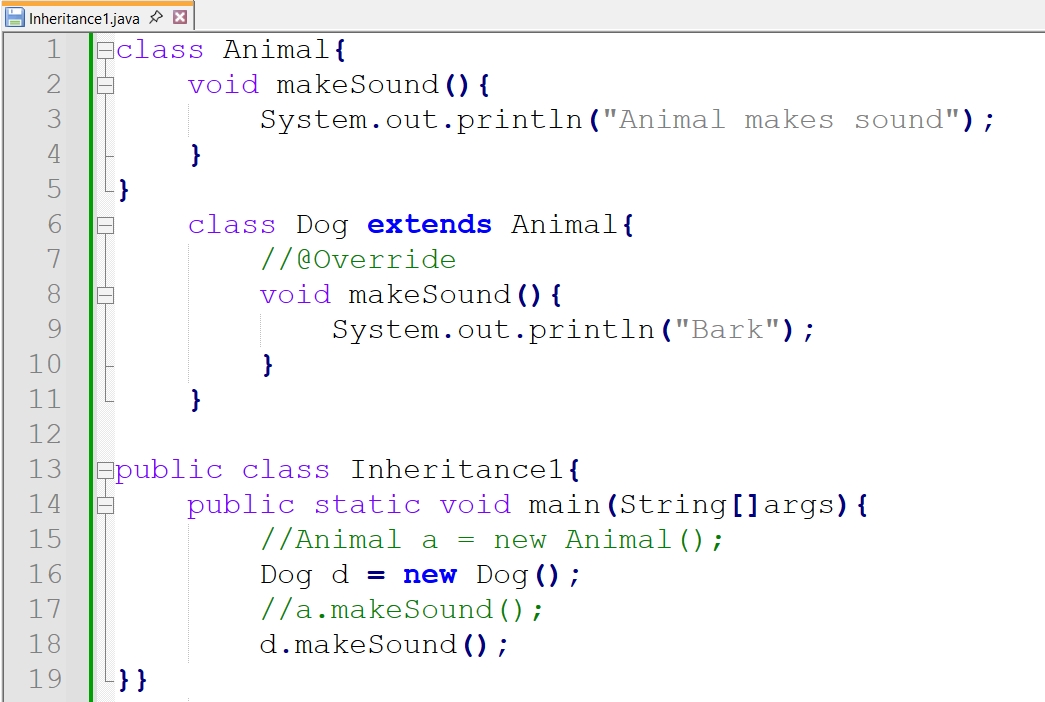
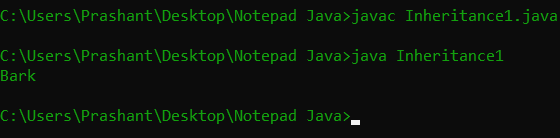
**Assignment No. 2**

**OOPS 4 Pillar Based**

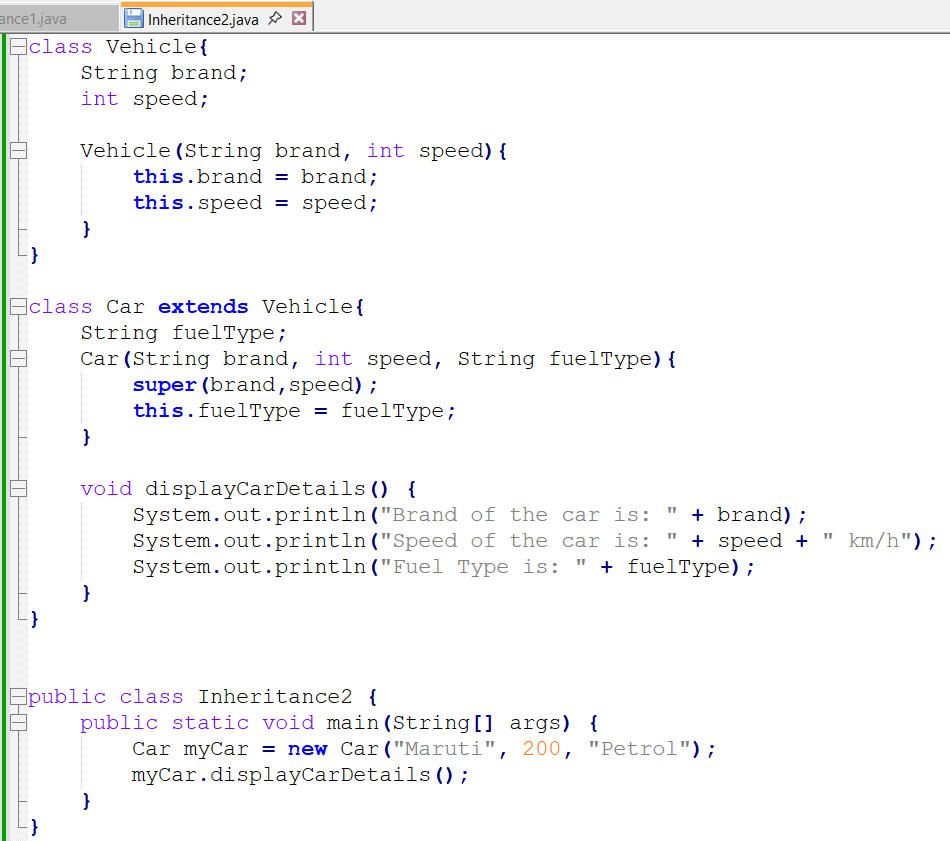
**1. Inheritance (5 Questions)**

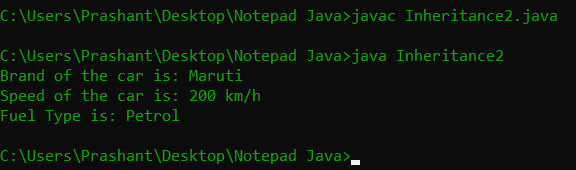
1. **Create a base class Animal** with a method makeSound(). Create a subclass Dog that overrides makeSound() to print "Bark".



