Shaheed Zulfikar Ali Bhutto
Institute of Science & Technology

Object-Oriented Programming Techniques

INHERITANCE, SUBCLASSES AND SUPER CLASSES

Session Outline

At the end of the this session, you will able to understand:

1. Inheritance in OOP
2. Subclasses
3. Superclasses
4. Classification by generalisation and specialisation

Topics:

1. Inheritance
2. Programming inheritance in Classes

Introduction to Inheritance

Generally, inheritance means:

"***Inheritance is the practice of passing on private property, titles, debts, rights, and obligations***" [1]

In Computer science, it can be:

1. **The mechanism** of basing an object or class upon another object or class, retaining similar implementation. Also defined as deriving new classes from existing ones and forming them into a hierarchy of classes [2].
2. **The mechanism** in which one class acquires the property of another class. For example, a child inherits the traits of his/her parents. With **inheritance**, we can reuse the fields and methods of the existing class. Hence, **inheritance** facilitates Reusability and is an important concept of OOPs [3].

Inheritance (cont.)

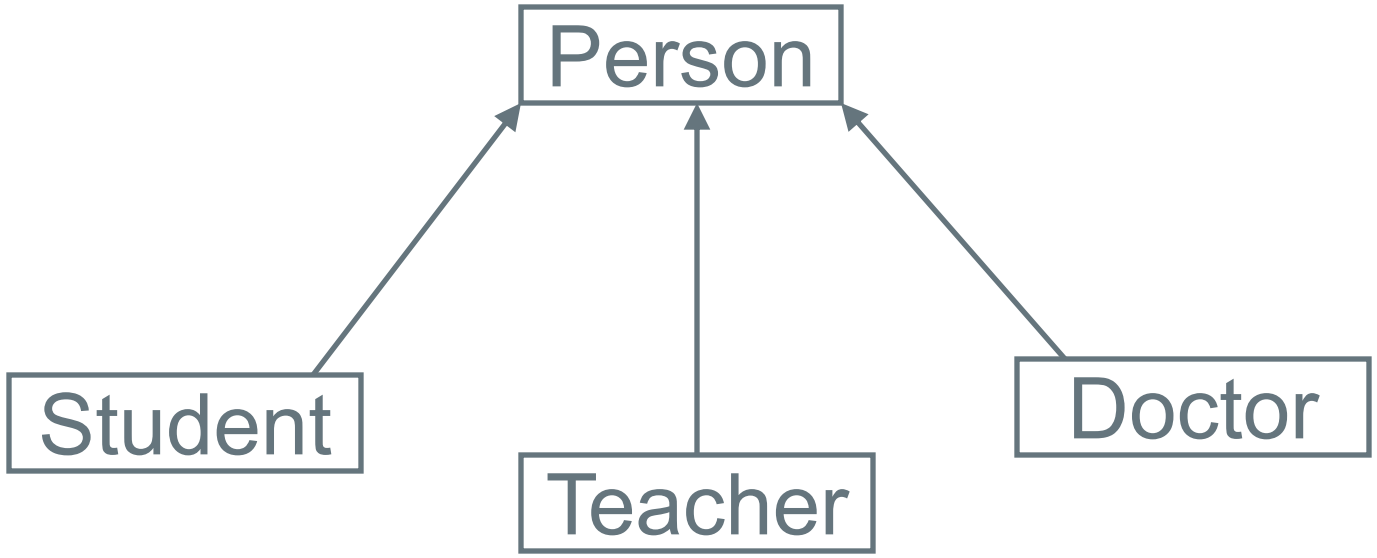
3. **The mechanism** in which new class of objects can be created conveniently by **inheritance**—the new class (called the **subclass**) starts with the characteristics of an existing class (called the **superclass**), possibly customizing them and adding unique characteristics of its own. In our car analogy, an object of class "`Convertible`" certainly *is an* object of the more *general* class "`SportsCar`" but more *specifically*, the roof can be raised or lowered [4].
4. **The mechanism** by a child class inherits characteristics of its parents class. Besides inherited characteristics, a child may have its own unique characteristics [5].

Inheritance in Classes

1. If a class B (`Convertible`) inherits from class A (`SportsCar`) then it contains all the characteristics (information structure and behavior) of class A (`SportsCar`).
2. The parent class (`SportsCar`) is called ***base* class** and the child class (`Convertible`) is called ***derived* class**
3. Besides inherited characteristics, derived class may have its own unique characteristics (the roof can be lowered and raised).

