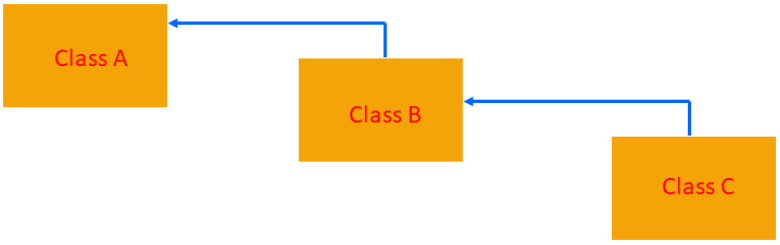
UNIT III – INHERITANCE AND POLYMORPHISM

TYPES OF INHERITANCE IN JAVA (SINGLE, MULTILEVEL, HIERARCHICAL):

INHERITANCE:

- A mechanism in which one object acquires all the properties and behaviors of parent object.
- That is, we can create new classes that are built upon existing classes.
- When we inherit from an existing class, we can reuse methods and fields of parent class, and we can add new methods and fields also.
- Inheritance represents the is-a relationship, also known as parent-child relationship.
- Defining a new class based on an existing class is called derivation.
- The derived class is also called the direct subclass of the base or super class.
- We can also derive classes from the derived class and so on.




```
class B extends A
{
        //definition of class B
}
```

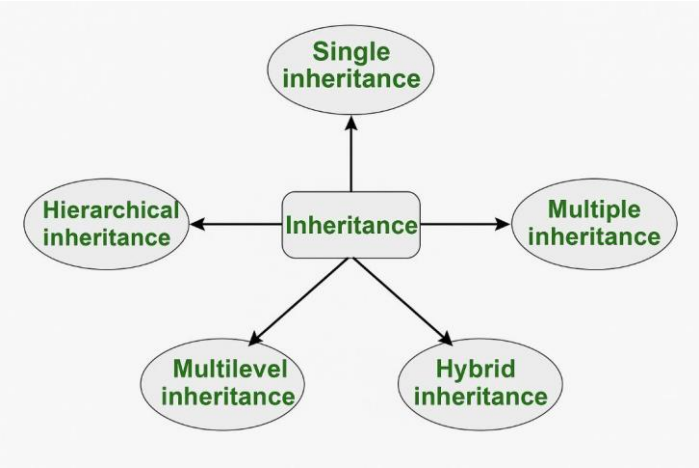
- The **extends** keyword indicates that we are making a new class that derives from an existing class.
- In the terminology of Java, a class that is inherited is called a super class. The new class is called a subclass
- The class B can have additional members in addition to the inherited members of class A.

PURPOSE OF INHERITANCE:

- **Code Reusability:** Reduces redundancy by allowing subclasses to reuse methods and fields from their superclass.

- **Extensibility:** Enables the creation of new classes based on existing ones, extending their functionality.
- **Polymorphism:** Allows objects of different subclasses to be treated interchangeably through their common superclass interface.
- **Method Overriding:** Provides a way for subclasses to provide a specific implementation for a method already defined in its superclass.

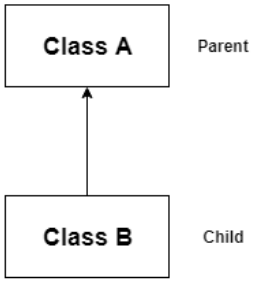
TYPES:



- **Single Inheritance:** A class inherits from only one superclass.
- **Multilevel Inheritance:** A chain of inheritance where a class inherits from a class, which in turn inherits from another class (e.g., Class C extends Class B, and Class B extends Class A).
- **Hierarchical Inheritance:** Multiple subclasses inherit from a single superclass.
- **Multiple Inheritance:** Not supported by Java.
- **Hybrid Inheritance:** A combination of two or more types of inheritance.

