|  |
| --- |
| **Computer Engineering Department - ITU** |
| **CE101L: Object Oriented Programming Lab** |

|  |  |
| --- | --- |
| **Course Instructor: Usama Bin Shakeel** | **Dated: 27/04/2022** |
| **Teaching Assistant: Aqsa Khalid** | **Semester: Spring 2022** |
| **Lab Engineer: Nadir Abbas** | **Batch: BSCE2021** |

# **Lab 8A. Use of Polymorphism & Virtual Functions in Classes and Objects**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Roll number** | **Report**  **(out of 100)** | **Scaled to 10** | **Total**  **(out of 10)** |
| NIMRA MAQBOOL | BSCE21012 |  |  |  |

Checked on: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## **Objective**

The objective of this lab is to observe the basic knowledge of programming classes in C++.

## **Equipment and Component**

|  |  |  |
| --- | --- | --- |
| **Component Description** | **Value** | **Quantity** |
| Computer | Available in lab | 1 |

## **Conduct of Lab**

1. Students are required to perform this experiment individually.
2. In case the lab experiment is not understood, the students are advised to seek help from the course instructor, lab engineers, assigned teaching assistants (TA) and lab attendants.

## **Theory and Background**

**Polymorphism means "many forms",** and it occurs when we have many classes that are related to each other by inheritance.Like we specified in the previous chapter; Inheritance lets us inherit attributes and methods from another class. Polymorphism uses those methods to perform different tasks. This allows us to perform a single action in different ways.

We can implement polymorphism in C++ using the following ways:

1. Function overloading (Compile time polymorphism)
2. Operator overloading (Compile time polymorphism)
3. Function overriding (Run time polymorphism)
4. Virtual functions (Run time polymorphism)

A **virtual function** is a member function in the base class that we expect to redefine in derived classes. Basically, a virtual function is used in the base class to ensure that the function is overridden. This especially applies to cases where a pointer of base class points to an object of a derived class.

**Lab Task**

**Task A: Polymorphism [Marks: 20]**

In this task, you are required to create one base class **Shape** and three child classes **Circle**, **Square**, and **Triangle** with the following data members and member functions,

***Protected Data Members of class Shape such as:***

c\_radius, s\_area, s\_length, s\_width, t\_base, t\_height (int)

c\_area, t\_area (float)

pie = 3.1416 (const int)

***Public Member Functions of class Shape such as:***

**void calculateArea() –** It will print “Parent Class is generic”.

**void Display() –** It will print “Your Name and Roll Number.

***Inherited Data Members of class Circle such as:***

It will inherit c\_radius and c\_area from shape class.

***Public Member Functions of class Circle such as:***

**void calculateArea() –** It will take input c\_radius and calculate area and display area of the circle.

**void Display() –** It will take input c\_radius and calculate, store in c\_area and display area of the circle.

***Inherited Data Members of class Square such as:***

It will inherit s\_length, s\_width and s\_area from shape class.

***Public Member Functions of class Square such as:***

**int calculateArea(int a) –** It will initialize s\_length and s\_width with a and calculate and return area of the square.

**void Display() –** It will take input s\_length and s\_width and calculate area, store in s\_area and display area of the square.

***Inherited Data Members of class Triangle such as:***

It will inherit t\_base, t\_height and t\_area from shape class.

***Public Member Functions of class Triangle such as:***

**float calculateArea(int b, int h) –** It will initialize t\_base and t\_height with b and h and calculate and return area of the triangle.

**void Display() –** It will take input t\_base and t\_height and calculate area, store in t\_area and display area of the triangle.

Create a **UML diagram.**

Do the following operations in main function:

1. Create objects of class **Shape, Circle, Square,** and **Triangle** and call member function **calculateArea()**. Write below its function overloading or function overriding.

Ans:

2. Create objects of class **Shape, Circle, Square,** and **Triangle** and call member function **Display()**. Write below its function overloading or function overriding.

Ans:

|  |
| --- |
| **Function.h:**  class shape { //created class of shape protected:  float s\_area;  double c\_radius;  float s\_length;  float s\_width; //declared protected members  float t\_base;  float t\_height;  float c\_height;  float t\_area;  const double pie = 3.1416; public:  void calculateArea(); //declared public members   void display();  };  class circle : protected shape { //inherited shape class public:  void calculateArea(double r); //declared public members   void display(double r);  };  class square : protected shape { //inherited shape class public:  void calculateArea(float a); //declared public members   void display(); };  class triangle : protected shape { //inherited shape class public:  float calculateArea(float b, float h); //declared public members   void display(); };  class shape1 { //created class of shape1 protected:  float s\_area;  double c\_radius;  float s\_length;  float s\_width; //declared protected members  float t\_base;  float t\_height;  float c\_height;  float t\_area;  const double pie = 3.1416; public:  void calculateArea(); //declared public members };  class circle1 : public shape1 { //inherited shape1 class public:  void calculateArea(); //declared public members };  class square1 : public shape1 { //inherited shape1 class public:  void calculateArea(); //declared public members };  class triangle1 : public shape1 { //inherited shape1 class public:  float calculateArea(); //declared public members };  **function.cpp:**  void shape::calculateArea() { //function of shape class  cout << "PARENT CLASS IS GENERIC" << endl; //displaying }  void shape::display() { //function of shape class  cout << "NAME : NIMRA MAQOOL" << endl;  cout << "ROLL NUMBER : BSCE21012" << endl; //displaying }  void circle::calculateArea(double r) { //function of circle class  cout << "ENTER RADIUS = ";  cin >> r; //taking input  c\_radius = r;  double area; //declaring  area = (pie \* ((c\_radius) \* (c\_radius))); //calculating  s\_area = area;  cout << "AREA = " << s\_area << endl; //displaying }  void circle::display(double r) { //function of circle class  cout << "ENTER RADIUS = ";  cin >> r; //taking input  c\_radius = r;  double area; //initializing  area = (pie \* ((c\_radius) \* (c\_radius))); //calculating  s\_area = area;  cout << "AREA OF CIRCLE = " << s\_area << endl; //displaying }  void square::calculateArea(float a) { //function of square class  cout << "ENTER A = ";  cin >> a; //taking input  s\_length = a;  s\_width = a;  float area; //declaring  area = s\_width \* s\_length; //calculating  s\_area = area;  cout << "AREA OF SQUARE = " << s\_area << endl; //displaying }  void square::display() { //function of square class  float a;  cout << "ENTER LENGTH (as length is equal to width) = ";  cin >> a; //taking input  s\_length = a;  s\_width = a;  float area; //initializing  area = s\_width \* s\_length; //calculating  s\_area = area;  cout << "AREA OF SQUARE = " << s\_area << endl; //displaying }  float triangle::calculateArea(float b, float h) { //function of triangle class  cout << "ENTER HEIGHT = ";  cin >> h;  t\_height = h;  cout << "ENTER BASE = ";  cin >> b; //taking input  t\_base = b;  float area;  float area1; //declaring  area = (t\_height \* t\_base); //calculating  area1 = (area) \* 1 / 2;  t\_area = area1;  cout << "AREA OF TRIANGLE = " << t\_area << endl; //displaying  return t\_area; }  void triangle::display() { //function of triangle class  float b;  float h;  cout << "DISPLAY IS CALLED.." << endl;  cout << "ENTER HEIGHT = ";  cin >> h; //taking input  t\_height = h;  cout << "ENTER BASE = ";  cin >> b; //taking input  t\_base = b;  float area;  float area1; //declaring  area = (t\_height \* t\_base);  area1 = (area) \* 1 / 2; //calculating  t\_area = area1;  cout << "AREA OF TRIANGLE = " << t\_area << endl; //displaying }  void shape1::calculateArea() { //function of shape1 class  cout << "PARENT CLASS IS GENERIC" << endl;  cout << "NAME : NIMRA MAQBOOL" << endl; //displaying  cout << "ROLL NUMBER : BSCE21012 " << endl; }  void circle1::calculateArea() { //function of circle1 class  double x;  cout << "ENTER RADIUS = ";  cin >> x; //taking input  c\_radius = x;  double Area; //declaring  Area = (pie \* ((c\_radius) \* (c\_radius))); //calculating  s\_area = Area;  cout << "AREA = " << s\_area << endl; //displaying }  void square1::calculateArea() { //function of square1 class  float A;  cout << "ENTER A = ";  cin >> A; //taking input  s\_length = A;  s\_width = A;  float area; //declaring  area = s\_width \* s\_length; //calculating  s\_area = area;  cout << "AREA OF SQUARE = " << s\_area << endl; //displaying }  float triangle1::calculateArea() { //function of triangle1 class  float b;  float h;  cout << "ENTER HEIGHT = ";  cin >> h; //taking input  t\_height = h;  cout << "ENTER BASE = ";  cin >> b; //taking input  t\_base = b;  float area;  float area1; //declaring  area = (t\_height \* t\_base); //calculating  area1 = (area) \* 1 / 2;  t\_area = area1;  cout << "AREA OF TRIANGLE = " << t\_area << endl; //displaying  return t\_area; }  **output:**  **Text  Description automatically generated**  Text  Description automatically generated  **Uml:**  **Diagram  Description automatically generated**  Diagram  Description automatically generated |

**Task B: Virtual Functions & Pure Virtual Function [Marks: 20]**

***Protected Data Members of class Shape such as:***

c\_radius, s\_area, s\_length, s\_width, t\_base, t\_height (int)

c\_area, t\_area (float)

pie = 3.1416 (const int)

***Public Member Functions of class Shape such as:***

**void calculateArea() –** It will print “Parent Class is generic, your roll number and name”.

***Inherited Data Members of class Circle such as:***

It will inherit c\_radius and c\_area from shape class.