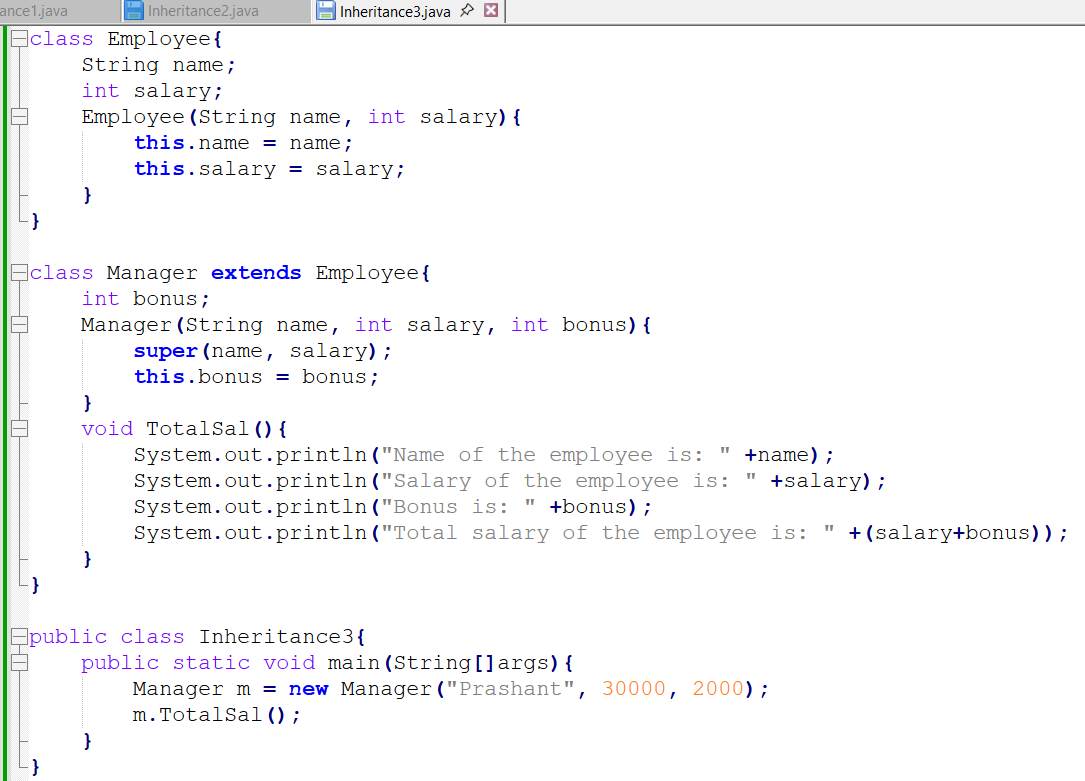


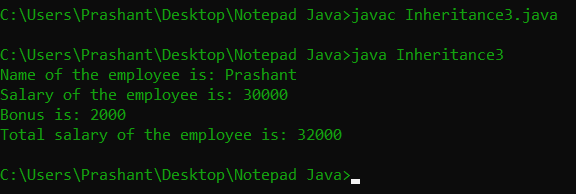
2. **Create a base class Vehicle** with properties brand and speed. Create a subclass Car that adds fuelType and a method displayCarDetails().



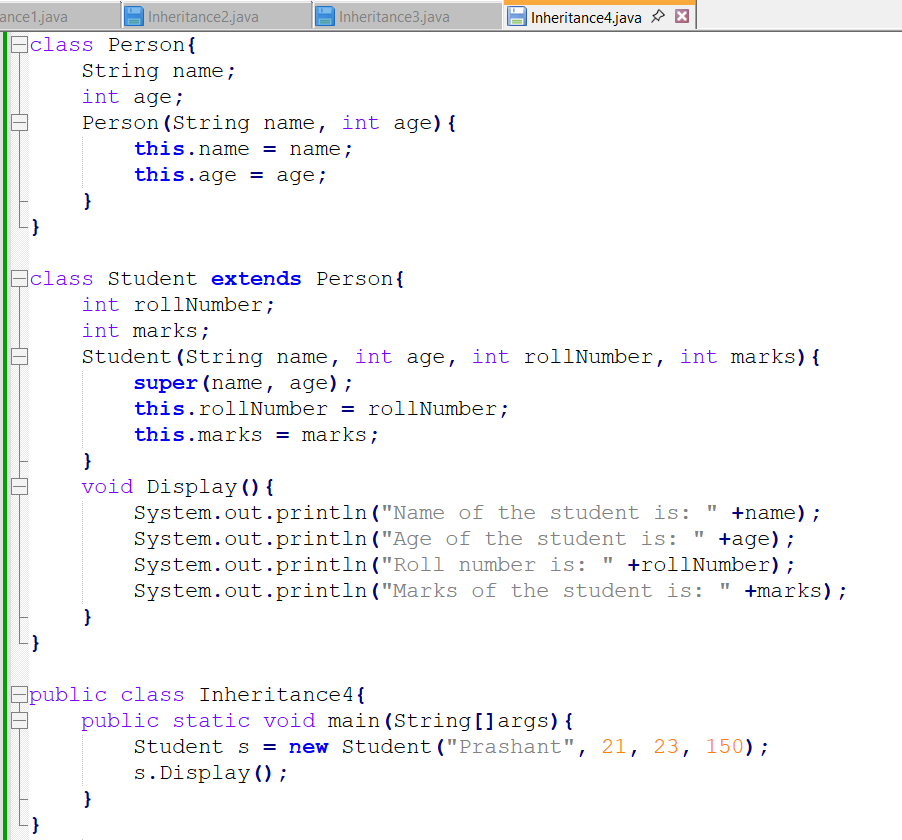


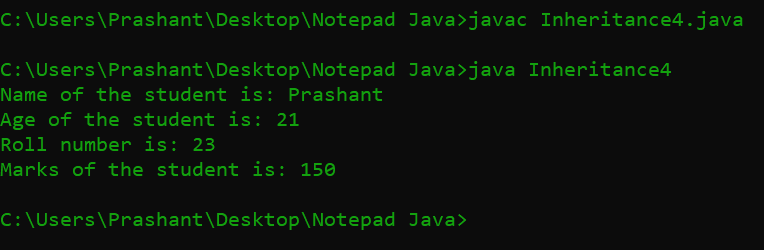
3. **Create a base class Employee** with attributes name and salary. Create a subclass Manager that adds bonus. Write a method to calculate the total salary.



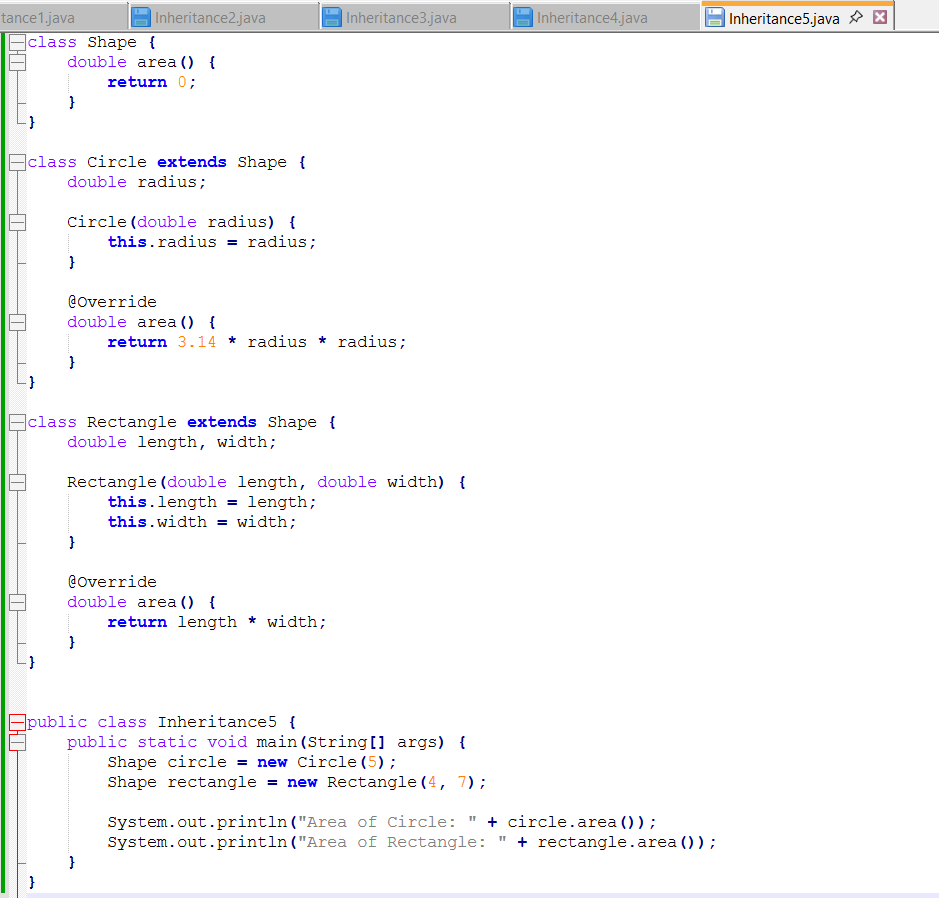


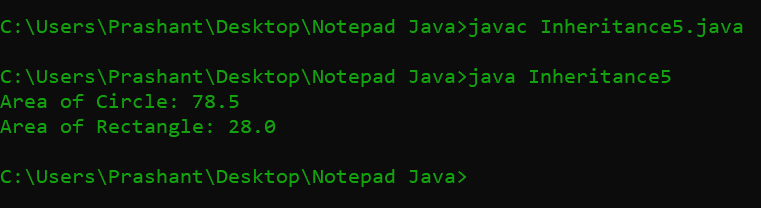
4. **Write a Java program** where a Person class has name and age. Create a subclass Student that adds rollNumber and marks.





