***1.Encapsulation + Getter/Setter***

WAP in Java

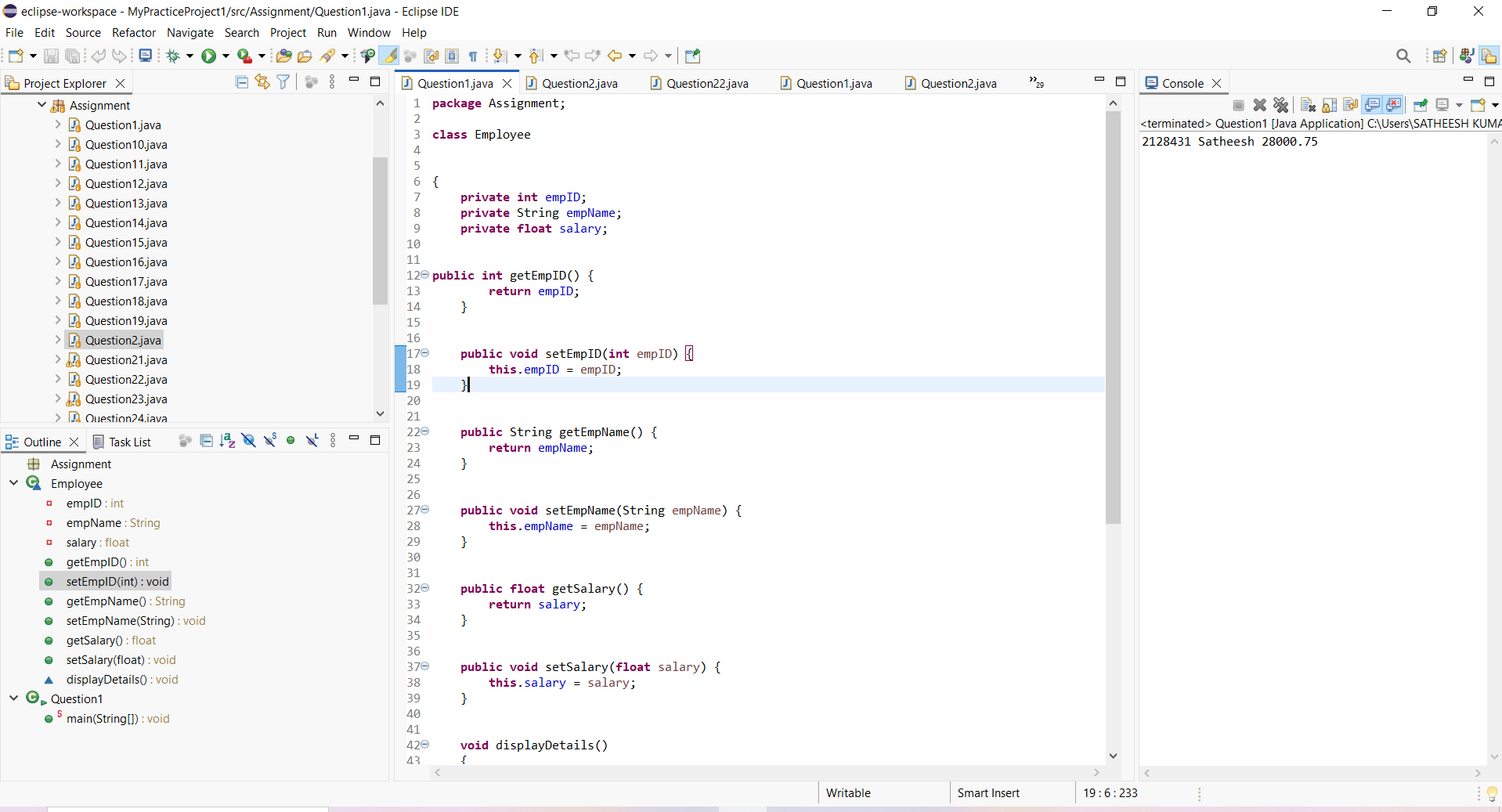
Create a class named Employee with private instance variables empId, empName, and salary.