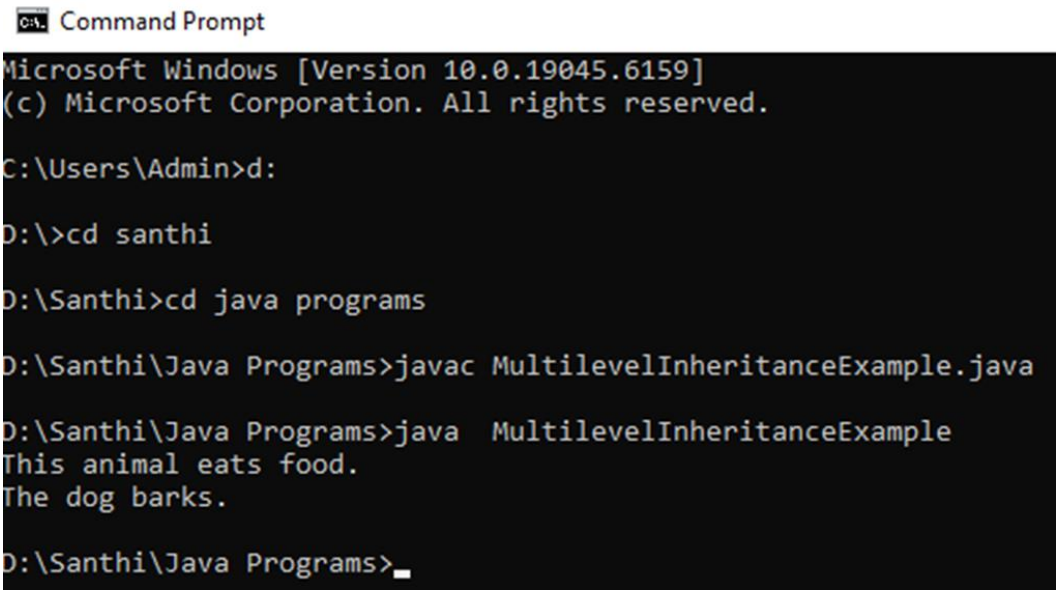
SINGLE INHERITANCE

```
class Animal
{
    void eat()
   {
        System.out.println("This animal eats food.");
    }
}
class Dog extends Animal
```




```
{
    void bark()
   {
        System.out.println("The dog
barks.");
    }
}
public class InheritanceExample
{
    public static void main(String[] args)
    {
        // Create an object of the Dog class
        Dog myDog = new Dog();
        // Call the 'eat()' method inherited from the Animal class
        myDog.eat();
        // Call the 'bark()' method specific to the Dog class
        myDog.bark();
    }
}
```

Output:



MULTI LEVEL INHERITANCE

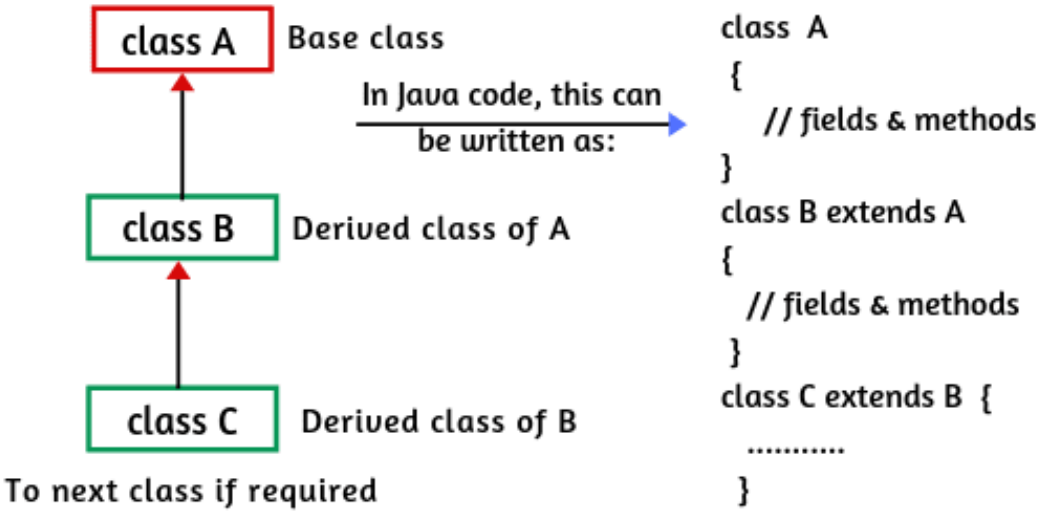


Fig: Multilevel Inheritance in Java

```
class Animal
{
    void eat()
   {
        System.out.println("This animal eats food.");
    }
}
class Dog extends Animal
{
    void bark()
   {
        System.out.println("The dog barks.");
    }
}
class BabyDog extends Dog
{
    void weep()
   {
        System.out.println("The babydog weeps.");
    }
}
public class MultilevelInheritanceExample
```

```
    {
        public static void main(String[] args)
        {
            // Create an object of the BabyDog class
            BabyDog myDog = new BabyDog();
            // Call the 'eat()' method inherited from the Animal class
            myDog.eat();
            // Call the 'bark()' method specific to the Dog class
            myDog.bark();
            // Call the 'weep()' method specific to the BabyDog class
            myDog.weep();
        }
    }
```

