National University of Computer & Emerging Sciences, Karachi
Computer Science Department
Spring 2021, Lab Manual - 10

Course Code: CL-217	**Course : Object Oriented Programming Lab**
Instructor(s) :	**Nida Munawar, Abeer Gouhar, Romasha Khurshid, M. Fahim, Sohail Afzal, Qaiser Abbas, Ali Fatmi**

Contents:

Abstract Classes:

Sometimes implementation of all functions cannot be provided in a base class because we don't know the implementation. Such a class is called abstract class. For example, let Shape be a base class. We cannot provide implementation of function draw() in Shape, but we know every derived class must have implementation of draw(). Similarly an Animal class doesn't have implementation of move() (assuming that all animals move), but all animals must know how to move. We cannot create objects of abstract classes.

Virtual Functions:

A pure virtual function (or abstract function) in C++ is a virtual function for which we don't have implementation (we do not have function body), we only declare it. A pure virtual function is declared by assigning 0 in declaration.

See the following example.

```
// An abstract class
class Test
{
    // Data members of class
public:
    // Pure Virtual Function
    virtual void show() = 0;

   /* Other members */
};
```

Note:

- The = 0 syntax doesn't mean we are assigning 0 to the function. It's just the way we define pure virtual functions.
- A pure virtual function is used if a function does not have any use in the base class but function must be implemented by all of its derived classes
- A class is abstract if it has at least one pure virtual function. In example given above, Test is abstract class as it has virtual function show().

A complete example :

A pure virtual function is implemented by classes which are derived from a Abstract class. Following is a simple example to demonstrate the same.

```
#include<iostream>
using namespace std;

class Base
{
  int x;
public:
   virtual void fun() = 0;
   int getX() { return x; }
};

// This class inherits from Base and implements fun()
class Derived: public Base
{
   int y;
public:
   void fun() { cout << "fun() called"; }
};

int main(void)
{
   Derived d;
   d.fun();
   return 0;
}
```

Output:

```
fun() called
```

- Object of abstract class cannot be created

```
// pure virtual functions make a class abstract
```

```
#include<iostream>
using namespace std;

class Test
{
   int x;
public:
    virtual void show() = 0;
    int getX() { return x; }
};

int main(void)
{
    Test t;
    return 0; }
```

Output:

```
Compiler Error: cannot declare variable 't' to be of abstract
 type 'Test' because the following virtual functions are pure
within 'Test': note:     virtual void Test::show()
```

- An abstract class can have constructors. Consider this code that compiles and runs fine.

```
#include<iostream>
using namespace std;

// An abstract class with constructor
class Base
{
protected:
   int x;
public:
  virtual void fun() = 0;
  Base(int i) { x = i; }
};
class Derived: public Base
{
    int y;
public:
    Derived(int i, int j):Base(i) { y = j; }
    void fun() { cout << "x = " << x << ", y = " << y; }
};
```

```
int main(void)
{
   Derived d(4, 5);
   d.fun();
   return 0; }
```

Output:

```
   x = 4, y = 5
```

Example: Abstract class and Virtual Function in calculating Area of Square and Circle:

```
// C++ program to calculate the area of a square and a circle

#include <iostream>
using namespace std;

// Abstract class
class Shape {
  protected:
   float dimension;

  public:
   void getDimension() {
      cin >> dimension;
   }

   // pure virtual Function
   virtual float calculateArea() = 0;
};

// Derived class
class Square : public Shape {
  public:
   float calculateArea() {
      return dimension * dimension;
   }
};
```

```
// Derived class
class Circle : public Shape {
  public:
   float calculateArea() {
      return 3.14 * dimension * dimension;
   }
};

int main() {
   Square square;
   Circle circle;

   cout << "Enter the length of the square: ";
            square.getDimension();
   cout << "Area of square: " << square.calculateArea() << endl;

   cout << "\nEnter radius of the circle: ";
   circle.getDimension();
   cout << "Area of circle: " << circle.calculateArea() << endl;
   return 0;
}
```

Output

```
Enter length to calculate the area of a square: 4
Area of square: 16

Enter radius to calculate the area of a circle: 5
Area of circle: 78.5
```

In this program, `virtual float calculateArea() = 0;` inside the `Shape` class is a pure virtual function.

That's why we must provide the implementation of `calculateArea()` in both of our derived classes, or else we will get an error.

Lab Tasks:

Task 1:

Define abstract class “Shape” that provides interface (through virtual functions) to the two derived classes “Rectangle” and “Triangle” to implement the function called “getArea()”. The program will output area of rectangle and area of triangle.

Task 2:

Create an abstract class Car (model, price). Include get and set methods from these fields. The setPrice Method must be abstract. Create two subclasses Color() and Company() from Car and include appropriate setPrice method in these classes. Finally write a code that uses Car class and sub classes to display information about instances.

Task 3:

Write a program to calculate bonus of the employees. The class master derives the information from both admin and account classes which intern derives information from class person. Create base and all derived classes having same member functions called getdata, display data and bonus. Create a base class pointer that capable of accessing data of any class and calculates bonus of the specified employee. (Hint: Use virtual functions)