### **Activity 01**

Define an abstract base class **shape** that includes protected data members for area and volume of a shape, public methods for computing area and volume of a shape (make the functions **virtual**), and a display function to display the information about an object. Make this class abstract.

Derive a concrete class **point** from the **shape** class. This **point** class contains two protected data members that hold the position of **point**. Provide no-argument and 2-argument constructors. Override the appropriate functions of base class.

Derive a class **Circle** publicly from the **point** class. This class has a protected data member of **radius**. Provide a no-argument constructor to initialize the fields to some fixed values. Provide a 3-argument constructor to initialize the data members of **Circle** class to the values sent from outside. Override the methods of base class as required.

Derive another class **Cylinder** from the **Circle** class. Provide a protected data member for height of cylinder. Provide a no-argument constructor for initializing the data members to default values. Provide a 4-argument constructor to initialize x- and y-coordinates, radius, and height of cylinder. Override the methods of base class.

Write a driver program to check the polymorphic behavior of this class.

### **Activity 02**

Let us create the shape class hierarchy. Create a **Shape** class which must be abstract since we are not going to create any instance of this class. Provide data members for storing area and volume in this class. Provide a virtual function for displaying the data members of this class, a virtual function for computing volume of a shape, and a virtual function for computing area of a shape.

Derive two abstract classes **TwoDimensional** and **ThreeDimensional** from this class. Provide a virtual member function to display whether an object is 2-dimensional or 3-dimensional. Also include virtual area and volume functions so these calculations can be performed for objects of each concrete class in hierarchy.

Implement the following hierarchy and write a driver program to test the polymorphic behavior of this class.