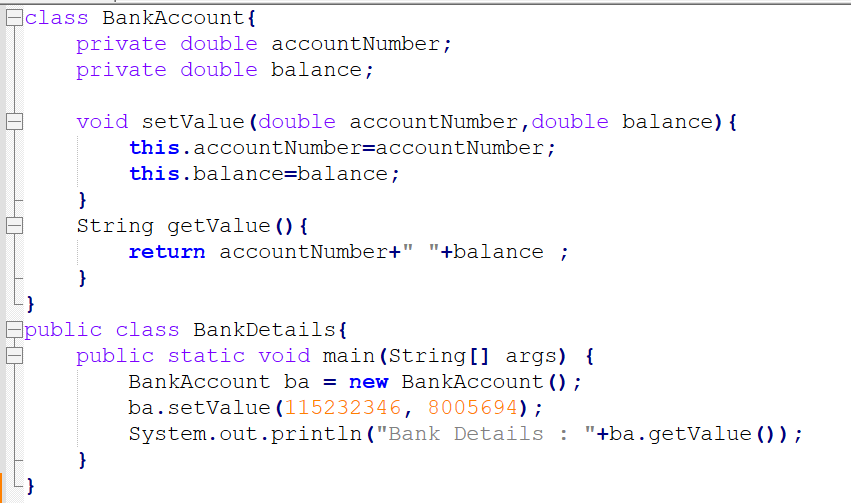
5. **Create a base class Shape** with a method area(). Create subclasses Circle and Rectangle that override area() to calculate their respective areas.





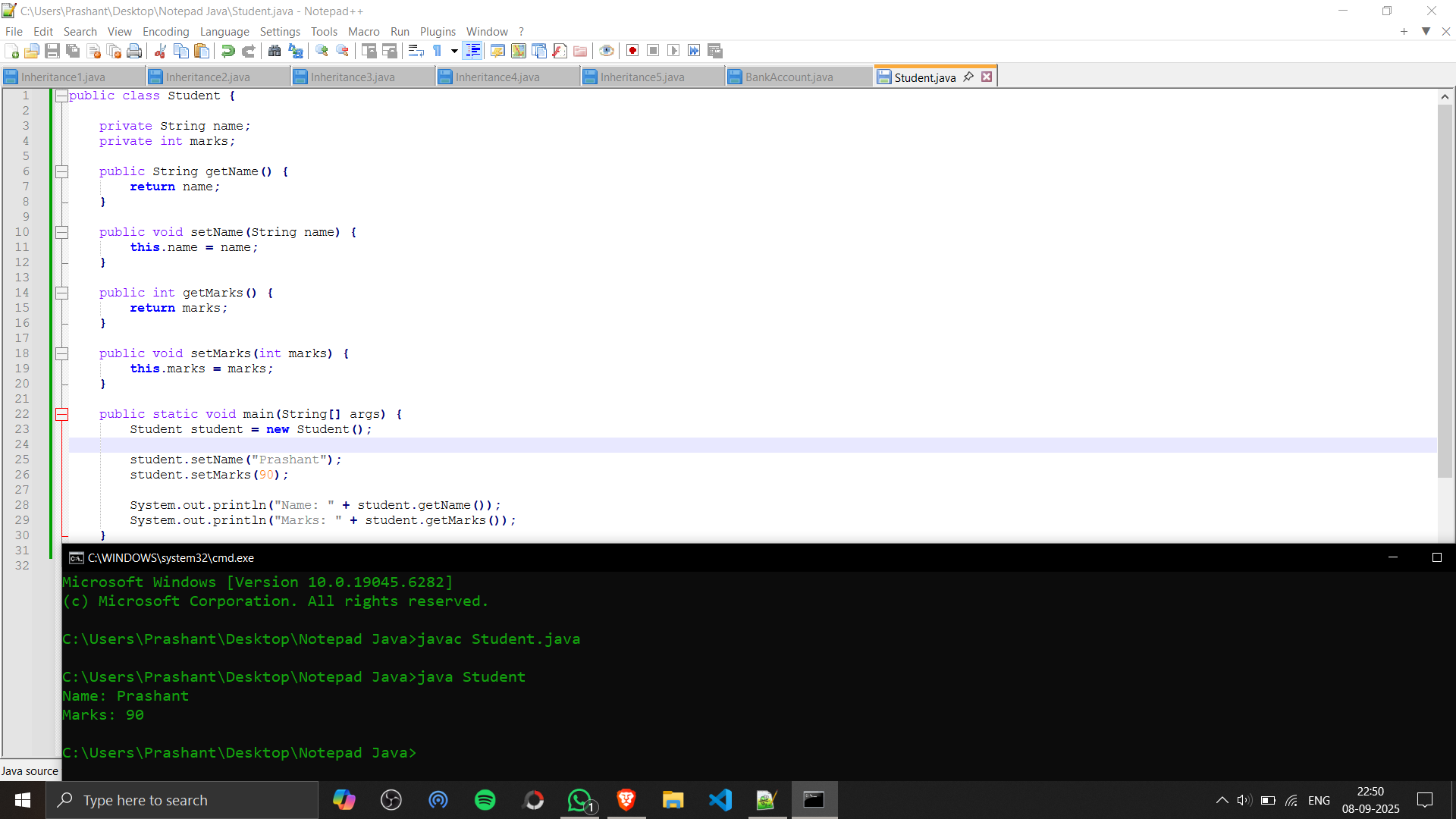
**2. Encapsulation (5 Questions)**

6. **Create a class BankAccount** with private attributes accountNumber and balance. Use getters and setters to access and modify them.

