Vidyavardhini’s College of Engineering & Technology

Department of Computer Science And Engineering (Data Science)

Experiment No. 8
Implement a program on multiple inheritance with interface.
Date of Performance:
Date of Submission:

Aim: Implement a program on multiple inheritance with interface.

Objective:

Implement multiple inheritance in a program to perform addition, multiplication and transpose operations on a matrix. Create an interface to hold prototypes of these methods and create a class input to read input. Inherit a new class from this interface and class. In main class create object of this child class and invoke required methods.

Theory:

- In Multiple inheritance, one class can have more than one superclass and inherit features from all parent classes. Java does not support multiple inheritance with classes. In java, we can achieve multiple inheritance only through Interfaces.
- An interface contains variables and methods like a class but the methods in an interface are abstract by default unlike a class. If a class implements multiple interfaces, or an interface extends multiple interfaces, it is known as multiple inheritance.
- However, Java supports multiple interface inheritance where an interface extends more than one super interfaces.
- A class implements an interface, but one interface extends another interface. Multiple inheritance by interface occurs if a class implements multiple interfaces or also if an interface itself extends multiple interfaces.
- The following is the syntax used to extend multiple interfaces in Java:

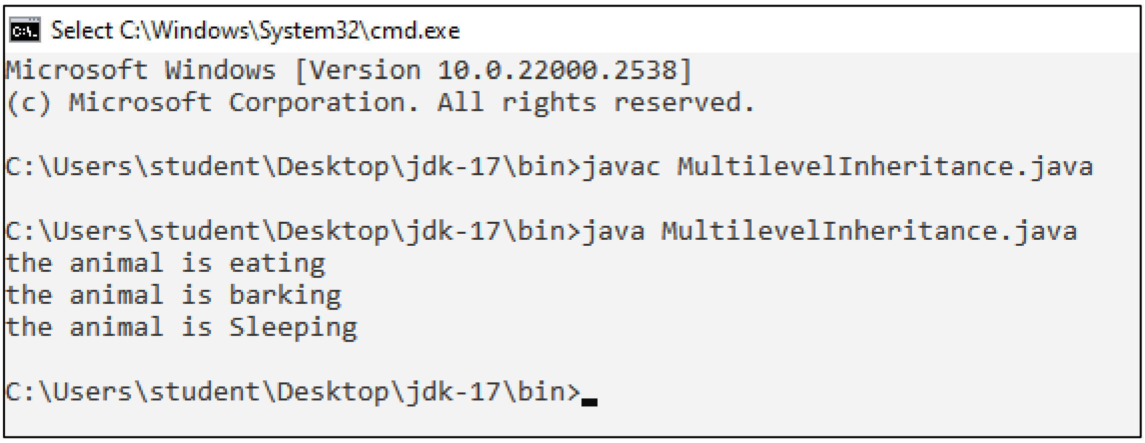
```
 access_specifier interface subinterfaceName extends superinterface1, superinterface2, …… {
// Body
}
```

Code:

```
    class MultilevelInheritance{
        public static void main(String args[])
    {
      PuppyDog puppy = new PuppyDog();
```

```
  puppy.eat();
  puppy.bark();
  puppy.sleep();
}
}
class Animal
    {
      void eat()
        {
            System.out.println("the animal is eating");
        }
    }
class Dog extends Animal
    {
     void bark()
       {
          System.out.println("the animal is barking");
       }
    }
class PuppyDog extends Dog
    {
     void sleep()
       {
          System.out.println("the animal is Sleeping");
       }
    }
```

Output:



```
Microsoft Windows [Version 10.0.22000.2538]
(c) Microsoft Corporation. All rights reserved.

C:\Users\student\Desktop\jdk-17\bin>javac MultilevelInheritance.java

C:\Users\student\Desktop\jdk-17\bin>java MultilevelInheritance.java
the animal is eating
the animal is barking
the animal is Sleeping

C:\Users\student\Desktop\jdk-17\bin>
```

Conclusion:

Interfaces are useful in Java for the following reasons:

Interfaces encourage you to design your code in an abstract way, focusing on what should be done rather than how it should be done. This makes your code more flexible and reusable.

Interfaces allow you to write polymorphic code, meaning that you can write code that can work with different types of objects. This makes your code more powerful and adaptable.

Interfaces provide a contract between classes that implement them. This contract ensures that all classes that implement the interface have the same methods and that those methods behave in a consistent way.

To implement an interface in Java, you use the implements keyword. The implements keyword tells the compiler that the class is implementing the specified interface.