SNS COLLEGE OF ENGINEERING

Coimbatore-107
An Autonomous Institution

COURSE NAME : 23CSB201 & Object Oriented Programming

II YEAR/ III SEMESTER

UNIT – II INHERITANCE, PACKAGES, INTERFACE

Topic: Inheritance – BasicS, Types

Dr.P.Poonkodi
Assistant Professor(SG)
Department of Computer Science and Technology

Introduction

- Inheritance is a process where one class acquires the properties (methods and attributes) of another class
- It allows a class (child or subclass) to inherit the properties and behaviors of another class (parent or superclass)
- This promotes **code reusability, hierarchy, and better organization** in a program
- Java supports several types of inheritance, each serving a different purpose
- However, **multiple inheritance using classes** is not supported in Java to avoid the **diamond problem**, but it can be achieved using interfaces

Key Features

1.Code Reusability: Avoids duplication by reusing existing class properties.

2.Hierarchy: Establishes a relationship between base and derived classes.

3.Extensibility: Allows modification of behavior without changing the base class.

4.Improved Maintainability: Enhances program structure and reduces redundancy.

Key Features

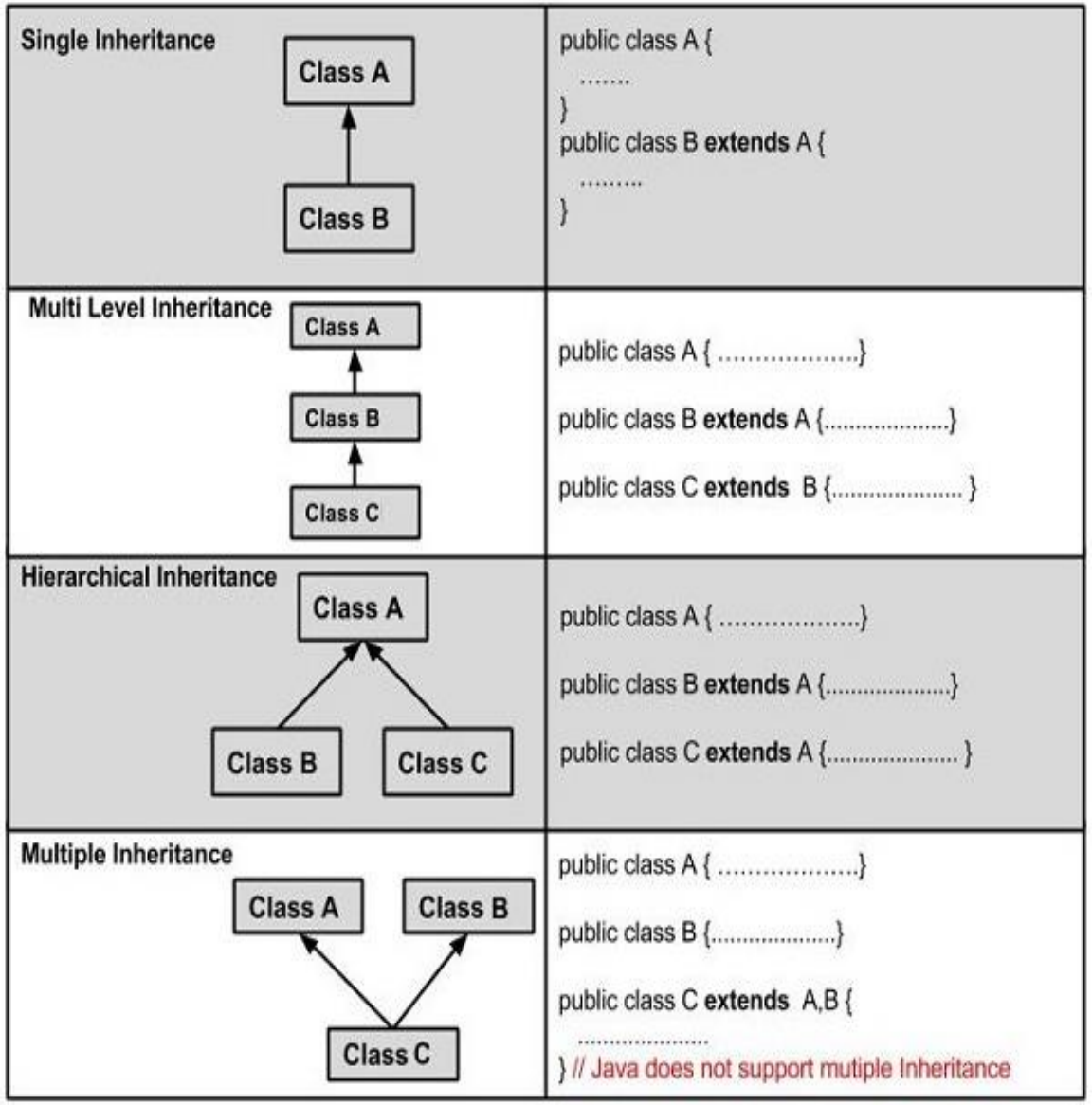
1.Code Reusability: Avoids duplication by reusing existing class properties.

2.Hierarchy: Establishes a relationship between base and derived classes.