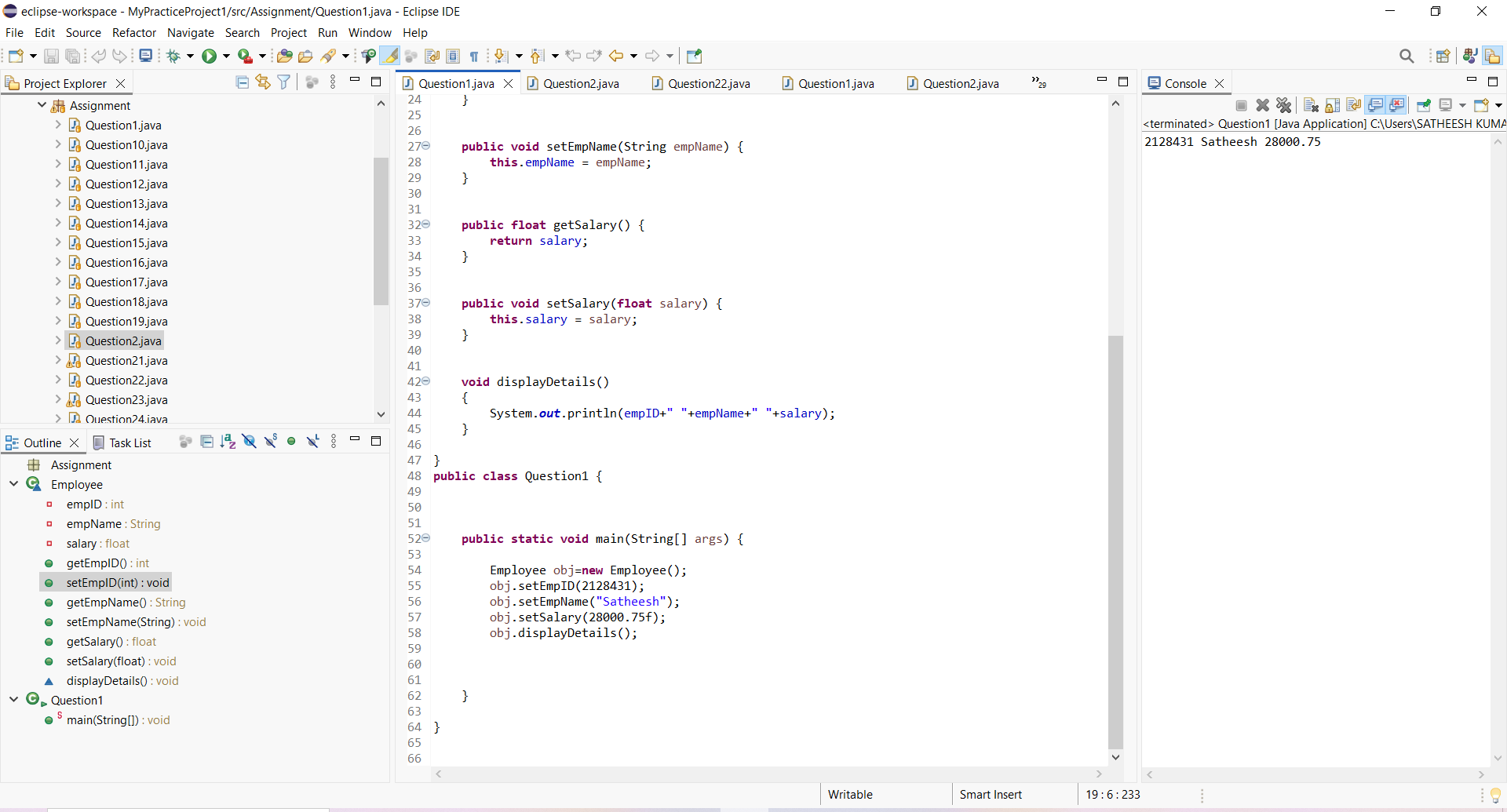
Provide public getters and setters for all variables.

Create a method displayDetails() to print employee details.

Create an object in the main method and assign values using setters then display them.

