



Green University of Bangladesh

Department of Computer Science and Engineering (CSE)

**Faculty of Sciences and Engineering, Semester: Spring, Year:
2024, B.Sc. in CSE (weekend)**

Lab Report # 05

Course Title: Object Oriented Programming Lab

Course Code: CSE-202

Section: 223_E1

Student Details

Name		ID
1.	Waliullah	223015026

Date : 17-05-2024

Submission Date : 25-05-2024

Course Teacher's Name : Abdullah Al Farhad

Assignment Status

Marks:

Signature:.....

Comments:.....

Date:.....

Title: Implementation of Interface and Multiple Inheritance.

1. Introduction:

- In object-oriented programming, interfaces and inheritance are fundamental concepts for structuring and organizing code. Interfaces define contracts that classes must adhere to, while inheritance allows classes to reuse functionality from parent classes. This approach promotes code reusability, abstraction, and polymorphism.

2. Objective:

- Create an interface **isEmergency** with a method **soundSiren**.
- Implement the **isEmergency** interface in a class **fireEmergency**.
- Create a class **smokeAlarm** without implementing any interface.
- Populate an array with instances of these classes.
- Identify and invoke methods on objects that implement the **isEmergency** interface.

3. Use-Case:

In real-world applications, this approach can be used in emergency response systems where different types of emergency responses (fire, medical, security) need to implement a common interface to ensure they can be handled uniformly. This design ensures that any new type of emergency can be integrated into the system easily by implementing the **isEmergency** interface.

- Fire alarm systems with different types of alarms (sirens, flashing lights).
- Security systems with diverse alert mechanisms (sirens, phone notifications).
- Disaster response systems with coordinated actions based on the emergency type.

4. Procedure:

- **Define an Interface:** Create an interface **isEmergency** with a method **soundSiren**.
- **Implement the Interface:** Write a class **fireEmergency** that implements the **isEmergency** interface and provides an implementation for the **soundSiren** method.
- **Create Another Class:** Write a class **smokeAlarm** with an empty body that does not implement any interface.
- **Create an Array of Objects:** In the main method, create an array **myArray** of the **Object** class.
- **Instantiate and Add Objects to Array:** Create instances of **smokeAlarm** and **fireEmergency**, and add them to **myArray**.
- **Identify and Interact with Interface Instances:** Use a loop to identify elements in **myArray** that implement **isEmergency** and call their **soundSiren** method.

5. Implementations and Output:

- Code:

```
// create a interface for isEmergency
interface isEmergency{
    void soundSiren();
}

// create a class fireEmergency and implements isEmergency interface
class fireEmergency implements isEmergency{
    public void soundSiren(){
        System.out.println("Siren sound!");
    }
}

// create a smokeAlarm class with no implementation and empty body
class smokeAlarm{
}

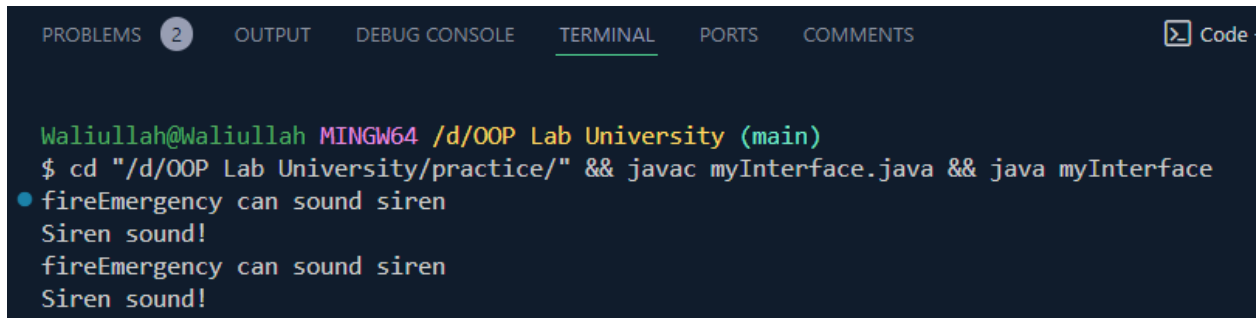
public class myInterface {
    public static void main(String[] args) {
        Object[] myArray = new Object[4];

        // Construct 2 SmokeAlarm objects and add to the array
        for (int i = 0; i < 2; i++) {
            myArray[i] = new smokeAlarm();
        }

        // Construct 2 FireEmergency objects and add to the array
        for (int i = 2; i < 4; i++) {
            myArray[i] = new fireEmergency();
        }

        // Loop through the array to identify instances of IsEmergency and call soundSiren
method
        for (Object element : myArray) {
            if (element instanceof isEmergency) {
                System.out.println(element.getClass().getName() + " can sound siren");
                ((isEmergency) element).soundSiren();
            }
        }
    }
}
```

- **Output:**



```
PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS Code

Waliullah@Waliullah MINGW64 /d/OOP Lab University (main)
$ cd "/d/OOP Lab University/practice/" && javac myInterface.java && java myInterface
• fireEmergency can sound siren
  Siren sound!
  fireEmergency can sound siren
  Siren sound!
```

6. Limitations:

This exercise is limited to a simple demonstration of interfaces and inheritance. It does not cover more advanced topics such as interface inheritance, multiple interfaces, or the complexities involved in real-world use cases like handling null references or dealing with multiple levels of inheritance.

- The code uses an `Object` array for simplicity, but typically you'd use an array of the interface type (`isEmergency[]`) for stronger type safety.
- The `smokeAlarm` class doesn't implement `isEmergency`, so it cannot sound the siren. Consider adding appropriate functionality if needed.

7. Conclusion:

Interfaces and inheritance are powerful tools for creating well-structured, reusable, and maintainable object-oriented code. This example showcases how to define an interface, implement it in classes, and leverage it to achieve a specific behavior (sounding an emergency siren) while maintaining flexibility for different emergency types.