National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Department of Computing

CS212: Object Oriented Programming

Class: BSCS-5AB

Lab 8: Inheritance & polymorphism

Date: 25-04-2016

Time: 09:00 am – 12:00 pm / 2pm to 5pm

Instructor: Ms. Hirra Anwar

Lab 8: Inheritance & Polymorphism in OOP C++

Introduction

Inheritance is the practice of passing on property, titles, debts, rights and obligations upon the death of an individual. It represents also to pass a characteristic, genetically. (Wikipedia)
In object-oriented programming (OOP), **inheritance** is a way to reuse code of existing objects, or to establish a subtype from an existing object.

Objectives

- Introduction to Inheritance
- Hierarchies
- Inheritance in Classes
- Polymorphism concepts

Tools/Software Requirement

You will need Visual Studio.

Description

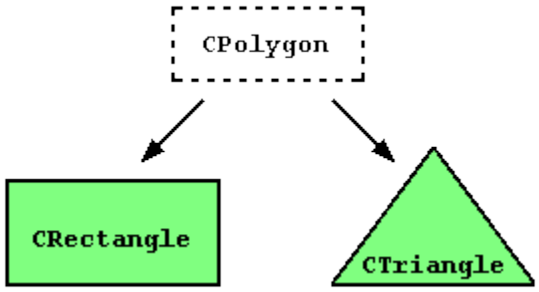
Follow all the content given below to understand the concept of the given topics. Run all the given examples to get a better understanding. You might need to use both inheritance and polymorphism concepts.

Lab Task 1

Follow the description with the lab practices provided below.

Inheritance between classes

This could be represented in the world of classes with a class CPolygon from which we would derive the two other ones: CRectangle and CTriangle.



Lab Practice 1

```
// derived classes
#include <iostream>
using namespace std;

class CPolygon {
  protected:
    int width, height;
  public:
    void set_values (int a, int b)
      { width=a; height=b;}
  };

class CRectangle: public CPolygon {
  public:
    int area ()
      { return (width * height); }
  };

class CTriangle: public CPolygon {
  public:
    int area ()
      { return (width * height / 2); }
  };

int main () {
  CRectangle rect;
  CTriangle trgl;
  rect.set_values (4,5);
  trgl.set_values (4,5);
  cout << rect.area() << endl;
  cout << trgl.area() << endl;
  return 0;
```

```
}
```

What is inherited from the base class?

In principle, a derived class inherits every member of a base class except:

- its constructor and its destructor
- its operator=() members
- its friends

Although the constructors and destructors of the base class are not inherited themselves, its default constructor (i.e., its constructor with no parameters) and its destructor are always called when a new object of a derived class is created or destroyed.

If the base class has no default constructor or you want that an overloaded constructor is called when a new derived object is created, you can specify it in each constructor definition of the derived class:

derived_constructor_name (parameters) : base_constructor_name (parameters) {...}

Lab Practice 2

```
// constructors and derived classes
#include <iostream>
using namespace std;

class mother {
  public:
    mother ()
      { cout << "mother: no parameters\n"; }
    mother (int a)
      { cout << "mother: int parameter\n"; }
};

class daughter : public mother {
  public:
    daughter (int a)
      { cout << "daughter: int parameter\n\n"; }
};
```

```
class son : public mother {
  public:
    son (int a) : mother (a)
      { cout << "son: int parameter\n\n"; }
};

int main () {
  daughter cynthia (0);
  son daniel(0);

  return 0;
}
```

Multiple inheritance

In C++ it is perfectly possible that a class inherits members from more than one class. This is done by simply separating the different base classes with commas in the derived class declaration. For example, if we had a specific class to print on screen (COutput) and we wanted our classes CRectangle and CTriangle to also inherit its members in addition to those of CPolygon we could write:

Lab Practice 3

```
// multiple inheritance
#include <iostream>
using namespace std;

class CPolygon {
  protected:
    int width, height;
  public:
    void set_values (int a, int b)
      { width=a; height=b;}
  };

class COutput {
  public:
    void output (int i);
  };

void COutput::output (int i) {
  cout << i << endl;
```

```
  }

class CRectangle: public CPolygon, public COutput {
  public:
    int area ()
      { return (width * height); }
  };

class CTriangle: public CPolygon, public COutput {
  public:
    int area ()
      { return (width * height / 2); }
  };

int main () {
  CRectangle rect;
  CTriangle trgl;
  rect.set_values (4,5);
  trgl.set_values (4,5);
  rect.output (rect.area());
  trgl.output (trgl.area());
  return 0;
}
```

Lab Task 2

Create a class Person which has derived classes Faculty and Student. Now derive class TA from classes Faculty and Student. Implement constructors and print function for each class.