***Program and output:***





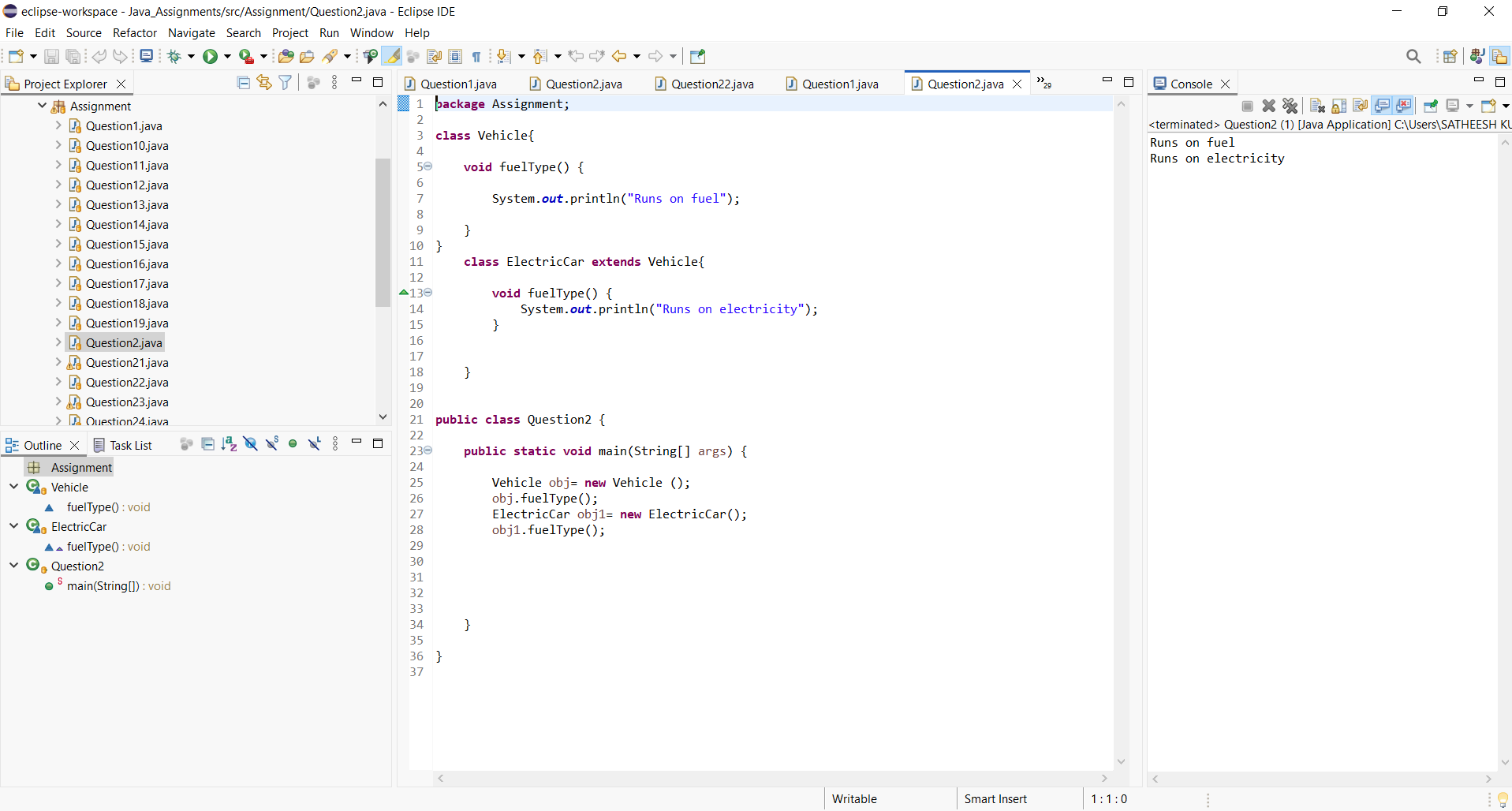
***2. Inheritance + Method Overriding***

Create a base class Vehicle with a method fuelType() which prints "Runs on fuel".

Create a child class ElectricCar and override the fuelType() method to print "Runs on electricity".

Create objects of both classes and call their respective methods.

***Program and output:***



***3. Constructor Overloading***

Create a class Product having instance variables productId, productName, and price.