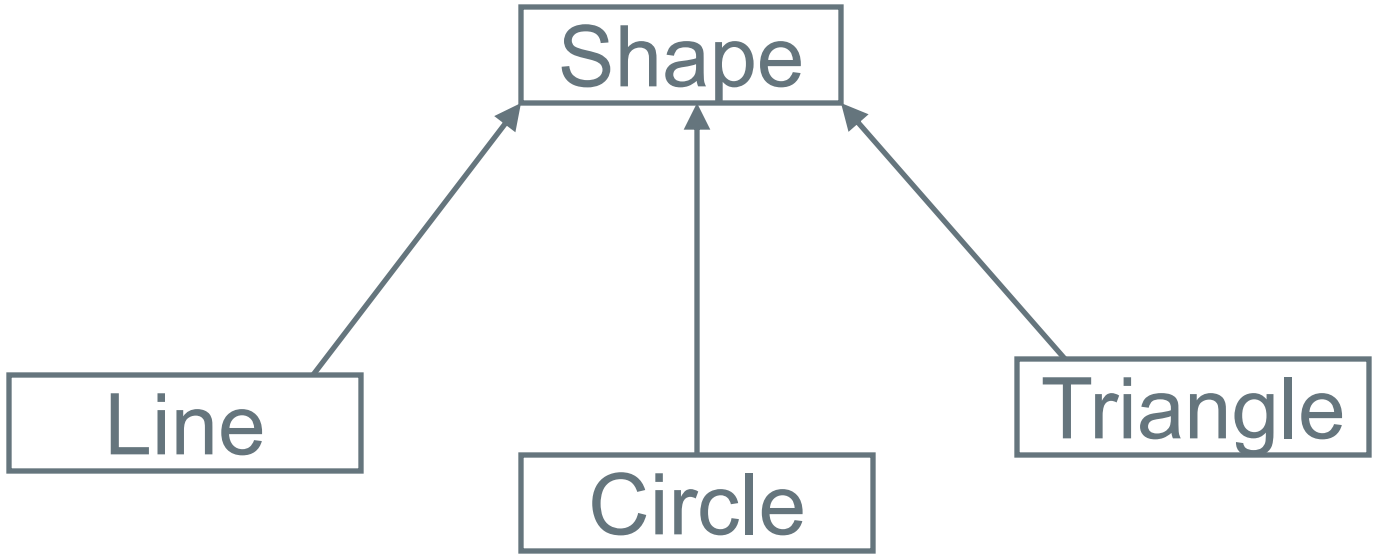
Examples - Person

1. Assume the Person class is generalized containing characteristics of **walk**, **eat**, and **age, name, gender.**
2. The three (3) derived classes can be:



Examples - Shape

1. Assume the Shape class is generalized containing characteristics of **computeArea**, **point, color, setColor, and rotate**.
2. The three (3) derived classes can be:



Inheritance - Advantages

1. One of the key **benefits** of **inheritance** is to minimize the amount of duplicate code in an application by sharing common code amongst several subclasses, where equivalent code exists in two related classes.
2. This also tends to result in a better organization of code and *smaller*, *simpler compilation units* [6].
3. Inheritance promotes reusability. When a class inherits or derives another class, it can access all the functionality of inherited class.
4. Reusability enhance reliability. The base class code will be already tested and debugged.

Inheritance - Advantages

5. As the existing code is reused, it leads to less development and maintenance costs.
6. Inheritance makes the sub classes follow a standard interface.
7. Inheritance helps to reduce code redundancy and supports code extensibility.
8. Inheritance facilitates creation of class libraries.
9. Inheritance allows us to inherit all the properties of base class and can access all the functionality of inherited class. It implements reusability of code.

