Green University of Bangladesh
Department of Computer Science and Engineering (CSE)
Faculty of Sciences and Engineering
Semester:(Spring, Year:2024), B.Sc. in CSE (Day)

Lab Report NO: 03
Course Title: Object-Oriented Programming Lab
Course Code: CSE 202 Section: 223 D9

Lab Experiment Name: Inheritance.

Student Details

	Name	ID
1.	Promod Chandra Das	231002005

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

Lab Report Status

Marks: …… **Signature:.....**
Comments:..... **Date:.....**

TITLE OF THE LAB REPORT EXPERIMENT:

- Inheritance

OBJECTIVES/AIM :

• Understanding to use of Inheritance using Java.
• Role of constructors.
• Implementing certain types of inheritances.
• Access modifiers.

PROCEDURE / ANALYSIS / DESIGN :

Problem analysis

In object-oriented programming, inheritance is the mechanism by which we attain reusability of certain com-
ponents through the means of using classes. The process fundamentally involves building a structure by making

a class and using this class for making other classes while retaining certain portions of the superclass(i.e the
one we modeled our new class upon). This concept varies from one programming language to another. Here,
we will only focus on java programming language-based implementation of the concept of inheritance. For this
at first, we need to understand some terms such as:

• superclass
• constructors
• subclass
• multilevel inheritance
• multiple inheritance

- **Superclass**

Simply put super class is a kind of class that other class inherits from. Generally most commonly used methods
and attributes are part of it. There ara also special kinds of superclasses where we only provide guidance for
subclasses (classes that inherits this superclass).

```
1
2 /*java class declaration:*/
3
4 class MyClass {
5 // field, constructor, and
6 // method declarations
7 }
```

- **Subclass**

A class that inherits properties from another class are called as subclass. It is often the case that a subclass
inherits another subclass. If class B inherits from class A, then A is the superclass and B is the subclass.

```
1 /*java subclass declaration:*/
2 class MyClass extends MySuperClass {
3 // field, constructor, and
4 // method declarations
5 }
```

- **Constructor**

Constructors initializes an object when it is created. A constructors is a methods that has the same name as
the class itself. All the supeclass and subclass can have constructor.

```
1 /*java declaration:*/
2 class MyClass {
3 // constructor
4 Myclass(parameters){
5 //
6 }
7 }
```

- **Multilevel Inheritance**

Let's consider 3 classes, A, B, C, then if class B inherits A then it is called single level inheritance. If further C
inherits B, then we have 2 levels. this is called multilevel inheritance. We can have more than 2 levels. When
we have more than 2 level of inheritance, we call it multilevel inheritance in java.

```
1 /*java declaration:*/
2 class A {
3 // constructor
4 A(parameters){
5 //
6 }
7 }
8 class B extends A{
9 //1st level
10 }
11 class C extends B{
12 //2nd level
13 }
```

- **Multiple Inheritance**

Again assume 3 classes: A, B, C. If class B inherits from both A and C we call it multiple inheritance.

```
1 /*java declaration:*/
2 class A {
3 // constructor
4 A(parameters){
5 //
6 }
7 }
8 class C{
9 C(parameters){
10 //
11 }
12 }
13 class B extends A,C{
14 //
15 }
```

- **Access Classifiers**

Summarization:

Location Private No Modifier Protected Public
same class yes yes yes no
same package subclass no yes yes yes
same package non-subclass no yes yes yes
different package subclass no no yes yes
different package non-subclass no no no yes

Table 1: Access Modifiers in Java

Implementation

• Create a java project name Lab(this also creates Lab.java file)
• Create a class under the package lab named Class1 (i.e create Class1.java file)
• Create another class under the package lab named Class2 (i.e create Class2.java file)

Lab.java file code:

```
1 /*Lab.java file*/
2 package Lab;
3 public class Lab {
4 public static void main(String[] args) {
5 Class1 cs1=new Class1("String passing.");
6 //cs1.show();
7 //cs1.show_string();
8 Class2 cs2=new Class2("Class2 string passing.",3);
9 cs2.show();
10 cs2.show_string();
11 cs2.show_count();
12 //System.out.println(cs2.name);
13 }
14 }
```

Create Class1.java file:

```
1 package lab;// this Class1 should be in the same package
2
3 public class Class1 {
4 String s="";
5 String name="Class1 name";
6
7 Class1(String s){
8 this.s=s;
9 }
10 public void show(){
11 System.out.println("Class1");
```

```
12 }
13 public void show_string(){
14 System.out.println(this.name);
15 }
16 }
```