Implement:

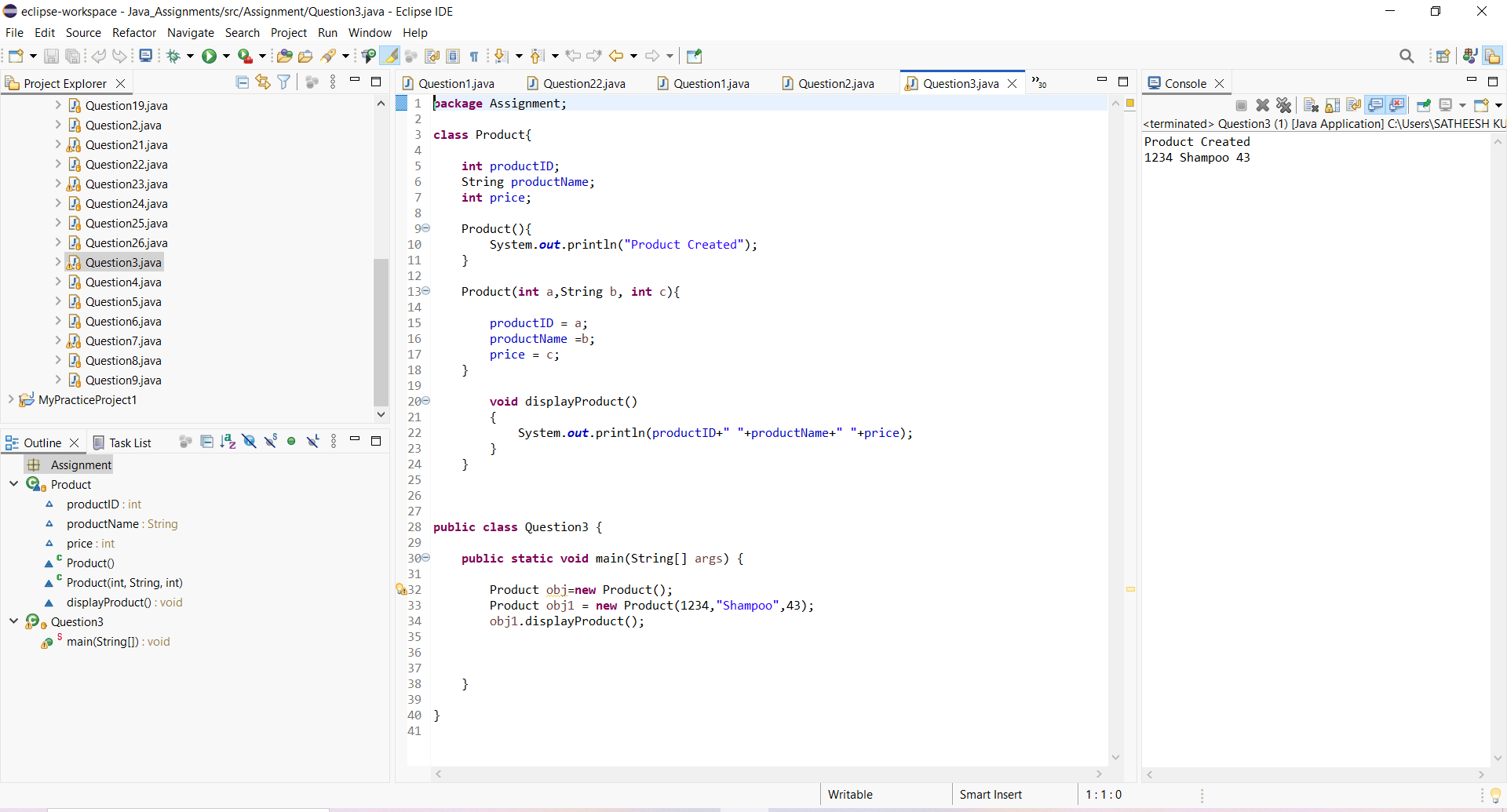
A default constructor that prints "Product Created".

A parameterized constructor that initializes the product details.

Write a method displayProduct() to print product details.

Create both types of objects in the main method.

