Experiment No. 8
Implement a program on Single Inheritance.
Date of Performance:
Date of Submission:

Aim: To implement the concept of single inheritance.

Objective: Ability to design a base and child class relationship to increase reusability.

Theory:
Single inheritance can be defined as a derived class to inherit the basic methods (data members and variables) and behavior from a superclass. It's a basic is-a relationship concept exists here. Basically, java only uses a single inheritance as a subclass cannot extend more superclass.
Inheritance is the basic properties of object-oriented programming. Inheritance tends to make use of the properties of a class object into another object. Java uses inheritance for the purpose of code-reusability to reduce time by then enhancing reliability and to achieve run time polymorphism. As the codes are reused it makes less development cost and maintenance. Java has different types of inheritance namely single inheritance, multilevel, multiple, hybrid. In this article, we shall go through a basic understanding of single inheritance concept briefly in java with a programming example. Here we shall have a complete implementation in java.

Syntax:

The general syntax for this is given below. The inheritance concepts use the keyword 'extend' to inherit a specific class. Here you will learn how to make use of extending keyword to derive a class. An extend keyword is declared after the class name followed by another class name. Syntax is,

```
class base class
{…. methods
```

}

class derived class name extends base class

{

methods … along with this additional feature

}

Java uses a keyword 'extends' to make a new class that is derived from the existing class. The inherited class is termed as a base class or superclass, and the newly created class is called derived or subclass. The class which gives data members and methods is known as the base class and the class which takes the methods is known as child class.

Code:

Example 1:

```
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);
    }
}
```

Output:-

```
C:\Users\student\Desktop>java Programmer.java"
Programmer salary is: 40000.0
Bonus of Programmer is: 10000
```

Example 2:-

```
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:-

```
C:\Users\student\Desktop>java TestInheritance.java"
barking...
eating...
```

Conclusion:

Comment on the Single inheritance.

→
Single inheritance is a key concept in object-oriented programming that allows a derived class to inherit from a single base class. This approach promotes:

- Code Reusability: Subclasses can reuse methods and attributes from the parent class, reducing code duplication.
- Simplicity: A clear class hierarchy makes it easier to understand relationships between classes.
- Maintenance: Changes in the base class automatically update all derived classes, simplifying maintenance.

The examples provided show how the Programmer class inherits from Employee, gaining access to the salary attribute, while the Dog class extends Animal, inheriting the eat() method.
Overall, while single inheritance has its limitations, such as the inability to inherit from multiple classes, it remains essential for creating organized and maintainable software in Java. Understanding its principles is crucial for effective software design.