NATIONAL UNIVERSITY OF COMPUTER & EMERGING SCIENCES

ISLAMABAD CAMPUS

OBJECT ORIENTED PROGRAMMING (CS217) – Lab SPRING 2020

POLYMORPHISM LAB 13

Virtual Functions:

A virtual function is a member function of the base class that is overridden in derived class. The classes that have virtual functions are called polymorphic classes.
The compiler binds virtual function at runtime, hence called **runtime polymorphism**.

```
Class Base {
    // data members
   public:
      //virtual function
  virtual return_Type func_name( args.. )
      {
        //function definition
      }
 }
```

```
Class derive : Base {
 // data members

   public:
       return_Type func_name( args.. )
      {
        //function definition
      }
```

Abstract class:

An abstract class is a class that is designed to be specifically used as a base class. An abstract class contains at least one **pure virtual function**. You declare a pure virtual function by using a pure specifier (= 0) in the declaration of a virtual member function in the class declaration.

```
class Base
{
    // Data members of class
public:
    // Pure Virtual Function
    virtual void show() = 0;
   /* Other members */
};
```

```
class Derived: public Base
{
// Data members of class
public:
    void show()
     {
          cout << "fun() called";
     }
};
```

Note: The main difference between '**virtual function**' and '**pure virtual function**' is that '**virtual function**' has its definition in the **base class** and also the inheriting derived classes redefine it. The **pure virtual function** has no definition in the base class, and all the inheriting derived classes have to redefine it.

Question 1:

Write a program to calculate the area of following shapes by using Inheritance. The base class is "**shape**" and the derived classes are **rectangle**, **triangle** and **circle**.
Attributes of all the classes are as under: (decide which attribute should be protected/private/public)
Class shape:

- String type
- float Width
- float height

Class Rectangle: (No additional attribute)
Class Triangle:

- Float base

Class Circle:

- float radius

1. Create mutator (Set) and accessor (Get) functions for all attributes of all classes.
2. Create a default constructor that initializes all of the attributes to default values.
3. Create a **virtual function** of **area ()** in shape class that does not do anything. (This function will be Overload by classes that inherit the base class)
4. Provide a **virtual display () function** in Shape provide implementation of display function for all classes, in Shape Class, as the function Display the value of type as "Shape". In Rectangle the Display function should display

```
cout<<"Type : "<<type;
cout<<"Width :"<<width;
cout<<"Height :"<<height;
```

5. Similarly provide the implementation of function display for all rest of classes according to their member functions. Now call the area function for each child class in **main()** to compute area. Call the display function as well. Also run **test-case** for this program

Note: If we make any function inside a base class virtual, then that function becomes virtual in all its derived classes. This means that we don't need to declare that function as virtual separately in its derived classes.

Question 2:

Suppose there are some employees working in a firm. The firm hires only two types of employee: either **driver** or **developer**

Now, you have to develop a software to store information about them. So, here is an idea there is no need to make objects of employee class. We will make objects to only driver or developer. Also, both must have some salary. So, there must be a common function to know about salary.

This need will be best accomplished with **abstract class**.

So, we can make 'Employee' an abstract class and 'Developer' and 'Driver' its subclasses.

```
class Employee                    //  abstract base class
{
virtual int getSalary() = 0;    // pure virtual function
};
```

Here **getSalary()** function in the class **Employee** is a pure virtual function. Since the Employee class contains this pure virtual function, therefore it is an **abstract base class**.
Since the abstract function is defined in the subclasses, therefore the function 'getSalary()' is defined in both the subclasses (**driver & developer**)of the class Employee.
Attribute for Employe class: (decide which attribute should be protected/private/public)
int Emp_no
Attribute for driver class:
Int salary
Attribute for developer class:
Int salary

1. Write default and parameterized constructors to initialize attributes of all classes. .
2. Make Employee class Abstract by declaring at least one pure virtual function getSalary() . You do not need to provide body for it as it is a pure virtual function and can only be implemented by child class of Employee. A pure virtual function is declared as below virtual float getSalary()=0;
3. Write a class driver, make it a child of Employee, declare its member function getsalary()
4. Write a class developer, make it a child of Employee, declare its member function getsalary()..
5. Since Employee class is abstract and cannot be instantiated, but we can create a pointer of it and make it point to the objects of child classes' one by one, i.e.

```
Base* ptr=new child (1, 4, 6)
```

Similarly instantiate all child classes in **main ()**. Also run **test-case** for this program

Question 3:

Write a program to calculate the perimeter of Shape class where shape class is an interface with two **pure virtual functions** i.e. **getarea**() and **getperimeter**() .These virtual functions will be implemented (defined) in its subclasses **Rectangle** and **Square** according to their requirements. So, an interface (shape) is just an abstract class with all pure virtual methods.

```
class Shape

{ public:

   virtual int getArea() = 0;

   virtual int getPerimeter() = 0;

}; is a pure virtual function, so we do not need to implement here
```

Attributes of Rectangle class:

- Int length
- Int width

Area of rectangle= width*height

Perimeter of recatangle= 2*(length +width)

Attributes of Square class:

- Side

Area of Square= side*side

Perimeter of square=4*side

Now call the getarea() & getperimeter() functions for each child class in **main()** to compute value.