## Exercise – 5

a) Write a JAVA program give example for “super” keyword.

programme

```
class Animal {
  public void animalSound() {
    System.out.println("The animal makes a sound");
  }
}

class Dog extends Animal {
  public void animalSound() {
    super.animalSound();
    System.out.println("The dog says: bow wow");
  }
}

public class Main {
   public static void main(String[] args) {
      Animal myDog = new Dog();
      myDog.animalSound();
   }
}
```

**out put**

```
The animal makes a sound
The dog says: bow wow
```

b) **Write a JAVA program to implement Interface. What kind of Inheritance can be achieved?**

**Programme**

**In Java, interfaces are a way to achieve abstraction and define contracts that implementing classes must adhere to. Interfaces can support multiple inheritance, meaning a class can implement multiple interfaces. However, a class can only extend one superclass (single inheritance).**

Here’s a simple example to demonstrate how to implement an interface in Java, along with the types of inheritance that can be achieved:

```
// Define an interface
interface Animal {
    void sound();  // abstract method
}

// Implementing the interface in a class
class Dog implements Animal {
    @Override
    public void sound() {
        System.out.println("Bark");
    }
}

// Another class implementing the same interface
class Cat implements Animal {
    @Override
    public void sound() {
        System.out.println("Meow");
    }
}

// Main class to test the interface implementation
public class Main {
    public static void main(String[] args) {
        Animal myDog = new Dog();
        Animal myCat = new Cat();

        myDog.sound();  // Output: Bark
        myCat.sound();  // Output: Meow
    }
}
```

out put

```
Bark
Meow
```

## Types of Inheritance Achieved with Interfaces

1. **Multiple Inheritance**: A class can implement multiple interfaces, allowing it to inherit behavior from multiple sources.
2. **Single Inheritance for Classes**: While a class can implement multiple interfaces, it can only extend one superclass. This is Java's way of avoiding the diamond problem associated with multiple inheritance in classes.

c.Write a JAVA program that implements Runtime polymorphism

```
class Animal {
    void sound() {
        System.out.println("Animal makes a sound");
    }
}
class Dog extends Animal {
    @Override
    void sound() {
        System.out.println("Dog barks");
    }
}
class Cat extends Animal {
    @Override
    void sound() {
        System.out.println("Cat meows");
    }
}
public class PolymorphismExample {
    public static void main(String[] args) {
        Animal myAnimal;
        myAnimal = new Dog();
        myAnimal.sound();
        myAnimal = new Cat();
        myAnimal.sound();
    }
}
```

out put:

```
Dog barks
Cat meows
```