Output:

Command Prompt

```
D:\Santhi\Java Programs>javac MultilevelInheritanceExample.java

D:\Santhi\Java Programs>java  MultilevelInheritanceExample
This animal eats food.
The dog barks.
The babydog weeps.

D:\Santhi\Java Programs>
```

HIERARCHICAL INHERITANCE:

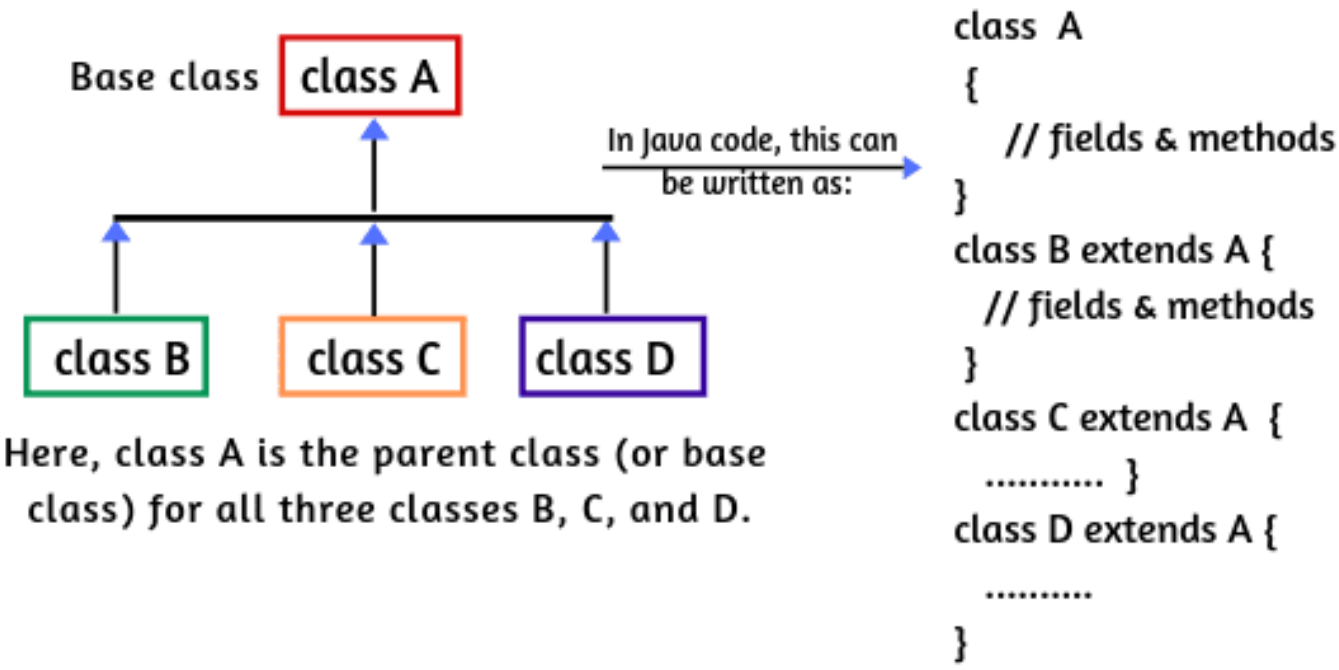
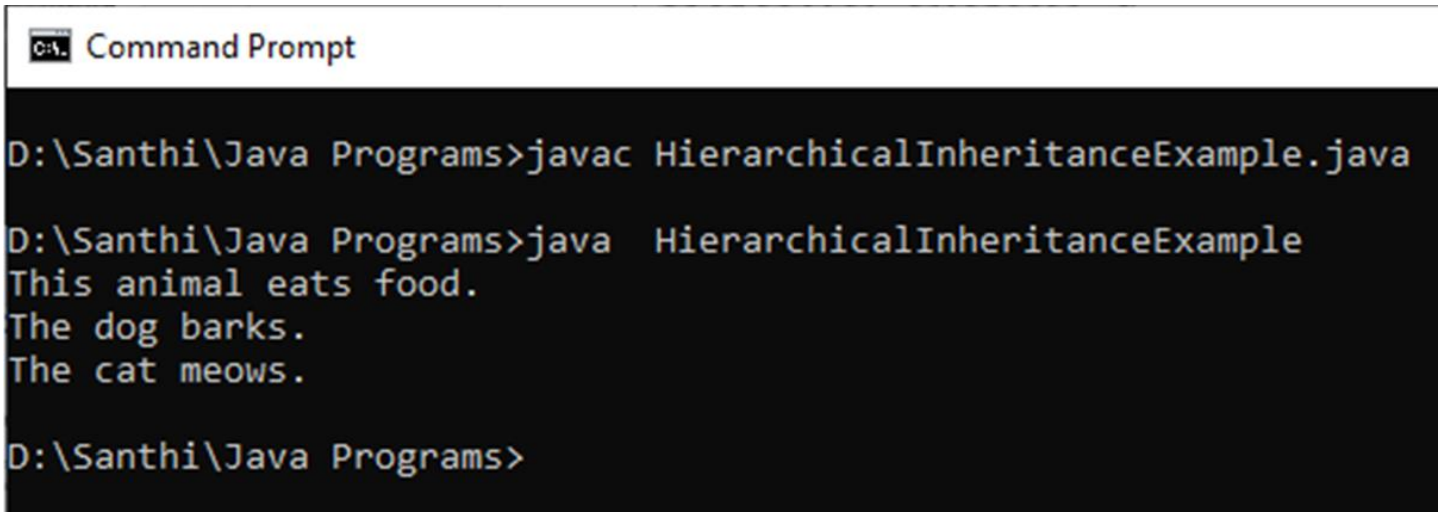


