Object Oriented Programming

LAB 12

Abstract classes, Virtaul Function, Interfaces

NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES

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, and different kinds of shapes can be children of the shape class. Therefore, we cannot provide one 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, and many animals have different ways to move. Eg. Humans move differently than snakes), but all animals must know how to move.
Lastly, we cannot create objects of abstract classes, as they are "incomplete" because they lack the function definitions in their class body.
An abstract class must have at least one pure virtual function.

Virtual Functions

A C++ virtual function is a member function in the base class that you redefine in a derived class. It is declared using the virtual keyword. A virtual function must be defined in the base class, even though it is not used. The prototypes of a virtual function of the base class and all the derived classes must be identical. If the two functions with the same name but different prototypes, C++ will consider them as the overloaded functions. We covered these in our previous lab!

Pure 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;
}
```

```
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;
}
```

```
Compiler Error: cannot declare
    variable 't' to be of abstract
type 'Test' because the following
    virtual functions are pure
```

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;
}
```

```
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;
}
```

```
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.

Important things to remember about abstract classes:

1. A class is considered abstract if it has at least one pure virtual function in it.
2. If a class inherits an abstract class and does not override the pure virtual function, then that (derived) class also becomes an abstract class, and cannot be instantiated.
3. A derived class inheriting from an abstract class is called a concrete class if it overrides the pure virtual function present in the abstract base class.
4. While you can't instantiate an object of an abstract class, you can still create a pointer for it. **This pointer can point to of objects of its concrete child classes**. This is helpful when we are generalizing things, and also helps with achieving runtime polymorphism.

Question#1:-

Design and implement a program that shows the relationship between person, student and professor. Your person class must contain two pure virtual functions named getData() of type void and isOutstanding() of type bool and as well getName() and putName() that will read and print the person name. Class student must consist of function name getData (), which reads the GPA of specific person and isOutstanding() function which returns true if the person GPA is greater than 3.5 else should return false. Class professor should take the respective persons publications in getData() and will return true in isOutstanding() if publications are greater than 100 else will return false . Your main function should ask the user either you want to insert the data in professor or student until and unless user so no to add more data.

Question#2:-

Imagine you're developing software for a newly opened gourmet restaurant called "Savory Delights." As part of the restaurant management system, you're tasked with implementing a module for handling menu items. The head chef at Savory Delights has provided detailed instructions for preparing and serving various types of dishes, ranging from appetizers to main courses and desserts. To facilitate this, you decide to create an abstract class called `MenuItem`, which defines pure virtual functions for `prepare()` and `serve()`. Additionally, you plan to create derived classes such as `Appetizer`, `MainCourse`, and `Dessert`, each tailored to the specific requirements of its category. For example, an `Appetizer` might involve assembling delicate hors d'oeuvres with precision, a `MainCourse` could require complex cooking techniques and plating arrangements to showcase the chef's culinary expertise, and a `Dessert` might involve intricate decoration and garnishing to delight the diner's senses. To ensure the software meets the chef's exacting standards, you'll need to test it rigorously by creating instances of each menu item type and verifying that the `prepare()` and `serve()` methods are executed flawlessly, replicating the meticulous attention to detail required in a high-end restaurant kitchen. This scenario-based approach not only demonstrates your ability to translate culinary concepts into software design but also ensures that the restaurant management system enhances the dining experience at Savory Delights by delivering impeccably prepared and presented dishes to discerning customers.