Green University of Bangladesh

Department of Computer Science and Engineering (CSE)

Faculty of Sciences and Engineering

Semester: (Spring, Year:2023), B.Sc. in CSE (Day)

Lab Report NO 01

Course Title	**: Object Oriented Programming Lab**
Course Code	**: CSE 202**
Section	**: D1**

Lab Report on the Topi: Abstract Class and Interface and How java can implement Multiple Inheritances.

Student Details

	Name	**ID**
1.	Jahidul Islam	221002504

Lab Date	**: 16/03/2023**
Submission Date	**: 23/05/2023**
Course Teacher's Name	**: Ayesha Khatun**

Lab Report Status

Marks: ………………………………… **Signature:.....................**

Comments:.. **Date:..............................**

Title:

Abstract Classes and Interfaces in Java: A Comparative Study

Abstract:

This lab report aims to provide a comprehensive understanding of abstract classes and interfaces in Java, including their definitions, purpose, implementation, and practical usage. Through theoretical explanations and code examples, we will explore the similarities, differences, and practical scenarios in which each construct is best suited. This report will contribute to a deeper comprehension of object-oriented programming concepts in Java.

1. Abstract Classes:

1.1 Definition:

An abstract class in Java is a class that cannot be directly instantiated and serves as a blueprint for subclasses. It provides a way to define common attributes and behaviors that subclasses can inherit and extend. Abstract classes can contain both abstract methods, which lack implementations, and concrete methods with complete implementations.

1.2 Characteristics of Abstract Class:

- An abstract class in Java cannot be directly instantiated; it serves as a blueprint for subclasses to inherit from.
- Abstract classes allow you to define common attributes and behaviors that subclasses should implement.
- Abstract classes can have both abstract methods (without implementations) and concrete methods (with complete implementations).
- Subclasses that inherit from an abstract class must implement all the abstract methods, while they can directly use the concrete methods provided by the abstract class.

1.3 Code Example: Abstract class

```
public abstract class Animal {
    protected String name;

    public Animal(String name) {
        this.name = name;
    }

    public abstract void sound();

    public void sleep() {
        System.out.println(name + " is sleeping.");
    }
}

public class Cat extends Animal {
    public Cat(String name) {
        super(name);
    }

    public void sound() {
        System.out.println(name + " meows.");
    }
}

public class Main {
    public static void main(String[] args) {
        Animal animal = new Cat("Tom");
        animal.sound();  // Output: Tom meows.
        animal.sleep();  // Output: Tom is sleeping.
    }
}
```

2. Interfaces:

2.1 Definition:

An interface in Java is a collection of abstract methods that define a contract for classes to implement. It provides a way to achieve multiple inheritances as a class can implement multiple interfaces. Interfaces are used to specify a set of methods that a class must implement without prescribing any implementation details.

2.2 Characteristics of Interface:

- An interface in Java defines a contract or a set of rules that classes must follow.
- Interfaces only contain abstract method declarations, without any implementations.
- Classes can implement multiple interfaces, allowing for flexibility and achieving multiple inheritances.
- Implementing an interface ensures that a class provides specific behaviors as defined by the interface, promoting consistency and code reuse.

2.3 Code Example: Interface

```
public interface Vehicle {
   void start();
   void stop();
}
public class Car implements Vehicle {
   public void start() {
      System.out.println("Car has started.");
   }
   public void stop() {
      System.out.println("Car has stopped.");
   }
}
public class Bike implements Vehicle {
   public void start() {
      System.out.println("Bike has started.");
   }
   public void stop() {
      System.out.println("Bike has stopped.");
   }
}
public class Main {
   public static void main(String[] args) {
      Vehicle car = new Car();
      car.start();  // Output: Car has started.
      car.stop();   // Output: Car has stopped.

      Vehicle bike = new Bike();
      bike.start(); // Output: Bike has started.
      bike.stop();  // Output: Bike has stopped.
   }
}
```

3. Comparative Analysis:

Abstract classes and interfaces have distinct characteristics and are used in different scenarios:

- Abstract classes are used when defining common attributes and behaviors for related classes in an inheritance hierarchy, whereas interfaces define a contract for unrelated classes to adhere to.
- Abstract classes can contain both abstract and concrete methods, whereas interfaces can only have abstract method declarations.
- A class can extend only one abstract class, but it can implement multiple interfaces.
- Abstract classes provide a default implementation for some methods, while interfaces do not provide any implementation.
- Interfaces are useful in achieving loose coupling, as classes can implement multiple interfaces and provide different behaviors for each.

4. Why Interface “interface” keyword is such important in Java?

- **Abstraction and Polymorphism:** Interfaces provide a way to achieve abstraction by defining a contract or set of methods that classes must implement. This allows for polymorphism, where different classes can be treated interchangeably based on their shared interface. This promotes code flexibility, extensibility, and modular design.

- **Multiple Inheritance:** Unlike classes, which can only extend a single superclass, Java allows a class to implement multiple interfaces. This enables a class to inherit behaviors from multiple sources, promoting code reuse and providing a solution to the limitation of single inheritance.

- **Loose Coupling:** Interfaces help in achieving loose coupling between components. By programming to interfaces rather than concrete classes, different implementations can be easily swapped out without affecting the code that uses the interface. This enhances code maintainability, modularity, and the ability to adapt to changing requirements.

- **API Design and Contracts:** Interfaces play a crucial role in defining APIs (Application Programming Interfaces) and specifying contracts. They define the methods and behaviors that classes implementing the interface must adhere to, establishing a clear and consistent programming interface. This aids in collaboration between developers, allowing for the creation of libraries, frameworks, and systems with well-defined interfaces.

5. Conclusion:

In conclusion, abstract classes and interfaces are essential constructs in Java's object-oriented programming paradigm. Abstract classes allow for defining common attributes and behaviors within an inheritance hierarchy, while interfaces provide a way to specify contracts for unrelated classes. Understanding the differences and proper use cases for each construct contributes to writing flexible and maintainable Java code.

Green University of Bangladesh
Department of Computer Science and Engineering (CSE)
Faculty of Sciences and Engineering
Semester: (Spring, Year:2023), B.Sc. in CSE (Day)

Lab Report NO:

Course Title :

Course Code :
Section :

Lab Report on the Topic:

Student Details

Name		ID
1.		

Lab Date :
Submission Date :
Course Teacher's Name :

Lab Report Status
Marks: ………………………………… **Signature:.....................**
Comments:.. **Date:..............................**