Lab 09

Inheritance

OBJECTIVE:

Things that will be covered in today's lab:

- Inheritance

THEORY:

Inheritance: The process by which a class incorporates the attributes and behaviors of a previously defined class.The new classes that we create from the existing classes are called*derived* classes and the existing classes are called *base* classes. Derived classes inherit the properties of base classes. Therefore, rather than creating completely new classes from scratch, we can take advantage of inheritance and reduce software complexity.Each derived class, in turn, becomes a base class for a future derived class. Inheritance can be either single inheritance or multiple inheritances. In single inheritance, the derived class is derived from a single base class; in multiple inheritances, the derived class is derived from more than one base class.

```
class derived-class: access-Specifier base-class
{
//member list
};
```

Where *access-specifier* is one of **public, protected,** or **private**, and base-class is the name of a previously defined class. If the *access-specifier* is not used, then it is private by default.

When deriving a class from a base class, the base class may be inherited through *public*, *protected* or*private* inheritance. The type of inheritance is specified by the *access-specifier* as explained above.

We hardly use *protected* or *private* inheritance, but public inheritance is commonly used. While using different type of inheritance, following rules are applied:

Public Inheritance:	When deriving a class from a public base class, public members of the base class become public members of the derived class and protectedmembersof the base class become protected members of the derived class. A base class's private members are never accessible directly from a derived class, but can be accessed through calls to the public and protected members of the base class
Protected Inheritance:	When deriving from a protected base class, public and protectedmembersof the base class become protected members of the derived class.

Private Inheritance: When deriving from a private base class, public and protectedmembersof the base class become private members of the derived class.

Multiple Inheritances:

C++ class can inherit members from more than one class and here is the extended syntax:

```
class derived-class: accessbase-a, acess base-b...
{
//member list

};
```

Where *access* is one of **public, protected,** or **private** and would be given for every base class and they will be separated by comma as shown above. Let us try the following example:

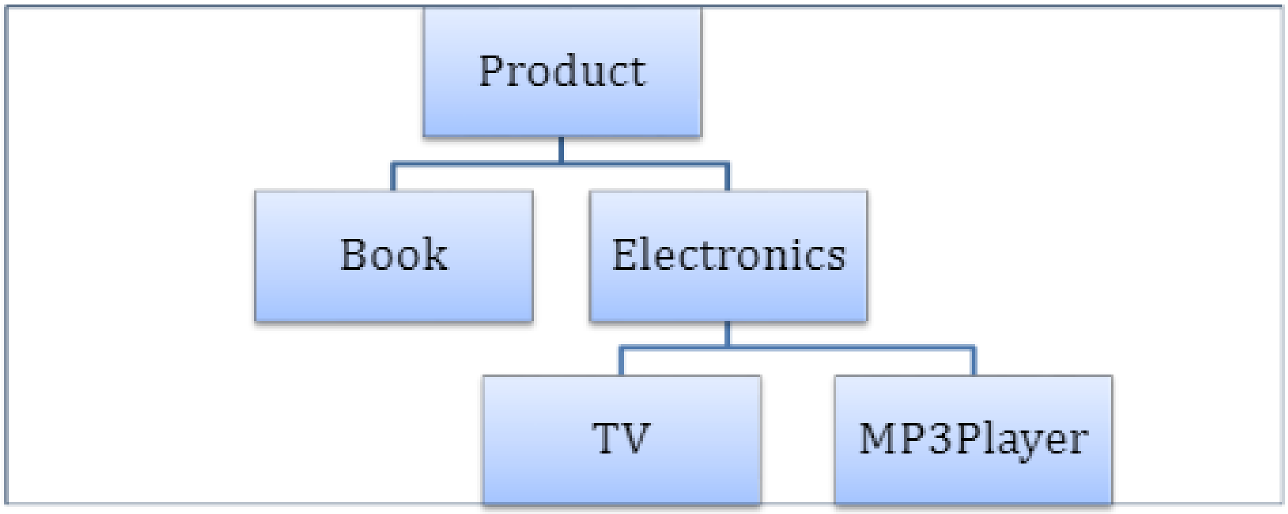
Example: What should be the output of this program?

```
class Shape
{
public:
void setWidth(int w)  { width = w; }
void setHeight(int h) { height = h;}
private:
int width;
int height;
};
class Rectangle: public Shape
{
public:
int getArea() { return (width * height); }
};
int main()
{
     Rectangle Rect;
     Rect.setWidth(5);
     Rect.setHeight(7);

     cout <<"Total area: "<< Rect.getArea() << endl;
return 0;
}
```