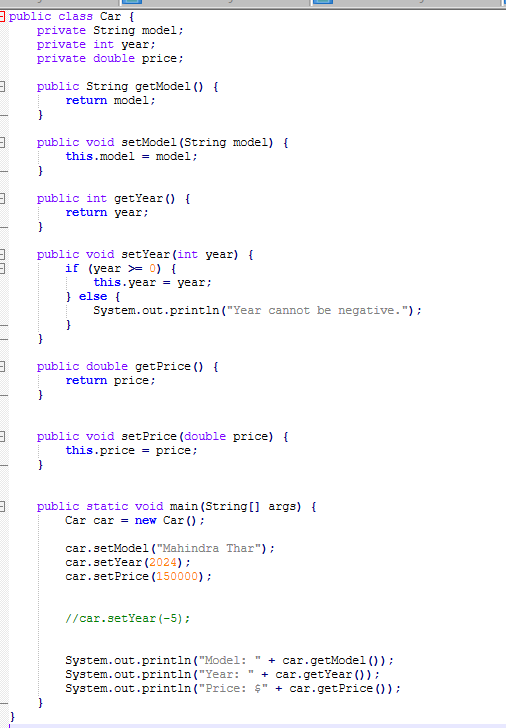


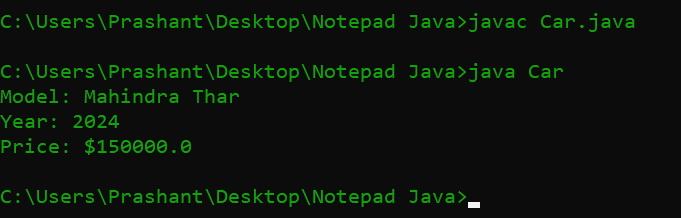


7. **Write a Java program** to create a Student class with private variables name and marks. Use getters to retrieve and setters to modify the values.

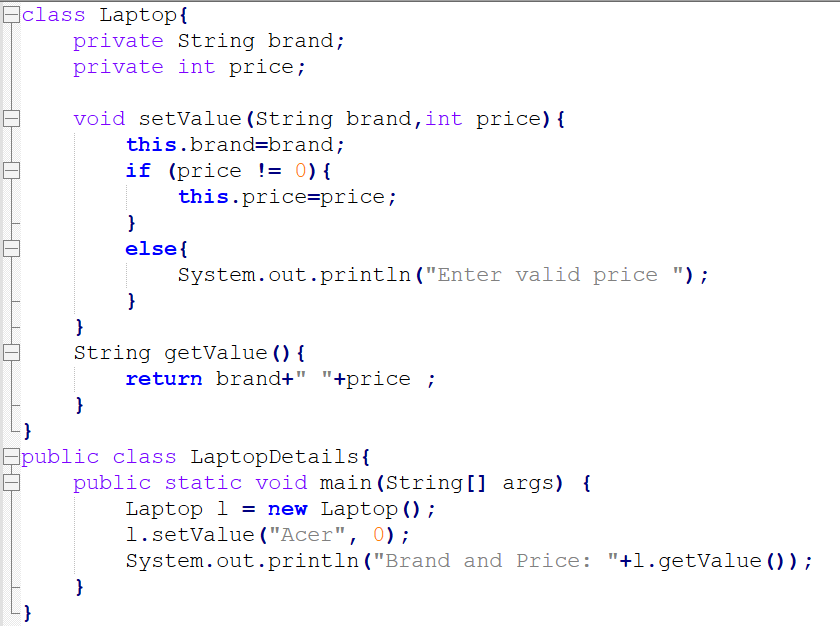


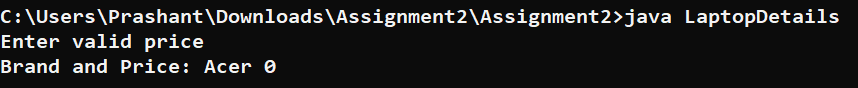
8. **Create a class Car** with private variables model, year, and price. Provide public methods to get and set values while ensuring year is not negative.



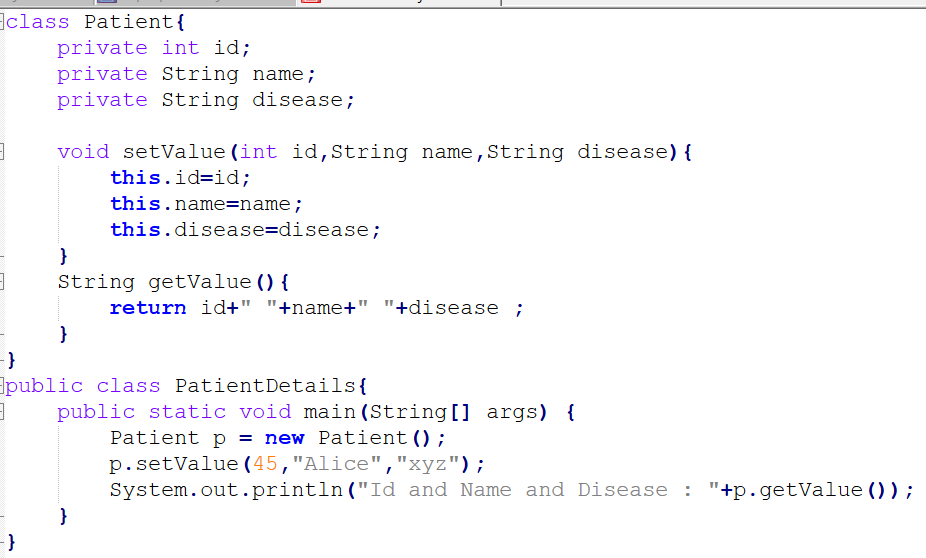


9. **Write a Java program** for a Laptop class with private attributes brand and price. Ensure price cannot be set below zero using validation inside the setter method.





10. **Create a Patient class** with private attributes id, name, and disease. Provide methods to set and get details and restrict modification of id once assigned.

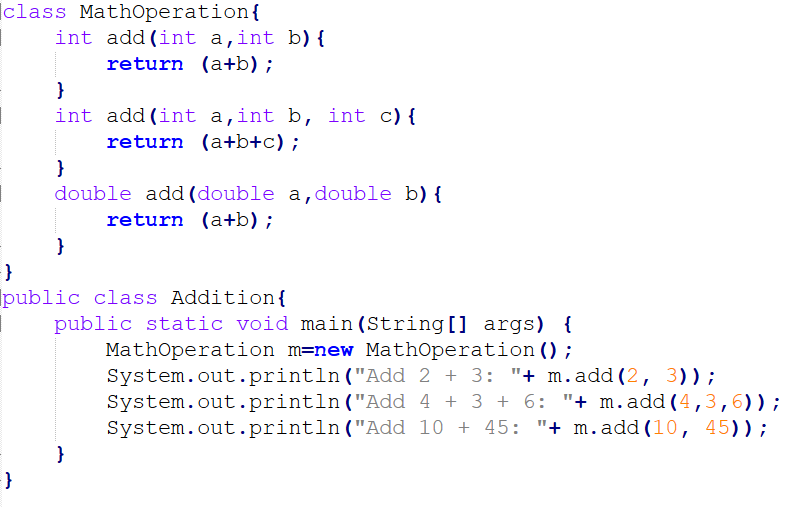


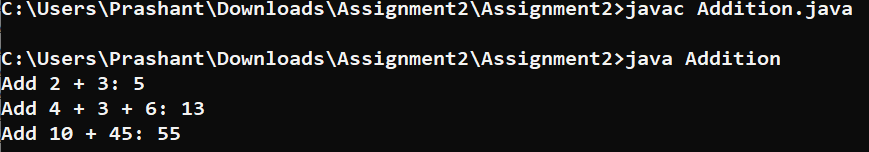


**3. Polymorphism (5 Questions)**

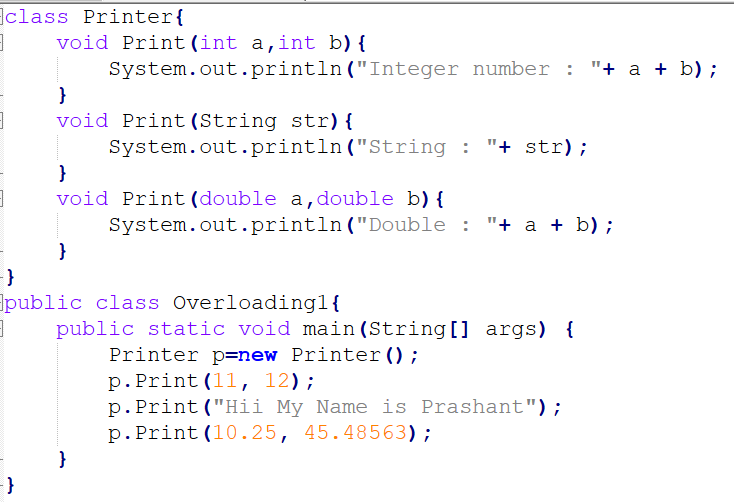
**(A) Compile-Time Polymorphism (Method Overloading)**