Fig: Hierarchical Inheritance in Java

```
class Animal
{
    void eat()
   {
        System.out.println("This animal eats food.");
    }
}
class Dog extends Animal
{
    void bark()
   {
        System.out.println("The dog barks.");
    }
}
class Cat extends Animal
{
    void meow()
   {
        System.out.println("The cat meows.");
```

```
    }
}
public class HierarchicalInheritanceExample
{
    public static void main(String[] args)
    {
        Dog myDog=new Dog();
        Cat myCat=new Cat();
        // Call the 'eat()' method inherited from the Animal class
        myCat.eat();
        // Call the 'bark()' method specific to the Dog class
        myDog.bark();
        // Call the 'meow()' method specific to the Cat class
        myCat.meow();
    }
}
```

Output:



PROGRAM FOR STUDENT MANAGEMENT SYSTEM:

```
// Parent Class
class Person
{
    String name;
    int age;
    void setPersonInfo(String n, int a)
```

```
    {
        name = n;
        age = a;
    }
    void displayPersonInfo()
    {
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
    }
}
class Student extends Person                    // Child Class 1
{
        String studentId;
        String course;
        void setStudentInfo(String n, int a, String id, String c)
        {
                setPersonInfo(n, a); // Call parent method
                studentId = id;
                course = c;
        }
    void displayStudentInfo()
    {
        displayPersonInfo();
        System.out.println("Student ID: " + studentId);
        System.out.println("Course: " + course);
    }
}

// Child Class 2
class GraduateStudent extends Student
{
    String thesisTopic;
    void setGraduateInfo(String n, int a, String id, String c, String topic)
```

```
    {
        setStudentInfo(n, a, id, c);
        thesisTopic = topic;
    }
    void displayGraduateInfo()
    {
        displayStudentInfo();
        System.out.println("Thesis Topic: " + thesisTopic);
    }
}
public class StudentManagementSystem
{
    public static void main(String[] args)
    {
        System.out.println("=== Undergraduate Student ===");
        Student s1 = new Student();
        s1.setStudentInfo("John Doe", 20, "S101", "Computer Science");
        s1.displayStudentInfo();
        System.out.println("\n=== Graduate Student ===");
        GraduateStudent gs1 = new GraduateStudent();
        gs1.setGraduateInfo("Alice Smith", 24, "G201", "Data Science",
                                "Machine Learning in IoT");
        gs1.displayGraduateInfo();
    }
}
```

Output:

Command Prompt

```
D:\San>javac StudentManagementSystem.java

D:\San>java StudentManagementSystem
=== Undergraduate Student ===
Name: John Doe
Age: 20
Student ID: S101
Course: Computer Science

=== Graduate Student ===
Name: Alice Smith
Age: 24
Student ID: G201
Course: Data Science
Thesis Topic: Machine Learning in IoT

D:\San>
```