Create Class2.java file:

```
1 package lab;
2 public class Class2 extends Class1{
3 int count=0;
4
5 Class2(String s, int count){
6 super(s);
7 this.count=c;
8 }
9 @Override
10 public void show(){
11 System.out.println("Class2.");
12 System.out.println(super.name);
13 super.show_string();
14 }
15 public void show_count(){
16 System.out.println("Class2 "+this.count);
17 }
18 }
```

5 Input/Output

```
Output of the program is given below.
Class2.
Class1 name
Class1 name
Class1 name
Class2 3
```

4. IMPLEMENTATION

• Create a java project name Lab(this also creates Lab.java file)

• Create a class under the package lab named Class1 (i.e create Class1.java file)
• Create another class under the package lab named Class2 (i.e create Class2.java file)

Lab.java file code:

```
1 /*Lab.java file*/
2 package Lab;
3 public class Lab {
4 public static void main(String[] args) {
5 Class1 cs1=new Class1("String passing.");
6 //cs1.show();
7 //cs1.show_string();
8 Class2 cs2=new Class2("Class2 string passing.",3);
9 cs2.show();
10 cs2.show_string();
11 cs2.show_count();
12 //System.out.println(cs2.name);
13 }
14 }
```

Create Class1.java file:

```
1 package lab;// this Class1 should be in the same package
2
3 public class Class1 {
4 String s="";
5 String name="Class1 name";
6
7 Class1(String s){
8 this.s=s;
9 }
10 public void show(){
11 System.out.println("Class1");
12 }
13 public void show_string(){
14 System.out.println(this.name);
15 }
16 }
```

Create Class2.java file:

```
1 package lab;
2 public class Class2 extends Class1{
3 int count=0;
4
5 Class2(String s, int count){
6 super(s);
7 this.count=c;
8 }
9 @Override
10 public void show(){
11 System.out.println("Class2.");
12 System.out.println(super.name);
13 super.show_string();
14 }
15 public void show_count(){
16 System.out.println("Class2 "+this.count);
17 }
18 }
```

5. TEST RESULT / OUTPUT :

```
Output of the program is given below.
Class2.
Class1 name
Class1 name
Class1 name
Class2 3
```

- **Lab Exercise (Submit as a report**

❖ Specific problem/problems for lab report may be provided by course teacher. Some suggested problems:

• Implement Multiple inheritance. 3 Classes A,B,C. Class C inherits both A and B.

• Try various combinations of public, private, protected in the given code and verify if it satisfies the table in table 1.

Code:-

```
// Problem 1: Implement Multiple Inheritance
// Define class A
class A {
    void displayA() {
        System.out.println("Inside Class A");
    }
}

// Define class B
class B {
    void displayB() {
        System.out.println("Inside Class B");
    }
}

// Define class C that inherits from both A and B
class C extends A {
    void displayC() {
        System.out.println("Inside Class C");
    }
}

// Problem 2: Try various combinations of public, private, protected in the given code and verify if it satisfies the table in Table 1.
class D {
    private int privateVar = 10;
    public int publicVar = 20;
    protected int protectedVar = 30;

    void display() {
        System.out.println("Private variable: " + privateVar);
        System.out.println("Public variable: " + publicVar);
        System.out.println("Protected variable: " + protectedVar);
    }
}

public class Main {
    public static void main(String[] args) {
```

```
        // Problem 1: Implement Multiple Inheritance
        C objC = new C();
        objC.displayA(); // Accessing method from class A
        objC.displayC(); // Accessing method from class C

        // Problem 2: Try various combinations of public, private, protected in the given code
        // and verify if it satisfies the table in Table 1.
        D objD = new D();
        // Trying to access privateVar will result in a compile-time error
        // System.out.println(objD.privateVar);

        // Accessing publicVar and protectedVar works fine
        System.out.println("Public variable: " + objD.publicVar);
        System.out.println("Protected variable: " + objD.protectedVar);

        // Trying to access privateVar through a method works as it's inside the class
        objD.display();
    }
}
```

- ANALYSIS AND DISCUSSION:

Inheritance is a fundamental object-oriented programming (OOP) concept that allows you to create new classes (subclasses) that inherit properties (fields) and behaviors (methods) from existing classes (superclasses). This promotes code reuse, reduces redundancy, and fosters a hierarchical organization within your program.