***Program and Output:***



***4. Method Overloading***

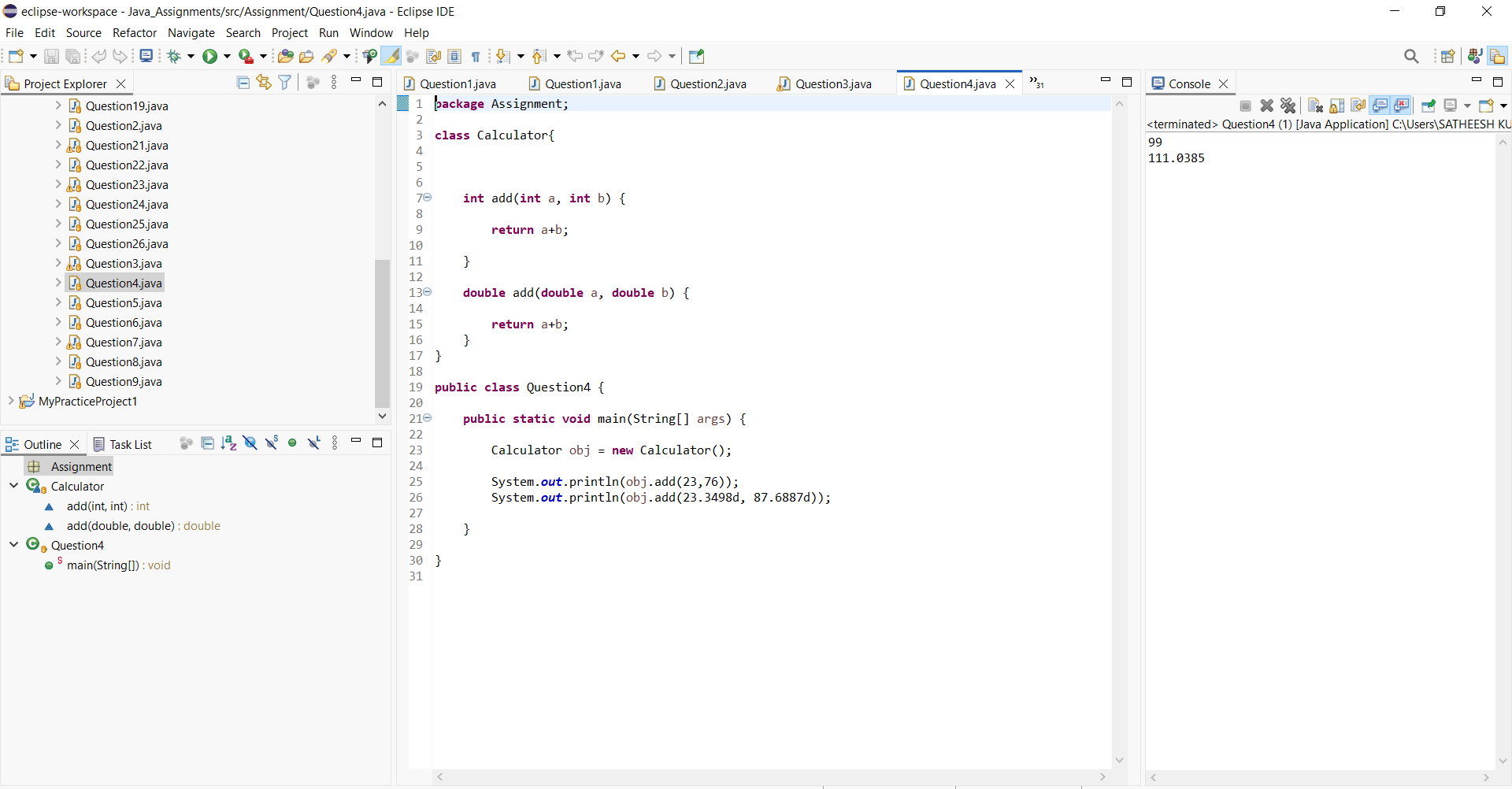
Create a class Calculator with overloaded methods add():

add(int a, int b)

add(double a, double b)

Call both methods inside the main method and print results.

