Experiment No. 7
Implement a program using super and final keyword.
Date of Performance:
Date of Submission:

Aim: To implement the concept of super and final keyword.

Objective: To understand the usage of superclass and final method , variables and class

Theory:

super and **final** keywords are two popular and useful keywords in Java. They also play a significant role in dealing with Java programs and their classes. In this chapter, you will learn about how to use super and final within a Java program.

Syntax: `super.<method-name>();`

- Super variables refer to the variable of a variable of the parent class.
- Super() invokes the constructor of immediate parent class.
- Super refers to the method of the parent class

Instance refers an instance variable of the current class by default, but when you have to refer parent class instance variable, you have to use super keyword to distinguish between parent class (here employee) instance variable and current class (here, clerk) instance variable.

What is final in Java?

Final is a keyword in Java that is used to restrict the user and can be used in many respects. Final can be used with:

- Class

- Methods
- Variables

A method declared as final cannot be overridden; this means even when a child class can call the final method of parent class without any issues, but the overriding will not be possible.

Once a variable is assigned with the keyword final, it always contains the same exact value. Again things may happen like this; if a final variable holds a reference to an object then the state of the object can be altered if programmers perform certain operations on those objects, but the variable will always refer to the same object. A final variable that is not initialized at the time of declaration is known as a blank final variable. If you are declaring a final variable in a constructor, then you must initialize the blank final variable within the constructor of the class. Otherwise, the program might show a compilation error.

Code:

```
public class Main {


    static class Animal1 {
        String color = "white";
    }

    static class Dog1 extends Animal1 {
        String color = "black";

        void printColor() {
            System.out.println(color);
            System.out.println(super.color);
        }
    }

        static class Animal2 {
        void eat() {
            System.out.println("eating...");
        }
    }

    static class Dog2 extends Animal2 {
        void eat() {
```

```
            System.out.println("eating bread...");
        }

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

        void work() {
            super.eat();
            bark();
        }
    }


    static class Animal3 {
        Animal3() {
            System.out.println("animal is created");
        }
    }

    static class Dog3 extends Animal3 {
        Dog3() {
            super();
            System.out.println("dog is created");
        }
    }


    public static void main(String[] args) {

        Dog1 d1 = new Dog1();
        d1.printColor();


        Dog2 d2 = new Dog2();
        d2.work();


        Dog3 d3 = new Dog3();


        final int x = 10;

        System.out.println("Value of final x: " + x);
    }
}
```

Output:

```
C:\Users\admin\Desktop>java Animal.java
black
white
eating...
barking...
animal is created
dog is created
Value of final x: 10
```

Conclusion:

Comment on how you have used super and final keyword.

→

- **super** Keyword: Allows access to parent class methods, variables, and constructors from a child class. It resolves ambiguities between parent and child class members and ensures proper initialization of inherited properties.
- **final** Keyword: Restricts modification of variables, methods, and classes. A final variable's value is immutable once set, a final method cannot be overridden, and a final class cannot be subclassed.

These keywords provide control and flexibility in Java's object-oriented programming, helping to create robust and maintainable code by clearly defining and enforcing behaviors and constraints.