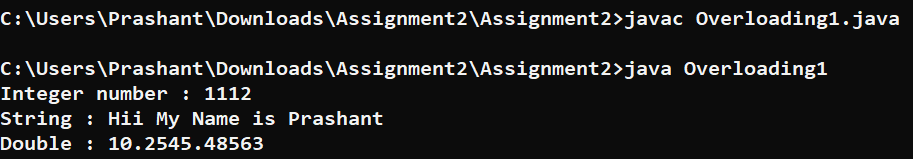
11. **Create a MathOperations class** with overloaded add() methods: one for two integers, another for three integers, and one for two double values.



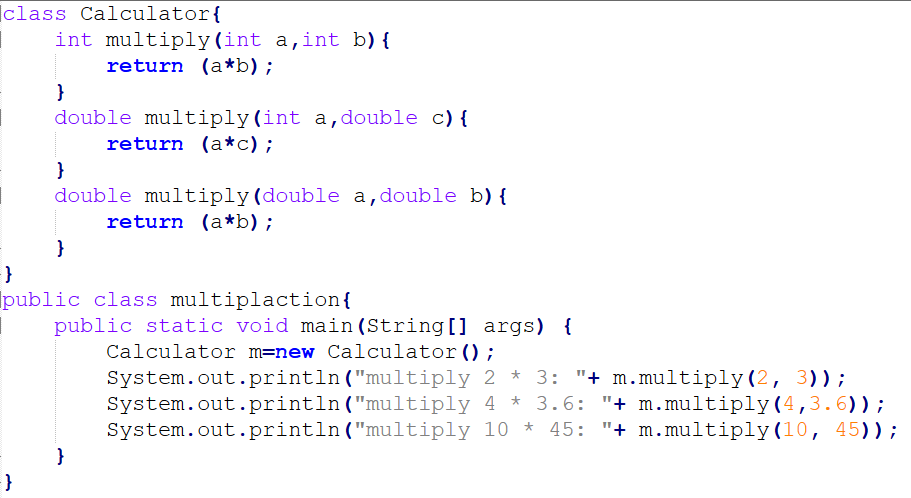


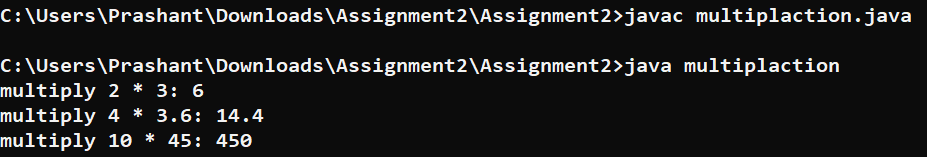
12. **Write a Java program** to create a class Printer that has multiple overloaded print()methods for String, int, and double values.



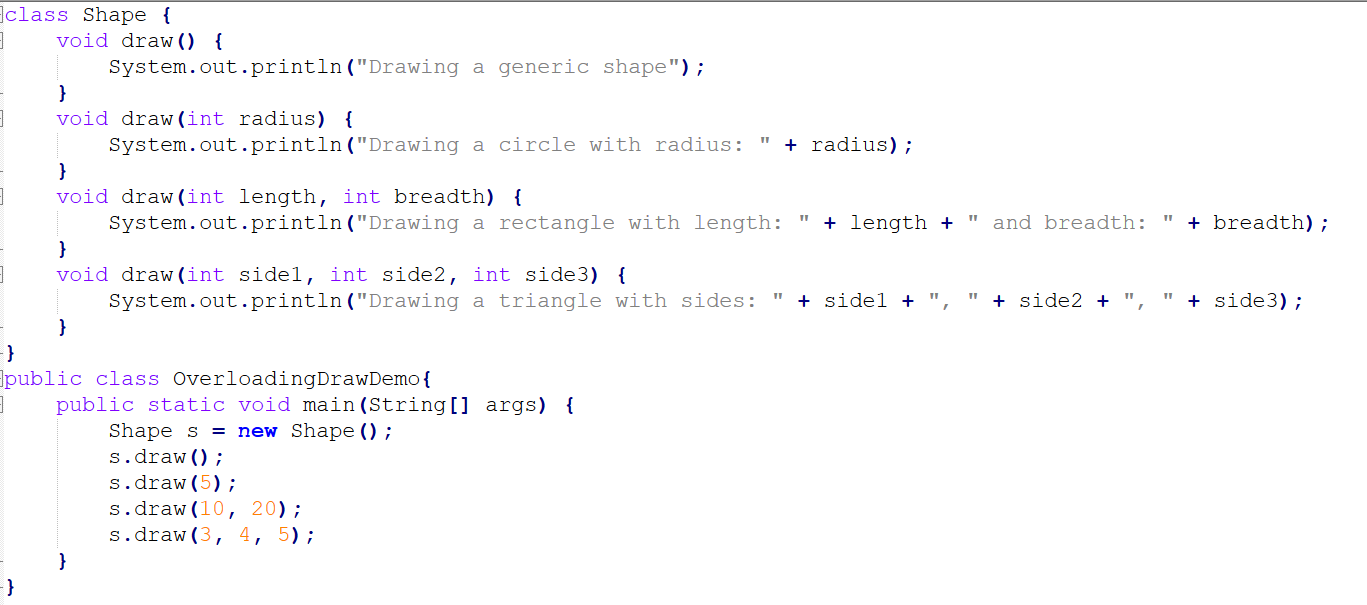


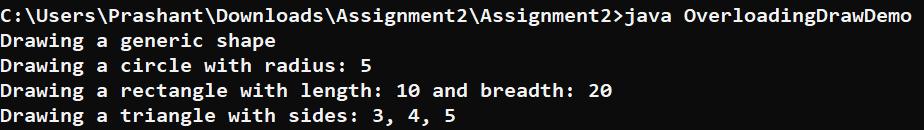
13. **Create a Calculator class** with overloaded multiply() methods to accept integers, doubles, and a mix of both.



