Nov 19, 2015

Computer Programming

ABDUL AZIZ

Lab Instructor	Abdul Aziz
Course	Computer Programming Lab
Duration	2hrs

Objectives:

In this lab, following topics will be covered:

- Virtual & Pure Virtual Functions
- Abstract Class

Consider the following example where a base class has been derived by other two classes:

```
#include <iostream>
using namespace std;
class Shape
{
   protected:
   int width, height;
   public:
   Shape( int a=0, int b=0)
     {
       width = a;
       height = b;
     }
     int area()
     {
        cout << "Parent class area :" <<endl;
        return 0;
     }
};
class Rectangle: public Shape
{
   public:
   Rectangle( int a=0, int b=0)
     {
       Shape(a, b);
     }
     int area ()
     {
        cout << "Rectangle class area :" <<endl;
        return (width * height);
     }

};
```

```
class Triangle: public Shape
{
   public:
   Triangle( int a=0, int b=0)
      {
       Shape(a, b);
      }
      int area ()
      {
         cout << "Triangle class area :" <<endl;
         return (width * height / 2);
      }
};
// Main function for the program
int main( )
{
   Shape *shape;
   Rectangle rec(10,7);
   Triangle  tri(10,5);
   // store the address of Rectangle
   shape = &rec;
   // call rectangle area.
   shape->area();
   // store the address of Triangle
   shape = &tri;
   // call triangle area.
   shape->area();
   return 0;
}
```

When the above code is compiled and executed, it produces following result:

```
Parentclass area
Parentclass area
```

VIRTUAL FUNCTION:

```
classShape{
protected:
int width, height;
public:
Shape(int a=0,int b=0)
{
	width = a;
	height = b;
}
virtualint area()
{
	cout <<"Parent class area :"<<endl;
return0;
}
};
```

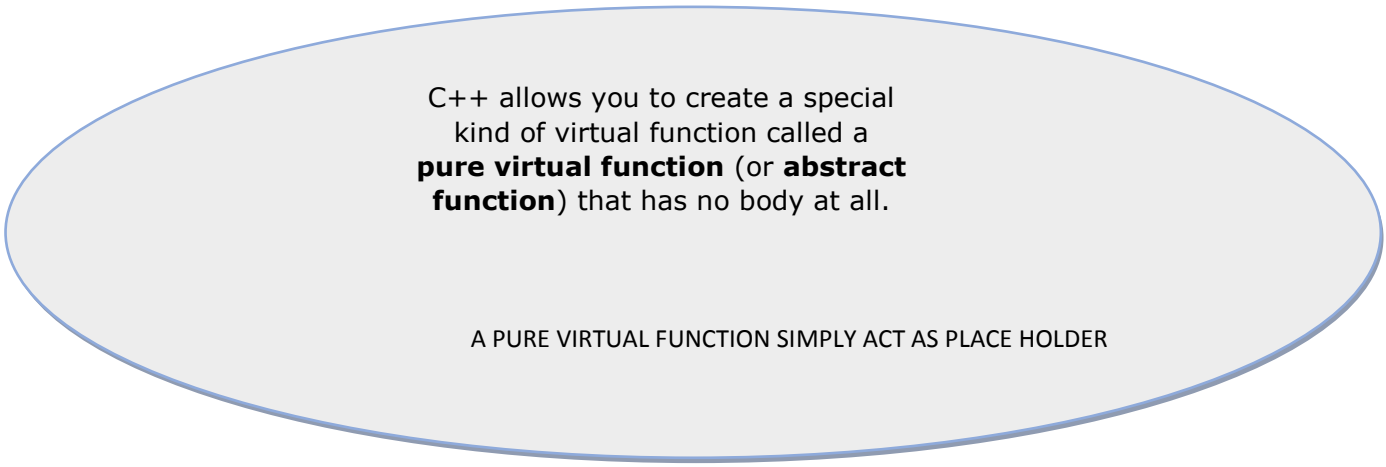
After this slight modification, when the previous example code is compiled and executed, it produces following result:

```
Rectangleclass area
Triangleclass area
```

- **PURE VIRTUAL FUNCTIONS**

Pure virtual functions are declared in the regular way, but the declaration ends with =0. This means, that

We don't want to define the function right now.



- **ABSTRACT CLASS:**

If a class contains at least one pure virtual function, then we name it abstract class.