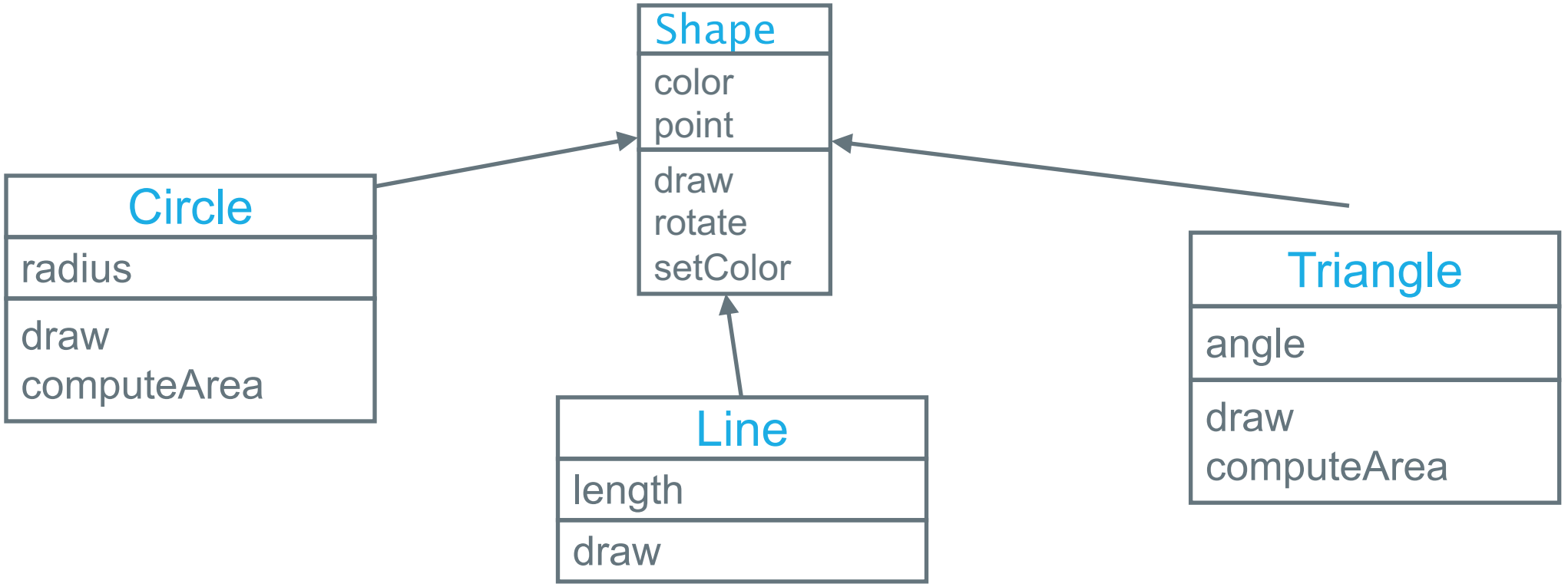
Inheritance - Disadvantages

Besides the advantages, there are some trade-offs as well:

1. Inherited functions work slower than normal function as there is indirection.
2. Improper use of inheritance may lead to wrong solutions.
3. Often, data members in the base class are left unused which may lead to memory wastage.
4. Inheritance increases the coupling between base class and derived class. A change in base class will affect all the child classes.