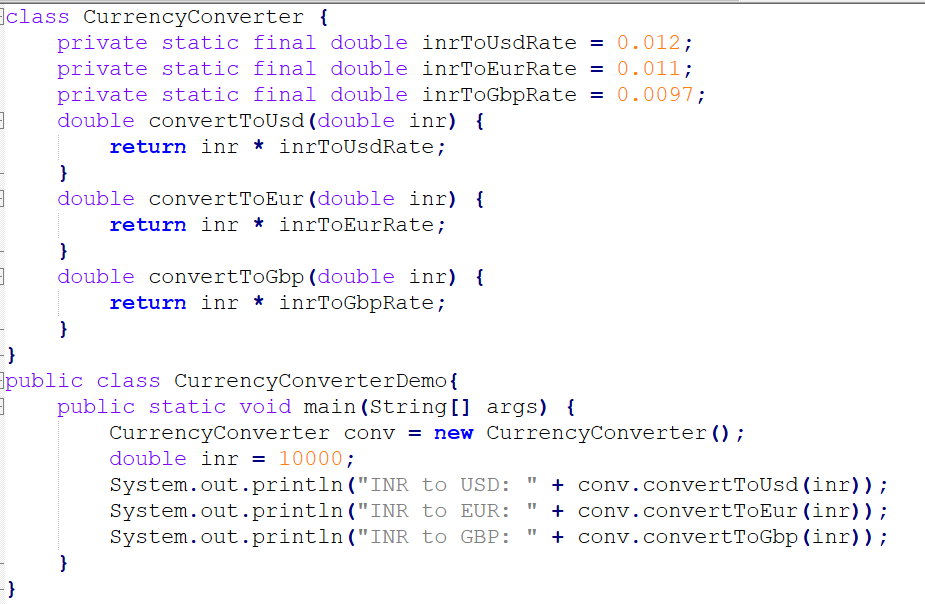


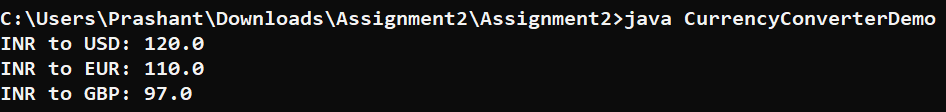
14. **Write a Java program** where a Shape class has overloaded draw() methods, accepting different numbers of parameters to draw different shapes.





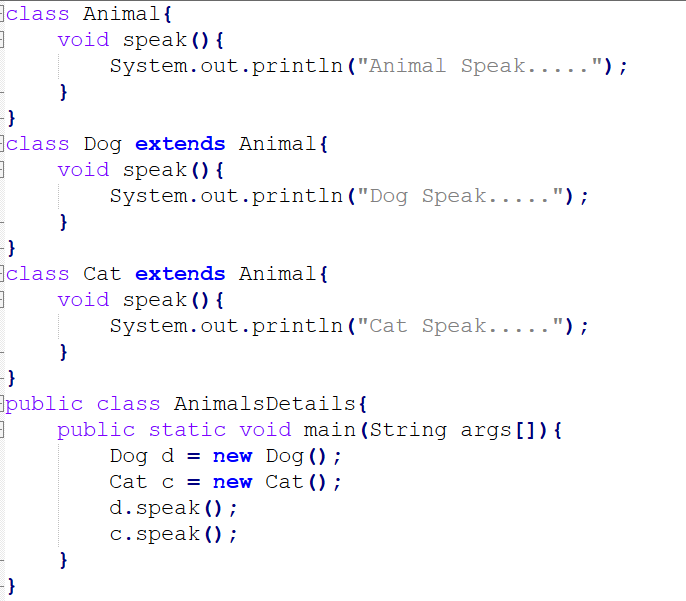
15. **Create a class CurrencyConverter** that has overloaded methods to convert different currencies (INR to USD, INR to EUR, etc.).

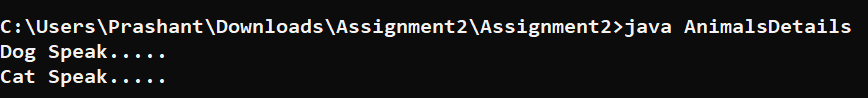




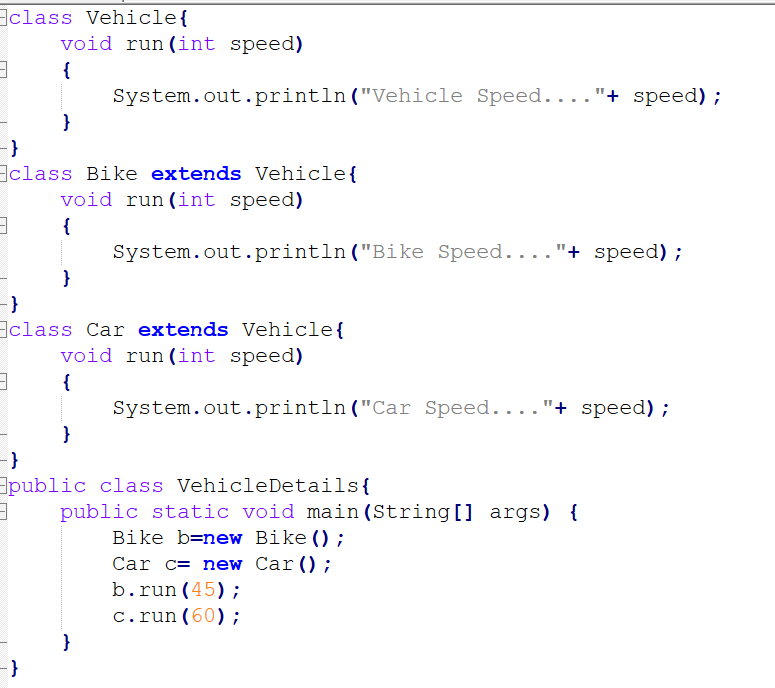
**(B) Runtime Polymorphism (Method Overriding)**

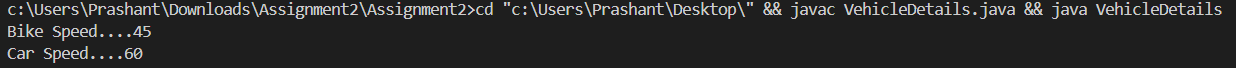
16. **Create a base class Animal** with speak() method. Create subclasses Dog and Cat that override speak() to print different sounds.



