**National University of Computer and Emerging Sciences**

Logo

Description automatically generated with medium confidence

**Lab Manual # 13**

**Object Oriented Programming**

| Course Instructor | Mr. Waqas Manzoor |
| --- | --- |
| Lab Instructor (s) | Hira Ilyas  Sidra Zafar |
| Section | BSE-2A |
| Semester | Spring 2022 |

Department of Computer Science

FAST-NU, Lahore, Pakistan

### **Downcasting and Object Slicing**

### **Exercise 1:**

Make a new application called Lab\_<your roll number>. Define and implement a class **Shape**. This class should provide:

* A string that holds the name of the shape.
* A constructor which takes string as a parameter to initialize the name of shape.
* A virtual function print() which prints out the point on the screen.
* A destructor which prints “~Shape() called” on the screen.

### **Exercise 2:**

Now define and implement a class Square and inherit it from Shape class. This class should contain:

* A private data member length that will store length of the sides of square.
* A constructor which takes two parameters (name and length) and initialize the data members accordingly.
* A destructor which prints “~Square() called” on the screen.
* A function print() which prints the information i.e. name and Area of the square .

### **Exercise 3**

Define and implement a class Rectangle and inherit it from Shape class. This class should provide:

* Two private data members length and width indicating the length and width of the rectangle.
* A constructor which takes three parameters (name , length and width) and initializes the data members accordingly.
* A destructor which prints “~Rectangle() called” on the screen.
* A function print() which prints the information i.e. name and Area of the rectangle .

Now, run the following main.

Square s1(“Quadliteral”, 4);

Shape \*shape\_ptr = &s1;

shape\_ptr->print();

This code compiles and runs without any problem because **shape\_ptr** points to an object of **square** class.

What will happen, if we try to downcast a base class pointer that is pointing to an object of the base class and not to an object of derived class? Try to compile and run this code:

Shape s(“Circle”);

Square \*square\_ptr = (Square\*)(&s);

Rectangle \*rec\_ptr = (Rectangle\*)(&s);

square\_ptr->print();

rec\_ptr->print();

Write your answer and the reason below:

|  |
| --- |

When you try to downcast base class pointer (Shape) that is not actually pointing up an object of the derived class (Square), you will get access to the memory that does not have any information about the derived class object. This is the main danger of downcasting.

You can use a safe cast that can help you to know if one type can be converted correctly to another type. For this purpose, use a dynamic cast.

Add two global functions in the code.

Shape\* create\_square(string S\_name, int value){

return new Square(S\_name, value);

}

Rectangle\* create\_rectangle(string S\_name, int len, int wid){

return new Rectangle(S\_name, len, wid);

}

Now try to run the below code and report the output.

Shape \*quad = create\_square("Quadliteral", 4);

Square\* sq = dynamic\_cast<Square\*>(quad);

if(sq){

sq -> get\_info();

}

Write your answer and the reason below:

|  |
| --- |

Now try to run the below code.

Rectangle \*rect = create\_rectangle("Quadliteral", 4, 5);

Shape\* quad1 = dynamic\_cast<Shape\*>(rect);

Square\* sq1 = dynamic\_cast<Square\*>(quad1);

if(sq1 == NULL){

cout<<"Invalid casting."<<endl;

}

Write your answer and the reason below:

|  |
| --- |

### **Exercise 4:**

Use the above two classes name Shape and Square. Modify their print functions such that along with the information it prints “ I am Base class Object/ I am derived class object”.

Now add a global finction named getInfo(Shape obj) with takes the base class object.

Run the following main and explain your answer.

Shape obj1(“Circle”);

Square obj2(“Quadliteral”,8);

getInfo(obj1);

getInfo(obj2);

### **Exception Handling**

# **Exercise 5: 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 6: Exception Class Practice**

Define an exception **class** called **tornadoException**. The class should have two constructors including the default constructor. If the exception is thrown with the default constructor, the method **what** should return **"Tornado: Take cover immediately!"** The other constructor has a single parameter, say **m**, of the **int** type. If the exception is thrown with this constructor, the method **what** should return **"Tornado: m miles away; and approaching!"** Write a C++ driver program to test the **class tornadoException.**

# **Exercise 7: 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.

### 

### 