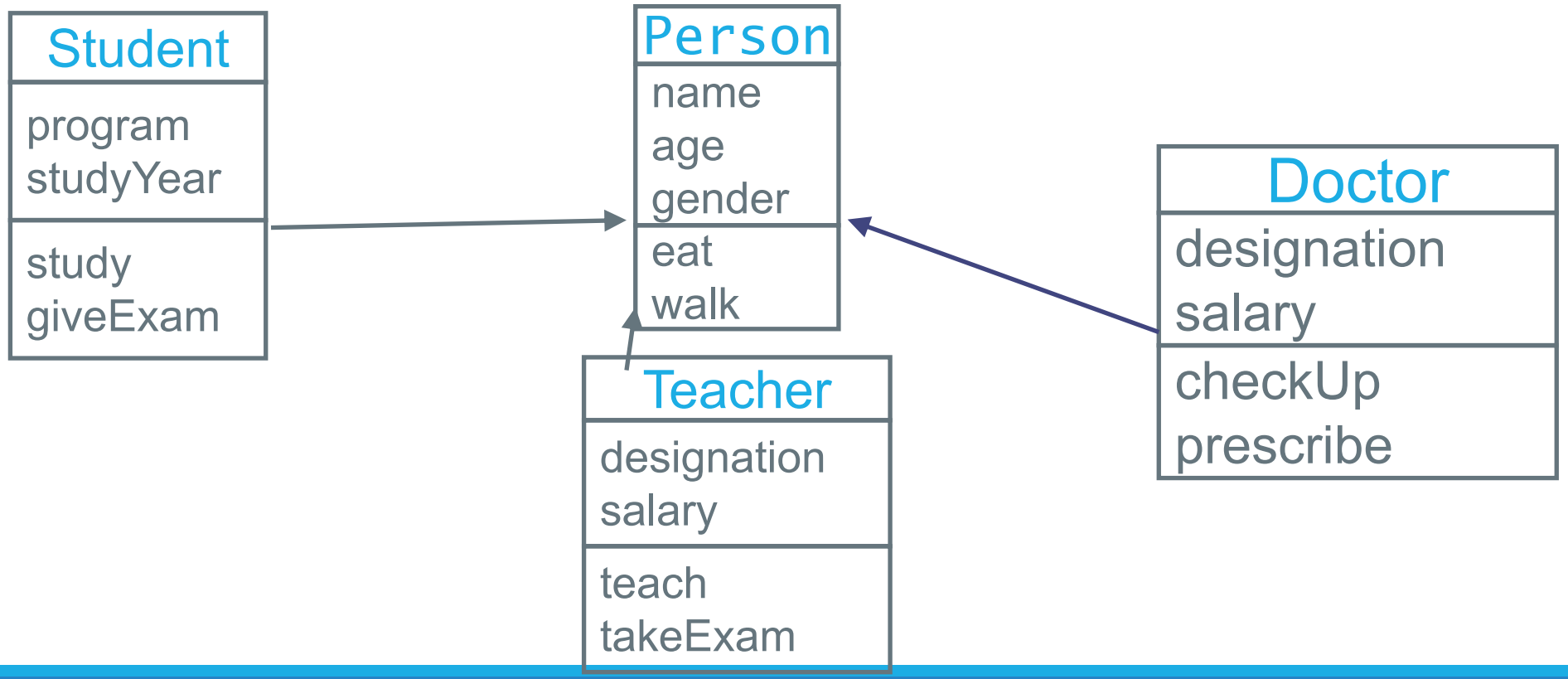
Inheritance – Resusability 1

Consider the following example Shape example



Inheritance – Resusability 2

Consider the following example `Person` example



Inheritance – Types

Below are the different types of inheritance which is supported by **Java**.

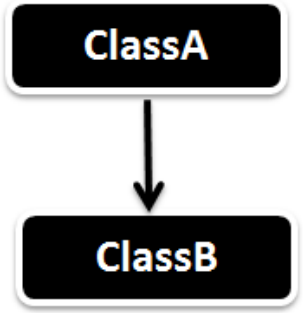
1. Single Inheritance
2. Multiple Inheritance **(Through Interface)**
3. Multilevel Inheritance
4. Hierarchical Inheritance
5. Hybrid Inheritance **(Through Interface)**

Lets see about each one of them one by one.

Inheritance – Single Inheritance

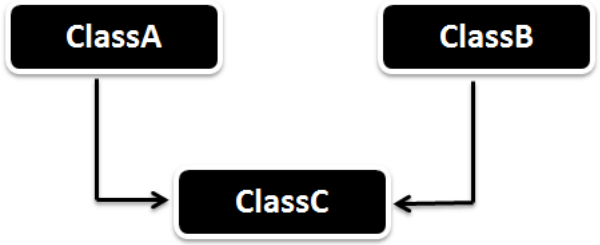
Single Inheritance is the simple inheritance of all, When a class extends another class(Only one class) then we call it as **Single inheritance.**

The below diagram represents the single inheritance in java where **Class B** extends only one class **Class A.** Here **Class B** will be the **Sub class** and **Class A** will be one and only **Super class**.



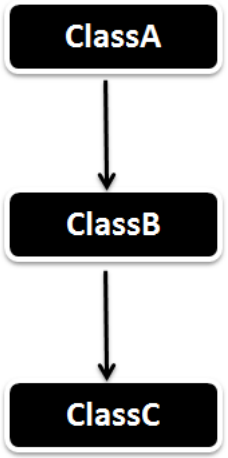
Inheritance – Multiple Inheritance

1. **Multiple Inheritance** is nothing but **a** class **extending more than** one class. Multiple Inheritance is not supported by many **Object Oriented Programming** languages such as **Java, Small Talk, C# etc.. (C++ Supports Multiple Inheritance).**
2. As the **Child** class has to manage the dependency of more than one **Parent** class. But you can achieve multiple inheritance in Java using **Interfaces**.