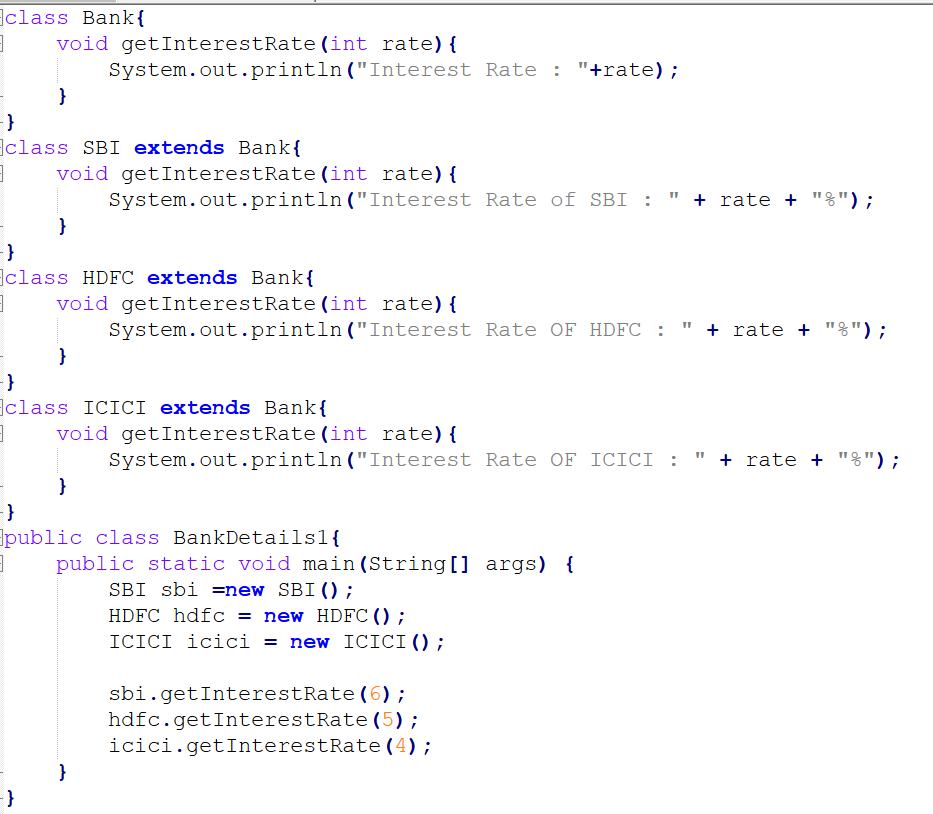


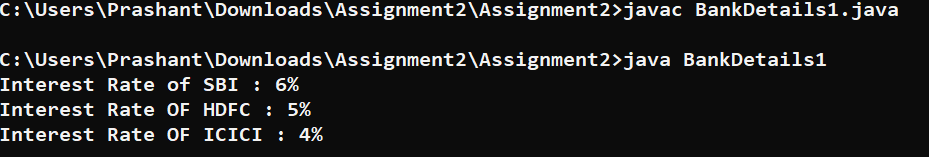
17. **Write a Java program** where a Vehicle class has a run() method. Create subclasses Bike and Car that override run() with specific messages.



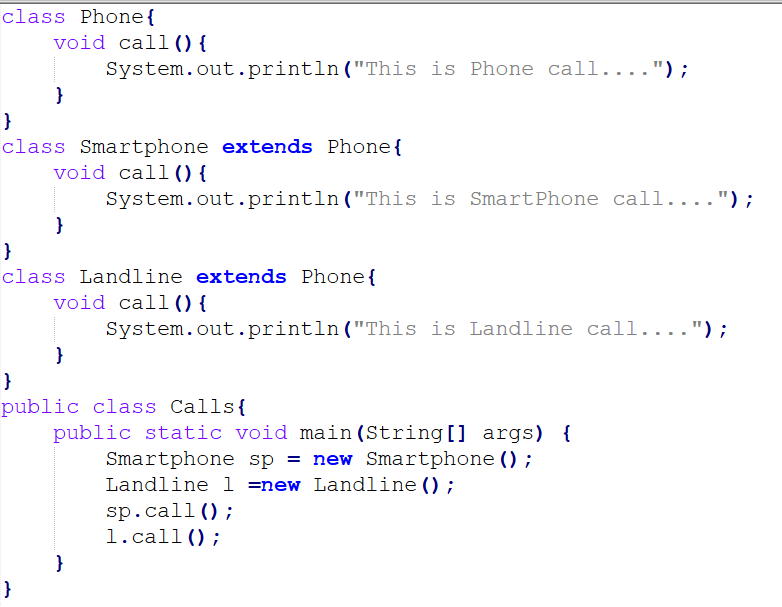


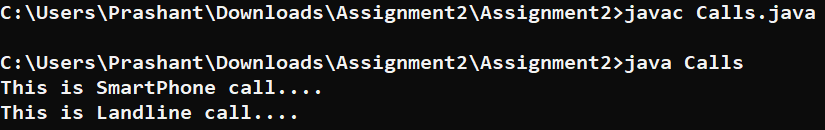
18. **Create a Bank class** with a method getInterestRate(). Create subclasses SBI, HDFC, and ICICI that override the method with their respective interest rates.



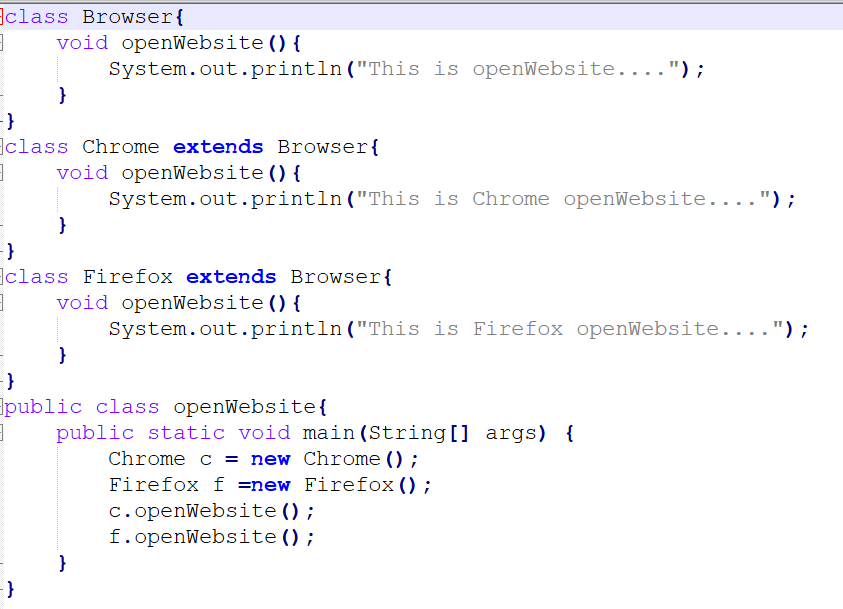


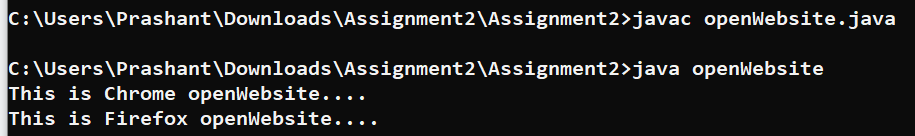
19. **Write a Java program** where a Phone class has a method call(). Create subclasses Smartphone and Landline that override call() differently.





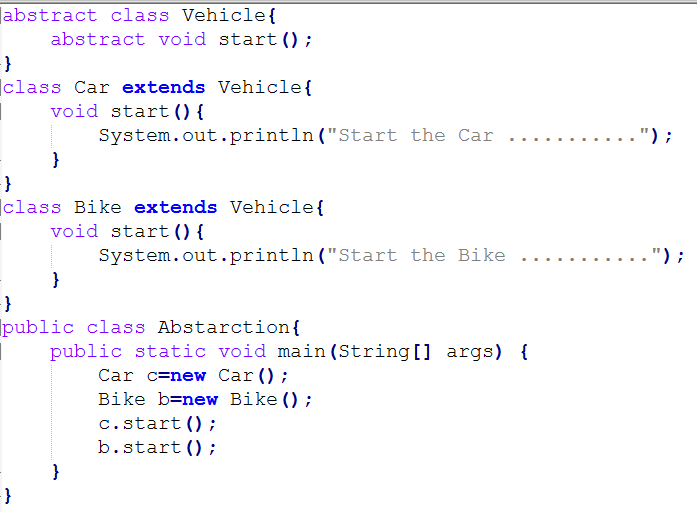
20. **Create a Browser class** with a method openWebsite(). Create subclasses Chrome and Firefox that override openWebsite() with specific implementation details.