***Program and output:***



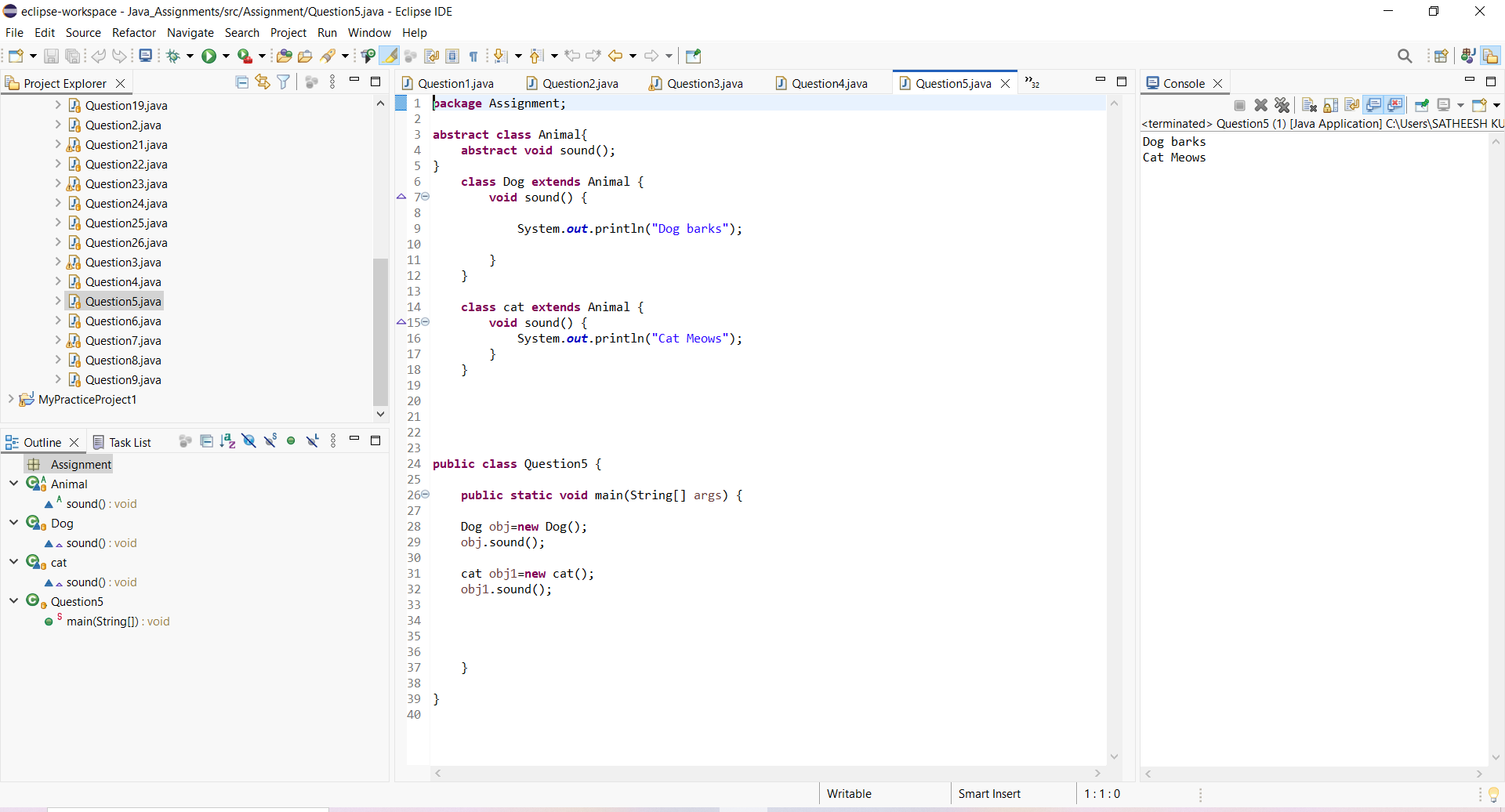
***5. Abstraction using Abstract Class***

Create an abstract class Animal with an abstract method sound().

Create two subclasses Dog and Cat and provide implementation for sound() method.

Create objects and call sound() for each.