Exercise 1:

We want to store some details about an **online shop**. This shop has several kinds of products. Each product is represented by a class. The products are categorized intro Electronics and Books. The Electronics is further subdivided into MP3Player and Television. The hierarchy is shown below:



For each product, system needs to store its regular price and discount. The system should provide a functionality to set or get a price for each product. For the products that are categorized as electronics, the name of the manufacturer should be stored. For MP3Players, system should store the *color* and *capacity* (in Giga Bytes). The system stores the size of each television. For each book, the following information is needed: Name of the book and Author's name. For each product, the system should calculate its discounted price.

Apply object oriented programming concepts and implement this system in C++. Implement setter and getter functions for each class and all other required functions. Use the following main().

```
void main(){
int choice1=0,choice2=0,capacity,size;
double price,discount;
string manufacturer,color,author,name;

cout<<"\n**************WELCOME TO ONLINE SHOP******************\n";
cout<<"\nPress 1 for Electronics."<<"\nPress 2 for Books\n";
cin>>choice1;
if (choice1==1){
      cout<<"\nPress 1 for MP3Players."<<"\nPress 2 for TV\n";
      cin>>choice2;
      if (choice2==1)  {
            cout<<"\nSet Attributes\n";
            cout<<"Price: ";
            cin>>price;
            cout<<"\nDiscount in %: ";
            cin>>discount;
            cout<<"\nManufacturer: ";
            cin>>manufacturer;
            cout<<"\nColor: ";
            cin>>color;
            cout<<"\nCapacity in GB: ";
            cin>>capacity;
            MP3player m1(price,discount,manufacturer,color,capacity);
            if (m1.getCapacity()==1)
                m1.setDiscount(10);
            else
                m1.setDiscount(50);
            cout<<m1.SalePrice();

        }
        elseif (choice2==2)   {
            cout<<"\nSet Attributes\n";
            cout<<"Price: ";
            cin>>price;
            cout<<"\nDiscount in %: ";
            cin>>discount;
            cout<<"\nManufacturer: ";
            cin>>manufacturer;
            cout<<"\nSize: ";
            cin>>size;
            TV t1(price,discount,manufacturer,size);
            cout<<t1.SalePrice();
        }
        else
            cout<<"\nInvalid value... try again later";
    }
    elseif (choice1==2){
        cout<<"\nSet Attributes\n";
        cout<<"Price: ";
        cin>>price;
        cout<<"\nDiscount in %: ";
        cin>>discount;
        cout<<"\nAuthor: ";
        cin>>author;
        cout<<"\nName: ";
        cin>>name;
        Book b1(price, discount, author,name);
        cout<<b1.SalePrice();
    }
    else
        cout<<"\nInvalid value... try again later";
    }
```

Post Lab:

a. A point in the x-y plane is represented by its x-coordinate and y-coordinate. Design a class, "*pointType*", that can store and process a point in the x-y plane. You should then perform operations on the point, such as setting the coordinates of the point, printing the coordinates of the point, returning thex-coordinate, and returning the y-coordinate. Also, write a program to testvarious operations on the point.

b. Every circle has a center and a radius. Given the radius, we can determine the circle's area and circumference. Given the center, we can determine its position in the x-y plane. The center of the circle is a point in the x-y plane. Design a class, "*circleType*", which can store the radius and center of the circle. Because the center is a point in the x-y plane and you designed the class to capture the properties of a point in Part a), you must derive the class "*circleType*" from the class "*pointType*". You should be able to perform the usual operations on the circle, such as setting the radius, printing the radius, calculating and printing the area and circumference, and carrying out the usual operations on the center. Also, write a program to test various operations on a circle.

c. Every cylinder has a base and height, wherein the base is a circle. Design a class, "*cylinderType*" that can capture the properties of a cylinder and perform the usual operations on the cylinder. Derive this class from the class "*circleType*" designed in Part b). Some of the operations that can be performed on a cylinder are as follows: calculate and print the volume, calculate and print the surface area, set the height, set the radius of the base, and set the center of the base. Also, write a program to test various operations on a cylinder.