3.Extensibility: Allows modification of behavior without changing the base class.

4.Improved Maintainability: Enhances program structure and reduces redundancy.

Types of Inheritance



1. Single Inheritance
2. Multilevel Inheritance
3. Hierarchical Inheritance
4. Multiple Inheritance (via Interfaces)
5. Hybrid Inheritance (via Interfaces)

Single Inheritance

- **Definition**:

  only one base class and one derived class

Example:

  A Student class inherits from a Person class

- **Use Case**:

  Simple hierarchical relationships

Example

```
class Person
{
    String name;

    void showName()
   {
        System.out.println("Name: " + name);
    }
}
```

Cond.,

```
class Student extends Person
{
    int studentId;

    void showStudentId()
    {
        System.out.println("Student ID: " + studentId);
    }
}
```

Cond.,

```
public class SingleInheritanceExample
{
    public static void main(String[] args)
    {
        Student student = new Student();
        student.name = "Alice";
        student.studentId = 101;
        student.showName();
        student.showStudentId();
    }
}
```

Multilevel Inheritance

- **Definition:**
  - ✓ A base class is inherited to a derived class and that derived class is further inherited to another derived class
  - ✓ Multilevel inheritance involves multiple base classes
- **Example:** Person → Student → GraduateStudent
- **Use Case:** Extending functionality over multiple levels.

Example

```
class Person
{
    String name;

    void showName()
  {
        System.out.println("Name: " + name);
    }
}
```

Cond.,

```
class Student extends Person
{
    int studentId;

    void showStudentId()
    {
        System.out.println("Student ID: " + studentId);
    }
}
```

Cond.,

```
class GraduateStudent extends Student
{
    String specialization;

    void showSpecialization()
    {
        System.out.println("Specialization: " + specialization);
    }
}
```

Cond.,

```
public class MultilevelInheritanceExample
{
    public static void main(String[] args)
    {
        GraduateStudent gradStudent = new GraduateStudent();
        gradStudent.name = "Bob";
        gradStudent.studentId = 102;
        gradStudent.specialization = “CST";
```

Cond.,

```
        gradStudent.showName();
        gradStudent.showStudentId();
        gradStudent.showSpecialization();
    }
}
```

Hierarchical Inheritance

Definition:

only one base class and multiple derived classes

Example:

Person is a parent class of Student and Teacher

Use Case:

When multiple entities share common attributes

Example

```
class Person
{
    String name;

    void showName()
   {
        System.out.println("Name: " + name);
    }
}
```

Cond.,

```
class Student extends Person {
    int studentId;

    void showStudentId() {
        System.out.println("Student ID: " + studentId);
    }
}
```

Cond.,

```
class Teacher extends Person
{
    String subject;

    void showSubject()
   {
        System.out.println("Teaches: " + subject);
    }
}
```

Cond.,

```
public class HierarchicalInheritanceExample
{
    public static void main(String[] args)
    {
        Student student = new Student();
        student.name = "Charlie";
        student.studentId = 103;
        student.showName();
        student.showStudentId();
```

Cond.,

```
        Teacher teacher = new Teacher();
        teacher.name = "Dr. Smith";
        teacher.subject = "Mathematics";
        teacher.showName();
        teacher.showSubject();
    }
}
```

Multiple Inheritance

Definition:

Java does not support multiple inheritance with classes, but it allows multiple inheritance using interfaces.

Example:

A Student class can implement both Sports and Academics interfaces.

Use Case:

When a class needs to inherit behaviors from multiple sources

Example

```
interface Sports
{
    void playSport();
}

interface Academics
{
    void study();
}
```

Class with interface

```
class Student implements Sports, Academics
{
    public void playSport()
   {
        System.out.println("Student plays football.");
    }
    public void study()
   {
        System.out.println("Student studies computer science.");
    }
}
```

Main class

```
public class MultipleInheritanceExample
{
    public static void main(String[] args)
    {
        Student student = new Student();
        student.playSport();
        student.study();
    }
}
```

Hybrid Inheritance

Definition:

Combination of two or more types of inheritance, implemented using interfaces

Example:

A StudentAthlete class inherits academic features from Academics and sports skills from Sports

Use Case:

When a system requires a mix of different inheritance types

Example

```
interface Sports
{
    void playSport();
}

interface Academics
{
    void study();
}
```

Example

```
class Person
{
    String name;

    void showName()
   {
        System.out.println("Name: " + name);
    }
}
```

Example

```
// Hybrid Inheritance: Person + Academics + Sports
class StudentAthlete extends Person implements Academics, Sports
{
    public void playSport()
   {
        System.out.println("StudentAthlete plays basketball.");
    }
    public void study()
   {
        System.out.println("StudentAthlete studies data science.");
    }
}
```

Example

```
public class HybridInheritanceExample
{
    public static void main(String[] args)
    {
        StudentAthlete studentAthlete = new StudentAthlete();
        studentAthlete.name = "David";
        studentAthlete.showName();
        studentAthlete.study();
        studentAthlete.playSport();
    }
}
```

References

- Java : the complete Reference (Eleventh Edition), Herbert Schildt, 2018.


Thank You!