Abstraction

1. Create an abstract class 'Parent' with a method 'message'. It has two subclasses each having a method with the same name 'message' that prints "This is first subclass" and "This is second subclass" respectively. Call the methods 'message' by creating an object for each subclass.
2. Create an abstract class 'Bank' with an abstract method 'getBalance'. $100, $150 and $200 are deposited in banks A, B and C respectively. 'BankA', 'BankB' and 'BankC' are subclasses of class 'Bank', each having a method named 'getBalance'. Call this method by creating an object of each of the three classes.
3. We have to calculate the percentage of marks obtained in three subjects (each out of 100) by student A and in four subjects (each out of 100) by student B. Create an abstract class 'Marks' with an abstract method 'getPercentage'. It is inherited by two other classes 'A' and 'B' each having a method with the same name which returns the percentage of the students. The constructor of student A takes the marks in three subjects as its parameters and the marks in four subjects as its parameters for student B. Create an object for eac of the two classes and print the percentage of marks for both the students.
4. An abstract class has a construtor which prints "This is constructor of abstract class", an abstract method named 'a_method' and a non-abstract method which prints "This is a normal method of abstract class". A class 'SubClass' inherits the abstract class and has a method named 'a_method' which prints "This is abstract method". Now create an object of 'SubClass' and call the abstract method and the non-abstract method. (Analyse the result)
5. Create an abstract class 'Animals' with two abstract methods 'cats' and 'dogs'. Now create a class 'Cats' with a method 'cats' which prints "Cats meow" and a class 'Dogs' with a method 'dogs' which prints "Dogs bark", both inheriting the class 'Animals'. Now create an object for each of the subclasses and call their respective methods.
6. We have to calculate the area of a rectangle, a square and a circle. Create an abstract class 'Shape' with three abstract methods namely 'RectangleArea' taking two parameters, 'SquareArea' and 'CircleArea' taking one parameter each. The parameters of 'RectangleArea' are its length and breadth, that of 'SquareArea' is its side and that of 'CircleArea' is its radius. Now create another class 'Area' containing all the three methods 'RectangleArea', 'SquareArea' and 'CircleArea' for printing the area of rectangle, square and circle respectively. Create an object of class 'Area' and call all the three methods.
7. Repeat the above question for 4 rectangles, 4 squares and 5 circles. Hint- Use array of objects.
8. Create an interface TVremote and use it to inherit another interface smart TVremote. Create a class TV which implements TVremote interface.

Polymorphism

1. Write a console based program to implement polymorphism using inheritance. Consider the example of Shape as base class with method show(). And then a child class Circle and Rectangle which inherit the base class Shape and override its method show(). Add one more Method with the name of getInfo(). This method would display the class name in which it is implemented. Do not override this method. When you will call the method getInfo() with child object it would still show the name of the base class, which implies that method has been directly inherited and was not overridden.

2. Write a subclass called SubClass that is derived from SuperClass and that adds an integer data field called data2 and a public method called checkCondition() that will check if data1 is equal to 10 and data2 is equal to 15, the checkCondition () method should return "Condition True!". Also, create methods called setData2() and getData2() for setting and retrieving the value of data1 and data2, as well as a constructor that accepts arguments for the starting values of data1 and data2. data1 is data member of SuperClass.

3. Create a class named Pizza that stores information about a single pizza. It should contain the following:

◎ Private instance variables to store the size of the pizza (either small, medium, or large), the number of cheese toppings, the number of pepperoni toppings, and the number of ham toppings
◎ Constructor(s) that set all of the instance variables.
◎ Public methods to get and set the instance variables.
◎ A public method named calcCost( ) that returns a double that is the cost of the pizza.
Pizza cost is determined by:

Small: $10 + $2 per topping
Medium: $12 + $2 per topping
Large: $14 + $2 per topping

◎ public method named getDescription( ) that returns a String containing the pizza size, quantity of each topping.

Write test code to create several pizzas and output their descriptions. For example, a large pizza with one cheese, one pepperoni and two ham toppings should cost a total of $22. Now Create a PizzaOrder class that allows up to three pizzas to be saved in an order. Each pizza saved should be a Pizza object. Create a method calcTotal() that returns the cost of order. In the runner order two pizzas and return the total cost.