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:

- **Testsuper1.java**

```
class Animal
{
	String color="white";
}
class Dog extends Animal
{
	String color="black";
	void printColor()
	{
		System.out.println(color);
		System.out.println(super.color);

	}
}
class TestSuper1
{
	public static void main(String args[])
	{
		Dog d=new Dog();
		d.printColor();
	}
}
```

- **Testsuper2.java**

```
class Animal
{
	void eat(){System.out.println("eating...");}
}
class Dog extends Animal
{
```

```
        void eat()
        {
                System.out.println("eating bread...");}
                void bark(){System.out.println("barking...");
        }
        void work()
        {
        super.eat();
        bark();
        }
}
class TestSuper2
{
        public static void main(String args[])
        {
                Dog d=new Dog();
                d.work();
        }
}
```

- **Testsuper3.java**

```
class Animal
{
        Animal()
        {
                System.out.println("animal is created");
        }
}
class Dog extends Animal
{
        Dog()
        {
                super();
                System.out.println("dog is created");
        }
}
class TestSuper3
{
        public static void main(String args[])
        {
                Dog d=new Dog();
        }
}
```

Conclusion:

The super and final keywords are integral to Java's object-oriented programming paradigm, each serving important roles in enhancing code structure and functionality:

- The super keyword facilitates interaction with parent class properties and methods, enabling effective inheritance and polymorphism. It allows child classes to leverage parent class functionality while maintaining clarity in their own implementations.
- The final keyword provides a mechanism for enforcing constraints, ensuring that certain values remain constant, methods remain unchangeable, and classes remain non-extendable. This contributes to code stability and predictability, which are crucial for maintaining large codebases.