***Public Member Functions of class Circle such as:***

**void calculateArea () –** It will take input c\_radius and calculate area, store in c\_area and display area of the circle.

***Inherited Data Members of class Square such as:***

It will inherit s\_length, s\_width and s\_area from shape class.

***Public Member Functions of class Square such as:***

**void calculateArea () –** It will take input s\_length and s\_width and calculate area, store in s\_area and display area of the square.

***Inherited Data Members of class Triangle such as:***

It will inherit t\_base, t\_height and t\_area from shape class.

***Public Member Functions of class Triangle such as:***

**void calculateArea () –** It will take input t\_base and t\_height and calculate area, store in t\_area and display area of the triangle.

Create a **UML diagram.**

Do the following operations in main function:

1. Create a pointer object of class **Shape** and store the address of the Circle class and call member function **calculateArea()**.

2. Use a pointer object of class **Shape** and store the address of the Square class and call member function **calculateArea()**.

3. Use a pointer object of class **Shape** and store the address of the Triangle class and call member function **calculateArea()**.

4. Paste your output in the space below:

|  |
| --- |
| **Output:**  **Text  Description automatically generated** |

5. Write a virtual keyword with the member function **calculateArea()** e.g. virtual void calculateArea() of base class and paste the output in the space below:

|  |
| --- |
| **Function.cpp:**  void shape2::calculateArea() { //function of shape2 class  cout << "PARENT CLASS IS GENERIC" << endl;  cout << "NAME : NIMRA MAQBOOL" << endl;  cout << "ROLL NUMBER : BSCE21012 " << endl; //displaying }  void circle2::calculateArea() { //function of circle2 class  double x;  cout << "ENTER RADIUS = ";  cin >> x; //taking input  c\_radius = x;  double Area; //declaring  Area = (pie \* ((c\_radius) \* (c\_radius))); //calculating  s\_area = Area;  cout << "AREA = " << s\_area << endl; //displaying }  void square2::calculateArea() { //function of square2 class  float A;  cout << "ENTER A = ";  cin >> A; //taking input  s\_length = A;  s\_width = A;  float area; //declaring  area = s\_width \* s\_length; //calculating  s\_area = area;  cout << "AREA OF SQUARE = " << s\_area << endl; //displaying }  void triangle2::calculateArea() { //function of triangle2 class  float b;  float h;  cout << "ENTER HEIGHT = ";  cin >> h; //taking input  t\_height = h;  cout << "ENTER BASE = ";  cin >> b; //taking input  t\_base = b;  float area;  float area1; //declaring  area = (t\_height \* t\_base); //calculating  area1 = (area) \* 1 / 2;  t\_area = area1;  cout << "AREA OF TRIANGLE = " << t\_area << endl; //displaying }  **function.h:**  class shape2 { //created class of shape2 protected:  float s\_area;  double c\_radius;  float s\_length;  float s\_width;  float t\_base; //declared protected members  float t\_height;  float c\_height;  float t\_area;  const double pie = 3.1416; public:  virtual void calculateArea(); //declared public members };  class circle2 : public shape2 { //inherited shape2 class public:  void calculateArea(); //declared public members };  class square2 : public shape2 { //inherited shape2 class public:  void calculateArea(); //declared public members };  class triangle2 : public shape2 { //inherited shape2 class public:  void calculateArea(); //declared public members };  **uml:**  **Diagram  Description automatically generated**  **Output:**  **Text  Description automatically generated** |

6. Change virtual void calculateArea() of base class to virtual void calculateArea() = 0, The = 0 tells the compiler that the function has no body and above virtual function will be called **pure virtual function**.