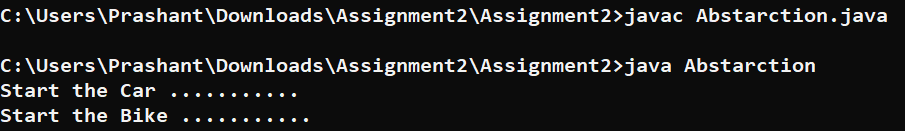




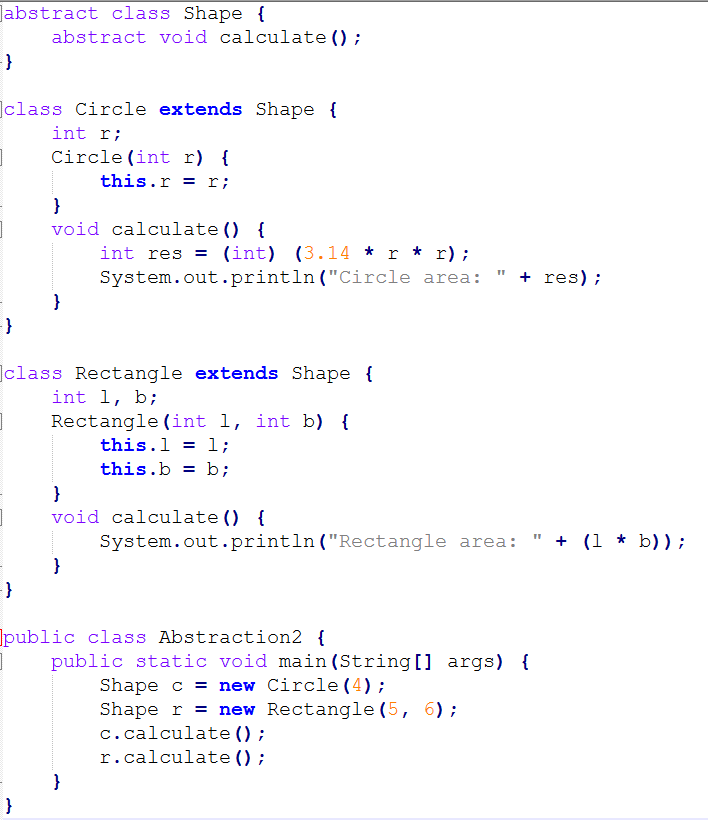
**4. Abstraction (5 Questions)**

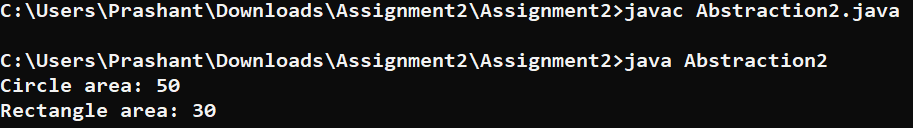
21. **Create an abstract class Vehicle** with an abstract method start(). Create subclasses Car and Bike that provide their own implementation of start().



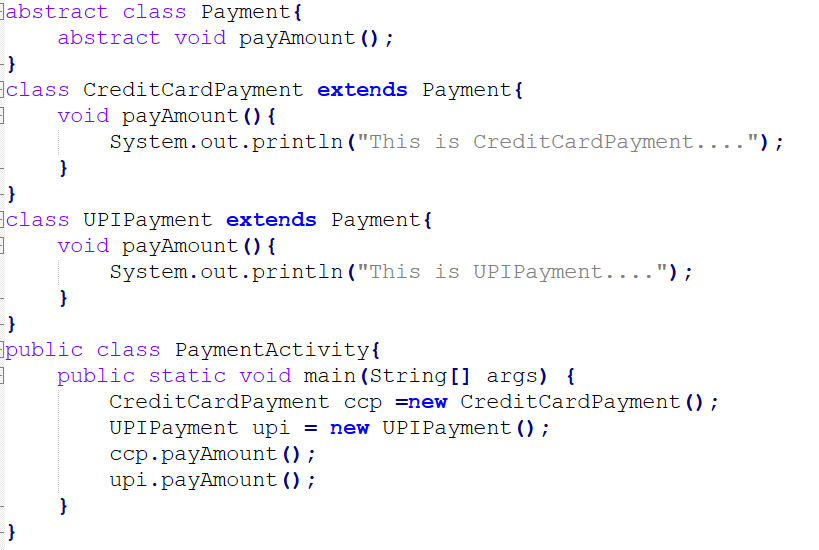


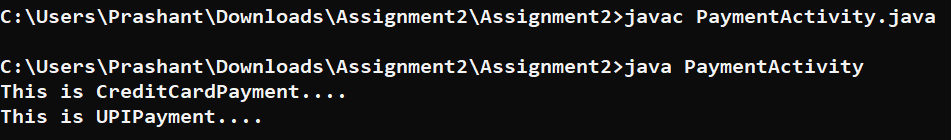
22. **Write a Java program** with an abstract class Shape that has an abstract method calculateArea(). Implement it in Circle and Rectangle classes.



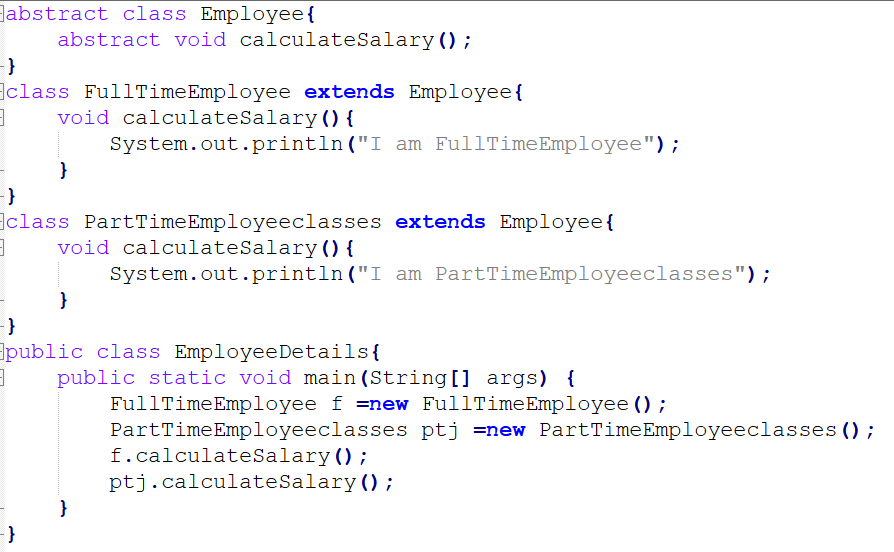


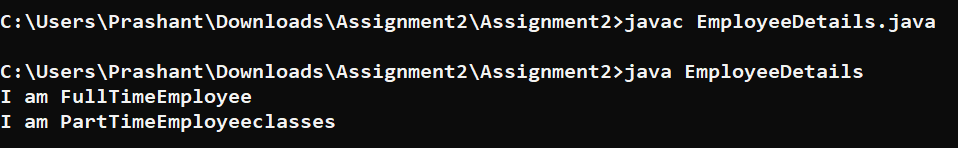
23. **Create an abstract class Payment** with an abstract method payAmount(). Create subclasses CreditCardPayment and UPIPayment that implement it differently.





24. **Write a Java program** with an abstract class Employee that has an abstract method calculateSalary(). Implement it in FullTimeEmployee and PartTimeEmployeeclasses.





25. **Create an abstract class Appliance** with abstract methods turnOn() and turnOff(). Implement these in Fan and Light classes.