Lab Task 3

Implement the following class hierarchy and write a program to test it

For the given classes *container.cpp*, *circle.cpp*, *Box.cpp* and *Cylinder.cpp* implement it according to the structure given in the diagram.

1. Using Cylinder.cpp, calculate the cylinder **volume** and **area** according to the scenario presented in the diagram.
2. Find the ***liquid percent loaded*** in the both container (e.g. Box and Cylinder)
3. Using Box.cpp calculate the volume of the box.

Hints:

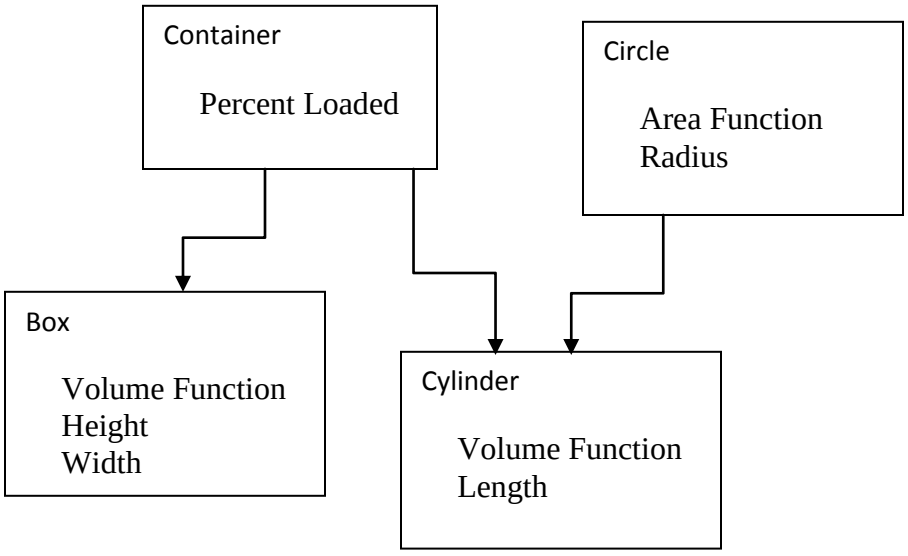
1. Area of the Circle

$A = \pi \cdot r \cdot r$

2. Volume of the cylinder

$V = \pi r^2 h$

3. Volume of the Box = Length*Width*Height
4. Percent loaded can be any quantity you can mention in percentage e.g. Box is 75% filled.



Lab Task 4:

(Package *Inheritance Hierarchy*) Package-delivery services, such as FedEx®, DHL® and UPS®, offer a number of different shipping options, each with specific costs associated. Create an inheritance hierarchy to represent various types of packages. Use class Package as the base class of the hierarchy, then include classes TwoDayPackage and OvernightPackage that derive

from Package. Base class Package should include data members representing the name, address, city, state and ZIP code for both the sender and the recipient of the package, in addition to data members that store the weight (in ounces) and cost per ounce to ship the package. Package's constructor should initialize these data members. Ensure that the weight and cost per ounce contain positive values. Package should provide a public member function calculateCost that returns a double indicating the cost associated with shipping the package. Package's calculateCost function should determine the cost by multiplying the weight by the cost per ounce. Derived class TwoDayPackage should inherit the functionality of base class Package, but also include a data member that represents a flat fee that the shipping company charges for two-day-delivery service. TwoDayPackage's constructor should receive a value to initialize this data member. TwoDayPackage should redefine member function calculate Cost so that it computes the shipping cost by adding the flat fee to the weight-based cost calculated by base class Package's calculateCost function. Class OvernightPackage should inherit directly from class Package and contain an additional data member representing an additional fee per ounce charged for overnight-delivery service. OvernightPackage should redefine member function calculateCost so that it adds the additional fee per ounce to the standard cost per ounce before calculating the shipping cost. Write a test program that creates objects of each type of Package and tests member function calculateCost.

Deliverables

Source code for lab tasks.