***Program and output:***



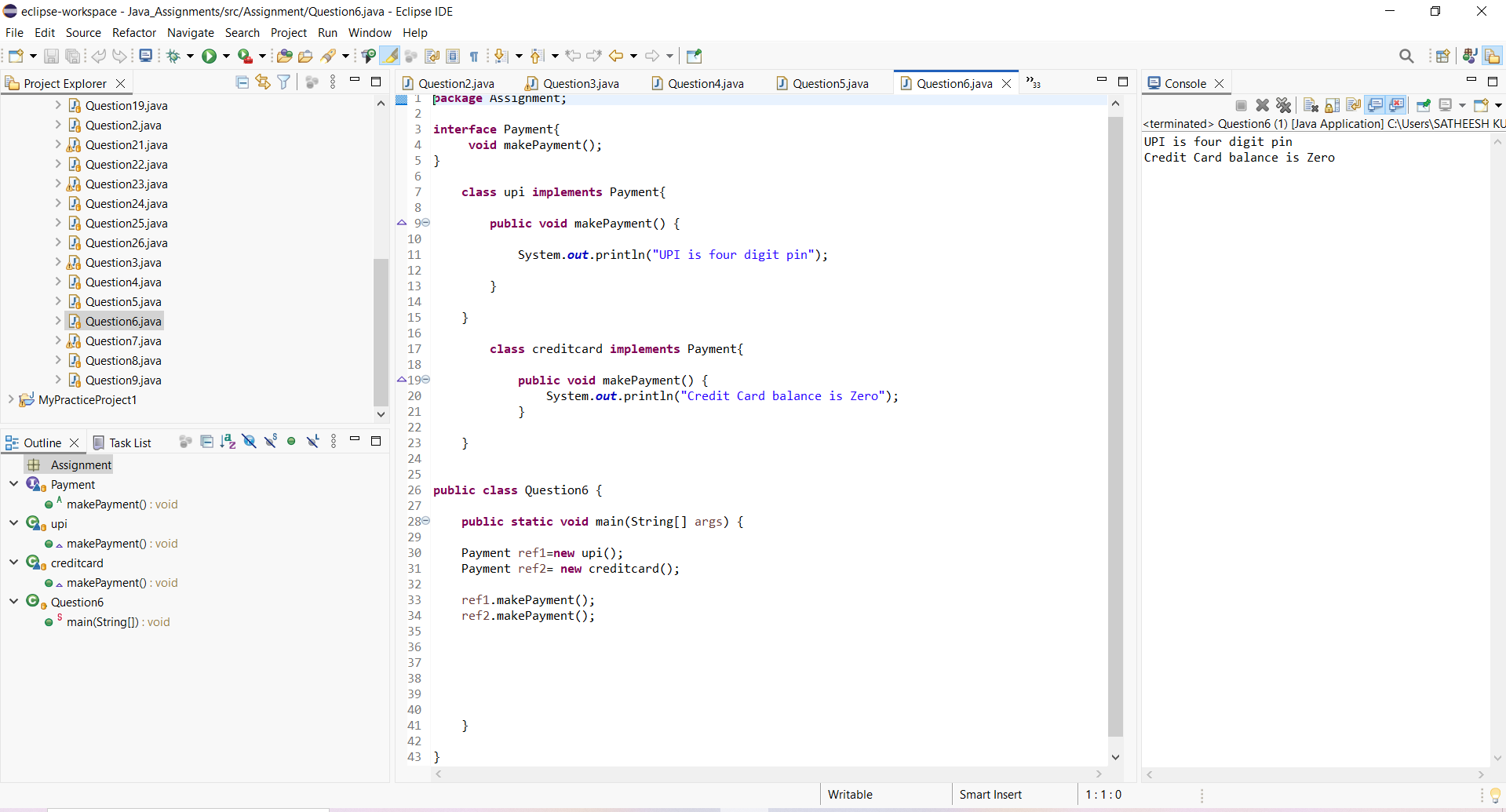
***6. Interface Implementation***

Create an interface Payment with a method makePayment().

Create two classes UPI and CreditCard implementing this interface and define their own payment messages.

Call the method through interface reference.