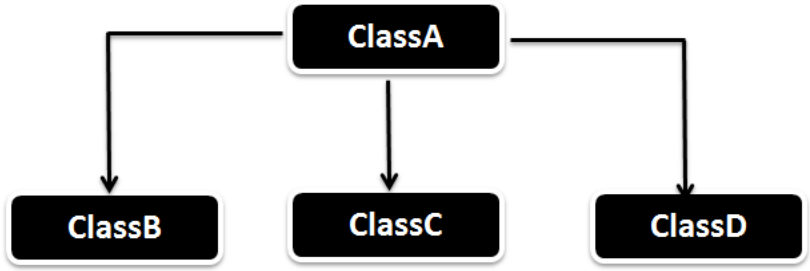
Inheritance – Multi-level Inheritance

1. In **Multilevel Inheritance** a derived class will be **inheriting a parent class** and as well as the derived class **act as the parent class** to other class.
2. As seen in the below diagram. **ClassB** inherits the property of **ClassA** and again **ClassB** act as a parent for **ClassC**.
3. In Short **ClassA** parent for **ClassB** and **ClassB** parent for **ClassC**.



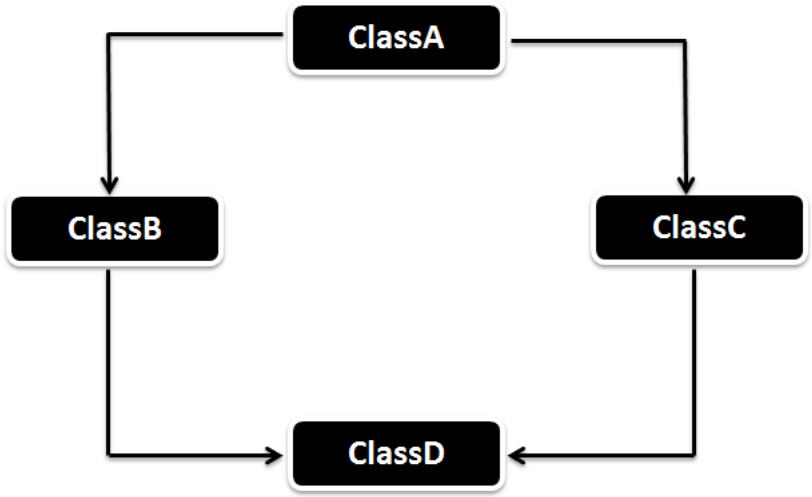
Inheritance – Hierarchical Inheritance

1. In **Hierarchical inheritance** one parent class will be inherited by **many** sub classes.
2. According to the example below, **ClassA** will be inherited by **ClassB, ClassC** and **ClassD**. **ClassA** will be acting as a parent class for **ClassB, ClassC** and **ClassD**.



Inheritance – Hybrid Inheritance

1. Hybrid Inheritance is the combination of both Single and Multiple Inheritance. Again Hybrid inheritance is also not directly supported in Java but only through interface(s).
2. Flow diagram of the Hybrid inheritance will look like below. As you can see **ClassA** will be acting as the **Parent** class for **ClassB & ClassC** and **ClassB & ClassC** will be acting as **Parent** for **ClassD.**



Programming Example 1 (cont.)

```
class Shape {
        public void DrawShape() {
                System.out.println("Draw of Shape Class"); }
            }
class Circle extends Shape {
        public void DrawCircle() {
                System.out.println("Draw of Circle Class");
        }
        }
```

Programming Example 1 (cont.)

```
Public static void Main(string[] Args)
        {
                Circle circle = new Circle();
                circle.DrawShape();
                //OUTPUT: Draw of Shape Class
                circle.DrawCircle();
                //OUTPUT: Draw of Circle Class
        }
```

(Class A)

(Class B)

Specialized Code/Unique to its own Class

Any questions?

Exercise Questions:

1. Write code of getter/setter for Person's class (included the inherited), Use your thinking and find appropriate class to write code.
2. Complete the class of Person and Shapes.
3. Give reasons why Problem-1 is correct or incorrect.
4. LAB ASSIGNMENT: Write code for each type of Inheritance.

Help/References/Guide

1. https://en.wikipedia.org/wiki/Inheritance
2. https://en.wikipedia.org/wiki/Inheritance_(object-oriented_programming)
3. https://www.guru99.com/java-class-inheritance.html
4. Deitel and Deitel, How to Program JAVA, 10E,
5. Courtesy of Mr. Saif Ullah Ejaz.
6. https://www.quora.com/What-are-advantages-of-inheritance-in-C++
7. https://javainterviewpoint.com/types-of-inheritance-in-java-singlemultiplemultilevelhierarchical-hybrid/

END

PRESENTATION MADE FOR: OBJECT ORIENTED PROGRAMMING.

DEPARTMENT OF COMPUTER SCIENCE, SZABIST, KARACHI.