IMPORTANT POINTS:

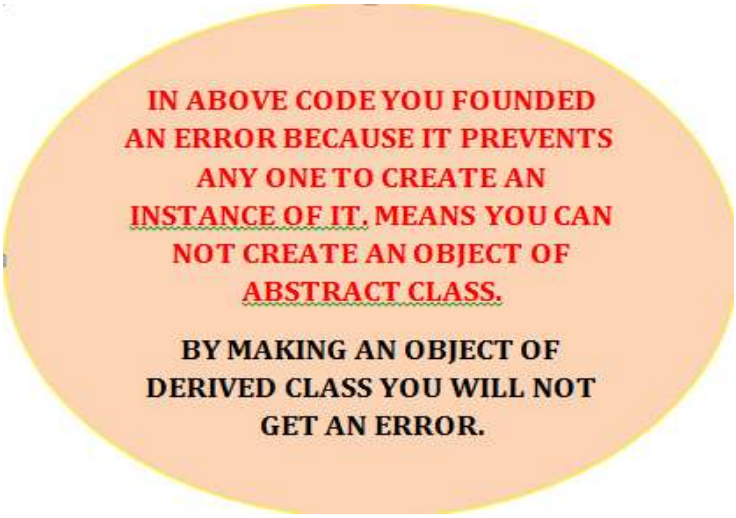
- **Allows the base class to provide only an interface for its derived classes.**
- **Prevents anyone from creating an instance of this class.**
- **A class is made abstract if atleast one pure virtual function defined.**

EXAMPLE 1:

```
#include <iostream>
using namespace std;
class MyInterface
{
 public:
 virtual void Display() = 0;
};
class MyClass1 : public MyInterface
{
 public:
 void Display() {
 cout << "MyClass1" << endl;
```

```
 }
};
class MyClass2 : public MyInterface
{
 public:
 void Display()
 { cout << "MyClass2" << endl;
 }
};
int main()
{            MyInterface I;
}
```

NOTE:

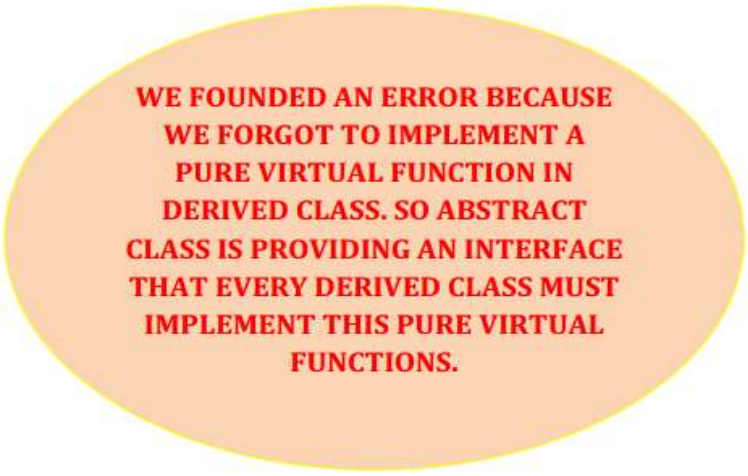


EXAMPLE 1 CONTINUED:

```
#include <iostream>
using namespace std;
class MyInterface
{
 public:
 virtual void Display() = 0;
};
class MyClass1 : public MyInterface
{
 public:

 void Display() {
 cout << "MyClass1" << endl;
 }
};
class MyClass2 : public MyInterface
```

```
{
 public:
 void Derived()
 {
 cout << "This is my derived class" << endl;
 }
};
main()
{
MyInterface I;
}
```



Exercise

1. **VIRTUAL FUNCTIONS**
   Consider the following class:

```
class Sale

{

 public:

 Sale( );

 Sale( double thePrice);

 double getPrice( ) const;

 void setPrice( double newPrice);

 virtual double bill( ) const;

 double savings(const Sale& other) const;

//Returns the savings if you buy other instead of the calling object.

 private:

 double price;

};
```

- **Implement the complete Sale class.**
- **Create a DiscountSale child class of Sale class. It has following data members and function members:**
  1. double discount.
  2. double bill( ) const

You need to redefine bill function in child class.

2. PURE VIRTUAL FUNCTIONS

Implement an abstract class Machine with a run method.

Then derive two concrete subclasses from Machine: WashingMachine, and Refrigerator.