7. Write your observations about virtual function and pure virtual function.

|  |
| --- |
| **Function.h:**  class shape3 { //created class of shape3 protected:  float s\_area;  double c\_radius;  float s\_length;  float s\_width;  float t\_base;  float t\_height;  float c\_height;  float t\_area;  const double pie = 3.1416; public:  virtual void calculateArea() = 0; //declared public members };  class circle3 : public shape3 { //inherited shape3 class public:  void calculateArea(); //declared public members };  class square3 : public shape3 { //inherited shape3 class public:  void calculateArea(); //declared public members };  class triangle3 : public shape3 { //inherited shape3 class public:  void calculateArea(); //declared public members };  **function.cpp:**  void circle3::calculateArea() { //function of circle3 class  double x;  cout << "ENTER RADIUS = ";  cin >> x; //taking input  c\_radius = x;  double Area; //declaring  Area = (pie \* ((c\_radius) \* (c\_radius))); //calculating  s\_area = Area;  cout << "AREA = " << s\_area << endl; //displaying }  void square3::calculateArea() { //function of square3 class  float A;  cout << "ENTER A = ";  cin >> A; //taking input  s\_length = A;  s\_width = A;  float area; //declaring  area = s\_width \* s\_length; //calculating  s\_area = area;  cout << "AREA OF SQUARE = " << s\_area << endl; //displaying }  void triangle3::calculateArea() { //function of triangle3 class  float b;  float h;  cout << "ENTER HEIGHT = ";  cin >> h; //taking input  t\_height = h;  cout << "ENTER BASE = ";  cin >> b; //taking input  t\_base = b;  float area;  float area1; //declaring  area = (t\_height \* t\_base); //calculating  area1 = (area) \* 1 / 2;  t\_area = area1;  cout << "AREA OF TRIANGLE = " << t\_area << endl; //displaying  }  **output:**  **Text  Description automatically generated**  **Uml:**  **Diagram  Description automatically generated**   1. **IN VIRTUAL THE VIRTUAL FUNCTION IS NOT COMPULSARAY TO CHANGE** 2. **BUT IN CASE OF PURE VIRTUAL S=WE HAVE TO WRITE THE CODE OF PURE VIRTUAL FUNCTION IN CHILD CLASS.** 3. **IT IS COMPULSARY TO OVERRIDE IT .** |

#### **Assessment Rubric for Lab**

**Method for assessment:**

Lab reports and instructor observation during lab sessions. Outcome assessed:

a. Ability to conduct experiments, as well as to analyze and interpret data (P) b. Ability to function on multi-disciplinary teams (A)

c. Ability to use the techniques, skills, and modern engineering tools necessary for engineering practice (P)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Performance metric** | **Task** | **CLO** | **Description** | **Max marks** | **Exceeds expectation** | **Meets expectation** | **Does not meet expectation** | **Obtained marks** |
| 1. Realization of experiment (a) | 1 | 1 | Functionality | 40 | Executes without errors excellent user prompts, good use of symbols, spacing in output. Through testing has been completed (35-40) | Executes without errors, user prompts are understandable, minimum use of symbols or spacing in output. Some testing has been completed (20-34) | Does not execute due to syntax errors, runtime errors, user prompts are misleading or non-existent. No testing has been completed (0-19) |  |
| 2. Teamwork (b) | 1 | 3 | Group Performance | 5 | Actively engages and cooperates with other group member(s) in effective manner (4-5) | Cooperates with other group member(s) in a reasonable manner but conduct can be improved (2-3) | Distracts or discourages other group members from conducting the experiment (0-1) |  |
| 3. Conducting experiment (a, c) | 1 | 1 | On Spot Changes | 10 | Able to make changes (8-10) | Partially able to make changes (5-7) | Unable to make changes (0-4) |  |
| 1 | 1 | Viva | 10 | Answered all questions (8-10) | Few incorrect answers (5-7) | Unable to answer all questions (0-4) |  |
| 4. Laboratory safety and disciplinary rules (a) | 1 | 3 | Code commenting | 5 | Comments are added and does help the reader to understand the code (4-5) | Comments are added and does not help the reader to understand the code (2-3) | Comments are not added (0-1) |  |
| 5. Data collection (c) | 1 | 3 | Code Structure | 5 | Excellent use of white space, creatively organized work, excellent use of variables and constants, correct identifiers for constants, No line-wrap (4-5) | Includes name, and assignment, white space makes the program fairly easy to read. Title, organized work, good use of variables (2-3) | Poor use of white space (indentation, blank lines) making code hard to read, disorganized and messy (0-1) |  |
| 6. Data analysis (a, c) | 1 | 4 | Algorithm | 20 | Solution is efficient, easy to understand, and maintain (15-20) | A logical solution that is easy to follow but it is not the most efficient (6-14) | A difficult and inefficient solution (0-5) |  |
| 7. Computer use (c) | 1 | 2 | Documentation & GitHub Submissions | 5 | Timely (4-5) | Late (2-3) | Not done (0-1) |  |
|  | Max Marks (total): | | | 100 | Obtained Marks (total): | | |  |

Lab Engineer Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_