Problem Set

1. Create a base class Animal with a method sound() that prints "This is an animal sound." Create a derived class Dog that extends Animal and overrides the sound() method to print "Bark". Test the functionality by creating an object of Dog in the main() method and calling the sound() method.
2. Define a base class Person with a constructor that initializes the name attribute. Create a subclass Student that inherits from Person and adds an attribute rollNo. Call the base class constructor from the subclass and print both the name and rollNo.
3. Create a class Vehicle with a method move() that prints "Vehicle is moving." Create a subclass Car that overrides the move() method to print "Car is moving fast." Demonstrate method overriding by creating objects of both Vehicle and Car classes and calling move().
4. Create a class Parent with a method showMessage() that prints "This is the parent class." Create a subclass Child that overrides the showMessage() method but also calls the parent class method using super keyword.
5. Create a class Person with a protected attribute name. Create a subclass Employee that inherits from Person and sets the value of name from within the subclass. Demonstrate the access of protected members from the subclass.
6. Create a base class Appliance with a method turnOn(). Then create a class WashingMachine that inherits from Appliance and adds a method startWash(). Finally, create a subclass AutomaticWashingMachine that inherits from WashingMachine and adds a method autoStart(). Demonstrate the multiple levels of inheritance in the main() method.
7. Create a base class Vehicle with an attribute maxSpeed and a constructor that sets it. Create a subclass Car that uses the super keyword to call the parent constructor to initialize maxSpeed. Add an additional attribute brand to Car and demonstrate constructor chaining.
8. Define a base class Shape with a method draw(). Create two subclasses Circle and Square that inherit from Shape and override the draw() method to print different messages. Demonstrate the concept of hierarchical inheritance by calling the draw() method on objects of Circle and Square.
9. Create a base class Animal and a subclass Cat. In the main() method, create an object of Cat and upcast it to the Animal class. Then downcast it back to Cat and call a method that only exists in the Cat class.
10. Create a class Employee with a method work(). Create two subclasses Manager and Developer that override the work() method. In the main() method, create an array of Employee objects that stores both Manager and Developer objects and call the work() method for each.

11. Create an abstract class Employee with an abstract method calculateSalary(). Then create two subclasses FullTimeEmployee and PartTimeEmployee that provide implementations for calculateSalary(). Demonstrate polymorphism by calling the method on objects of both subclasses.
12. Create a class Person with a method getDetails() that returns the person's name. Create a subclass Employee that overrides getDetails() to return both the name and the employee ID. Use the super keyword in Employee to call getDetails() of the Person class and combine the result.
13. Create a class Vehicle with a final method start(). Create a subclass Car that tries to override the start() method and observe the error. Also, try to make the Vehicle class final and observe how it affects the inheritance hierarchy.
14. Create two interfaces Teacher and Researcher, both containing a method work(). Create a class Professor that implements both interfaces and provides its own implementation of work(). Call the work() method from a Professor object to demonstrate multiple inheritance.
15. Create two interfaces A and B, both with a method display(). Create a class C that implements both interfaces. Handle the diamond problem by providing an implementation for the display() method in C. Demonstrate this in the main() method.