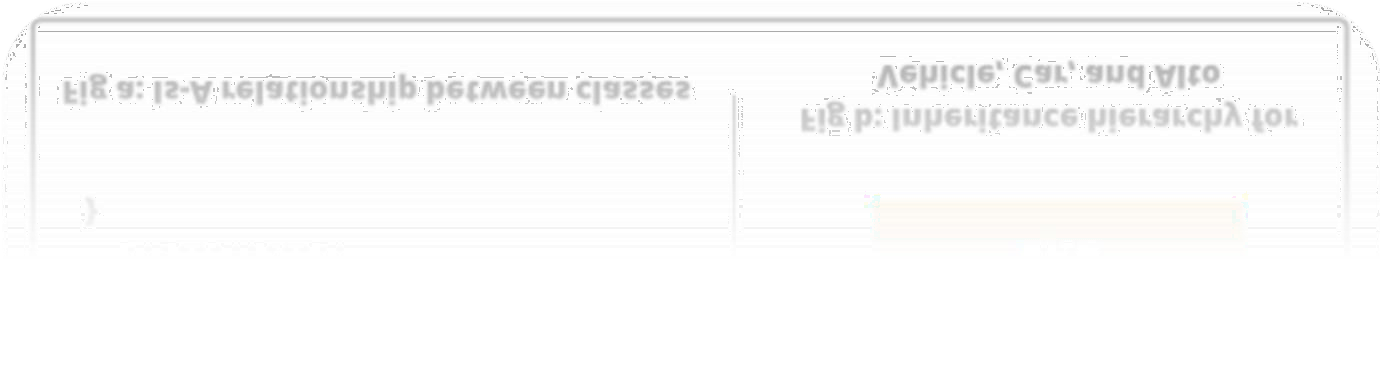
* Inheritance is one of the key features of OOP that allows us to create a new class from an existing class.
* The new class that is created is known as subclass (child or derived class) and the existing class from where the child class is derived is known as superclass (parent or base class).
* Inheritance is a process by which a child class acquires all the properties and behaviors of the parent class.



#### Inheritance

**Engr. A. Hafeez Babar (MUET-SZAB)** [**hafeez.babar@muetkhp.edu.pk**](mailto:hafeez.babar@muetkhp.edu.pk)

A car is a type of Vehicle, so inheritance hierarchy might begin from the Vehicle class as follows:



public class Vehicle { }

public class Car extends Vehicle { }

public class Alto extends Car { }

* + Vehicle is the parent class (superclass) of Car.
  + Car is the child class (subclass) of Vehicle.
  + Car is the parent class of Alto.
  + Alto is the child class of Vehicle.
  + Car inherits from Vehicle.
  + Alto inherits from both Vehicle and Car.
  + Alto is derived from Car.
  + Car is derived from Vehicle.
  + Alto is derived from Vehicle.
  + Alto is a subtype of both Vehicle and Car.

**Engr. A. Hafeez Babar (MUET-SZAB)** [**hafeez.babar@muetkhp.edu.pk**](mailto:hafeez.babar@muetkhp.edu.pk)



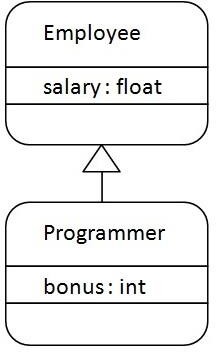
##### *In the terms of IS-A relationship, the following statements are true:*

“Car extends Vehicle” means “Car IS-A Vehicle.” “Alto extends Car” means “Alto IS-A Car.”

And we can also say that “Alto IS-A Vehicle”.



**Example:**

class Employee{

float salary=40000;

}

class Programmer extends Employee{ int bonus=10000;

public static void main(String args[]){

Programmer p=new Programmer(); System.out.println("Programmer salary is:"+p.salary); System.out.println("Bonus of Programmer is:"+p.bonus);

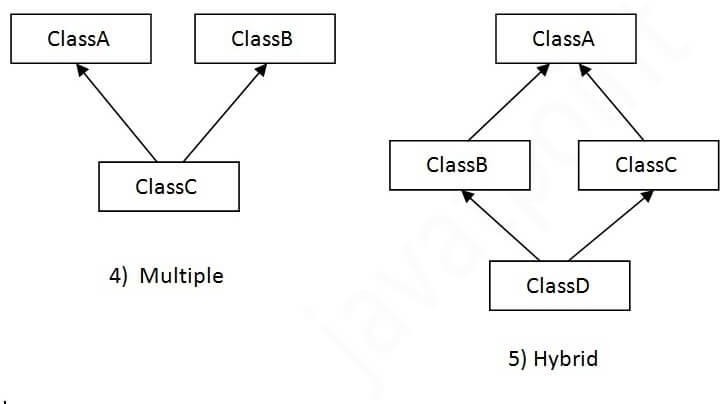
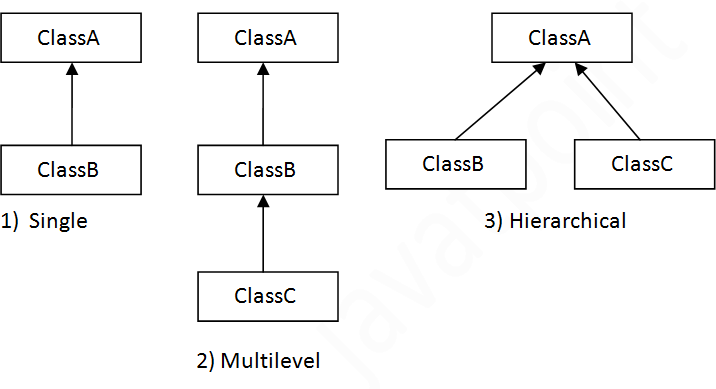
}

