LAB ACTIVITY 7:

Inheritance and Polymorphism

**Learning Outcomes**

This Lab sheet encompasses 7 activities (Activity 7A, 7B, 7C, 7D, 7E, 7F and 7G).

By the end of this lab, students should be able to:

1. Construct Inheritance:

* Build the protected access specifier.
* Build constructors in inheritance.
* Build keyword super in Java programs

1. Construct Polymorphism

* Construct overriding method.

Activity 7A

Activity Outcome: Define Super and usage of super keyword in Java Code

Briefly explain in your own words the definitions below.

Super is basically can be used to refer data from the main parent class variable

Super

We use super when we have one class that inherit other class as their reference

Usage of Super keyword

Activity 7B

Activity Outcome: Build constructors in inheritance and build super keyword in java code

Rewrite the program bellow, used **super** keyword to distinguish between parent class instance variable and current class instance variable.

1. **class** Vehicle
2. {
3. **int** speed=50;
4. }
6. **class** Bike3 **extends** Vehicle
7. {
8. **int** speed=100;
9. **void** display()
10. {
11. System.out.println(speed);
12. }
14. **public** **static** **void** main(String args[])
15. {
16. Bike3 b=**new** Bike3();
17. b.display();
18. }
19. }

**Code without super**

Text

Description automatically generated

**Output**



**Code with super added**

Text

Description automatically generated

Output



Rewrite the program bellow, use **super** keyword to invoke the parent class constructor

1. **class** Vehicle
2. {
3. int speed;
4. Vehicle(int speed)
5. {
6. this.speed =speed;
7. System.out.println("Vehicle is created"+ this.speed);
8. }
9. }
11. **class** Bike5 **extends** Vehicle
12. {
13. Bike5()
14. {
15. super(50);
16. System.out.println("Bike is created");
17. }
19. **public** **static** **void** main(String args[])
20. {
21. Vehicle v= new Vehicle(100);
22. Bike5 b=**new** Bike5();
24. }
25. }

Code

A screenshot of a computer

Description automatically generated with medium confidence

Output



Activity 7C

Activity Outcome: Syntax overriding method

Briefly explain in your own words the definitions below.

*Overriding is basically when a subclass is using the same method of the parent class*

Method Overriding

Activity 7D

Activity Outcome: Construct Method Overriding in java

**Procedure:**

**Step 1**: Type the following code

**Step 2:** Save the program

**Step 3:** Compile and execute the program

**Step 4:** Write the output

|  |
| --- |
| class Sum  {  int n;    void add (int num) // add method in super class  {  int resul t = 0;  this.n = num;  for (int i=1; i<=n; i++)  result = result + i;  System.out.println("The sum is " +result);  }  }  class Even extends Sum  {  void add (int num) // add method in subclass  {  int result = 0;  this.n = num;  for (int i=2; i<=n; i+=2)  result = result + i;  System.out.println("The sum is " +result);  }  }  class Act7D  {  public static void main (String args[])  {  Sum obj1 = new Sum();  Even obj2 = new Even();  obj1.add(10); // calling the Add method of super class  obj2.add(10); // calling the Add method of sub class  }  } |

Activity 7E

Activity Outcome: Define Polymorphism

Briefly explain in your own words the definitions below.

Polymorphism



Activity 7F

Activity Outcome: Implement polymorphism in Java Code.

Write a program to create a class named shape. In this class we have two sub classes rectangle, and triangle. All the shapes must have a method called getArea(), which returns the area of that particular shape. Write the code based the Class Diagram and Shape.java below: -

|  |
| --- |
| OOP_PolymorphismShape.png  Output:  Rectangle of length=4 and width=5, subclass of Shape of color="red"  Area is 20.0  Triangle of base=4 and height=5, subclass of Shape of color="blue"  Area is 10.0 |

|  |
| --- |
| //Shape.java  **public** **class** Shape  {  **private** String color;    // Constructor  **public** Shape (String color)  {  **this**.color = color;  }    **public** String toString()  {  **return** "Shape of color=\"" + color + "\"";  }    **public** **double** getArea()  {  System.*err*.println("Shape unknown! Cannot compute area!");  **return** 0;  }  } |