**National University of Computer and Emerging Sciences**



**Lab Manual 11**

**Object Oriented Programming**

4th June, 2021

|  |  |
| --- | --- |
| Course Instructor | Miss Abeeda |
| Lab Instructor (s) | Siddiqua Nayyer  Dilawar Shabbir |
| Section | A |
| Semester | Spring 2021 |

Department of Computer Science

FAST-NU, Lahore, Pakistan

# **Objectives**

After performing this lab, students shall be able to:

* Perform exception handling.
* Performing Multiple inheritance

# **: Template Practice**

Consider the definition of the following function template:

*template <class type>*

*type surprise(type x, type y)*

*{*

*return x + y;*

*}*

Understand the code. What is the output of the following statements? Make a .cpp file to execute this code and add the output as comments in this file.

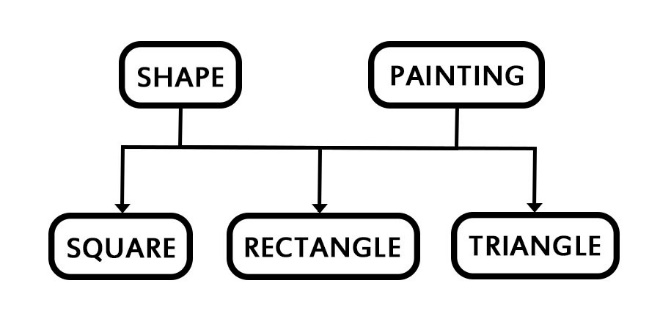
1. *cout << surprise(5, 7) << endl;*
2. *string str1 = "Sunny";*

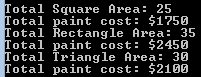
*string str2 = " Day";*

*cout << surprise(str1, str2) << endl;*

# **Exercise 1: Multiple Inheritance**

In this exercise, we are working with the following classes to implement multiple inheritance: **Shape, Painting, Square, Rectangle,** and **Triangle.**

Implement the class hierarchy given below:

1. Class **Shape** has following protected data members: **length, breadth, height.**
2. Class **Square, Rectangle,** and **Triangle** all have **getArea()** method.
   1. Area of square = length \* lenth
   2. Area of rectangle = length\* breadth
   3. Area of triangle = ½ \* breadth \* height
3. Class **Painting** has method **getCost(area)** which returns the cost of painting a shape by multiplying paint cost with area of the shape.
4. For simplicity, you only create **one .h** file which contains the headers of all classes and **one .cpp** file which contains the implementation of all classes. Also, you are not required to use polymorphism in this question
5. Consider the following output(You can input your own values):

**Exercise 2:**

Create two classes named Mammals and MarineAnimals. Create another class named

BlueWhale which inherits both the above classes. Now, create a function in each of these

classes which prints &quot;I am mammal&quot;, &quot;I am a marine animal&quot; and &quot;I belong to both the

categories: Mammals as well as Marine Animals&quot; respectively. Now, create an object for

each of the above class and try calling

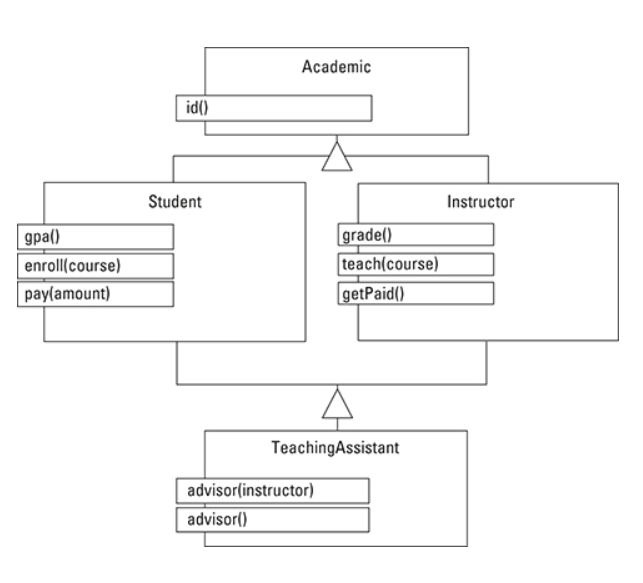
1 - function of Mammals by the object of Mammal

2 - function of MarineAnimal by the object of MarineAnimal

3 - function of BlueWhale by the object of BlueWhale

4 - function of each of its parent by the object of BlueWhale.

**Exercise 3:**



For the given above class diagram, Implement the entire hierarchy of the Class diagram i.e.

define all the classes along with their attributes and their inheritance.

**Exception Handling**

# **: Exception Handling Practice**

Consider the following C++ code:

int numOfItems;

double unitCost;

try

{

cout << "Enter the number of items: ";

cin >> numOfItems;

cout << endl;

if (numOfItems < 0)

throw numOfItems;

cout << "Enter the cost of one item: ";

cin >> unitCost;

cout << endl;

if (unitCost < 0)

throw unitCost;

cout << "Total cost: $"

<< numOfItems \* unitCost << endl;

}

catch (int num)

{

cout << "Negative number of items: " << num

<< endl;

cout << "Number of items must be nonnegative."

<< endl;

}

catch (double dec)

{

cout << "Negative unit cost: " << dec

<< endl;

cout << "Unit cost must be nonnegative."

<< endl;

}

Answer the following:

1. What is the output if the input is 25 5.50?
2. What is the output if the input is -55 2.8?
3. What is the output if the input is 37 -4.5?
4. What is the output if the input is -10 -2.5?

# **Exercise 4: Exception Handling Problem**

Write a program that prompts the user to enter a person’s date of birth in numeric form such as 8-27-1980. The program then outputs the date of birth in the form: August 27, 1980. Your program must contain three exception classes: **invalidDay, invalidMonth,** and **invalidYear**. If the user enters an invalid value for day, then the program should throw and catch an **invalidDay** object. Follow similar convention for the invalid values of month. Handle leap year value with **invalidYear** exception.