}



# Types of inheritance in java

* + - On the basis of class, there can be three types of inheritance in java: single, multilevel and hierarchical.
    - In java programming, multiple and hybrid inheritance is supported through interface only.



# Example: Single Inheritance

Dog class inherits the Animal class, so there is the 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();

}}

#### Output:

barking... eating...

# Example: Multilevel Inheritance

When there is a chain of inheritance, it is known as multilevel inheritance.

* + - * BabyDog class inherits the Dog class which again inherits the Animal class, so there is a multilevel inheritance.

class Animal{



void eat(){System.out.println("eating...");}

}

class Dog extends Animal{

void bark(){System.out.println("barking...");}

}

class BabyDog extends Dog{

void weep(){System.out.println("weeping...");}

}

class TestInheritance2{

public static void main(String args[]){ BabyDog d=new BabyDog(); d.weep();

d.bark();

d.eat();

}}

#### Output:

weeping... barking... eating...

# Example: Hierarchical Inheritance

When two or more classes inherits a single class, it is known as hierarchical inheritance. Dog and Cat classes inherits the Animal class, so there is 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();

//c.bark();//C.T.Error

}}

#### Output:

meowing... eating...

# Why multiple inheritance is not supported in java?

To reduce the complexity and simplify the language, multiple inheritance is not supported in java.

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 you call it from child class object, there will be ambiguity to call the method of A or B class.

***Note:*** Since compile-time errors are better than runtime errors, Java renders compile-time error if you inherit 2 classes. So whether you have same method or different, there will be

class A{

void msg(){System.out.println("Hello");}

}

class B{

void msg(){System.out.println("Welcome");}

}

class C extends A,B{//suppose if it were

public static void main(String args[]){

C obj=new C();

obj.msg();//Now which msg() method would be invoked?

compile time error. }



}

**OUTPUT:** Compile Time Error



**super Keyword in Java Inheritance**

#### Example: super Keyword in

**Inheritance**

class Animal {

// method in the superclass public void eat() { System.out.println("I can eat");

}

}

// Dog inherits Animal class Dog extends Animal {



// overriding the eat() method @Override

public void eat() {

// call method of superclass super.eat();

System.out.println("I eat dog food");

}

// new method in subclass public void bark() {

System.out.println("I can bark");

}

}

class Main {

public static void main(String[] args) {

// create an object of the subclass

Dog labrador = new Dog();

// call the eat() method labrador.eat(); labrador.bark();

}

}

#### OUTPUT:

I can eat

I eat dog food I can bark

# protected Members in Inheritance

if a class includes protected fields and methods, then these fields and methods are accessible from the subclass of the class.

#### Example: protected Members in Inheritance

class Animal {

protected String name;

protected void display() {

System.out.println("I am an animal.");

class Main {

public static void main(String[] args) {

// create an object of the subclass

Dog labrador = new Dog();

}

} // access protected field and method

// using the object of subclass

class Dog extends Animal {



public void getInfo() { System.out.println("My name is " + name);

}

}

labrador.name = "Rocky"; labrador.display();

labrador.getInfo();

}

}

#### Output

I am an animal. My name is Rocky



# Inheritable Members of Classes and Interfaces in Java

## Types of Fields/Members Classes Interfaces Instance variables Yes Not applicable Static variables Yes Only constants Abstract methods Yes Yes

Instance methods Yes Java 8, default methods

Static methods Yes Java 8, Inherited no, Accessible yes Constructor No Not applicable

Initialization blocks No Not applicable



**The instanceof Keyword**

* use the instanceof operator to check determine whether Mammal is actually an Animal, and dog is actually an Animal.

interface Animal{}

class Mammal implements Animal{} public class Dog extends Mammal {

public static void main(String args[]) { Mammal m = new Mammal();

Dog d = new Dog();

System.out.println(m instanceof Animal); System.out.println(d instanceof Mammal); System.out.println(d instanceof Animal);

#### Output

true true true

}



}



#### Key Points to Remember

1. Building hierarchical relationships between related classes is called inheritance. It is also known as “IS-

A” relationship.

1. It is the most important and powerful feature of OOPs concepts in Java that facilitates to reuse of code written in a class inside other classes.
2. It helps to organize information in a hierarchical form that is easy to understand.
3. There are five types of inheritance in Java. They are single-level, multilevel, hierarchical, multiple, and hybrid inheritance. The usable forms of inheritance are single inheritance, hierarchical inheritance, and multilevel inheritance.
4. Java does not support multiple inheritance through classes.
5. The main advantage of using inheritance is that it allows code to be reused.