***Program and Output:***



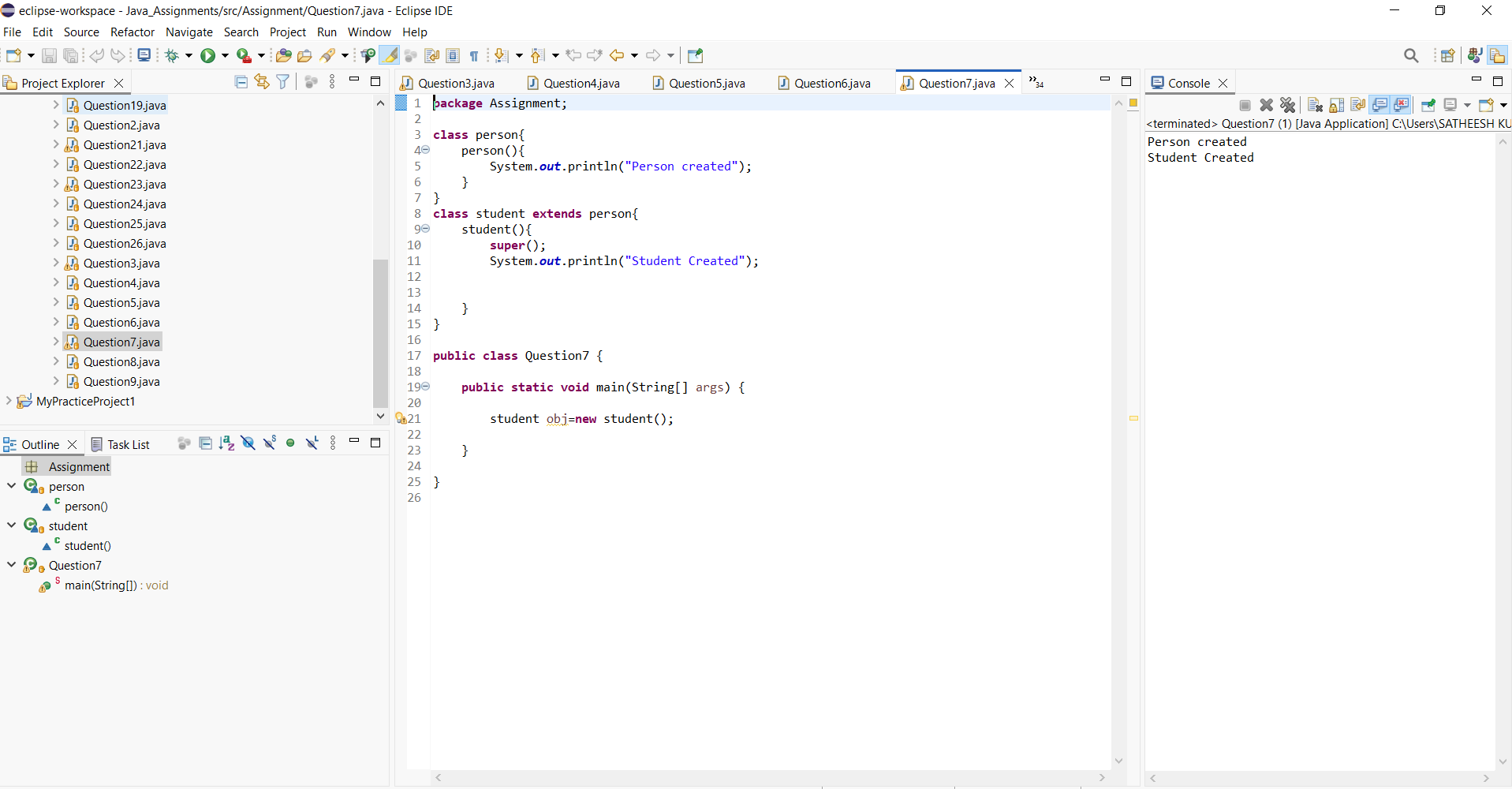
***7. Use of super keyword***

Create a class Person with a constructor that prints "Person Created".

Create a subclass Student that calls the parent constructor using super() and prints "Student Created".

Create an object and observe the order of execution.

***Program and Output:***



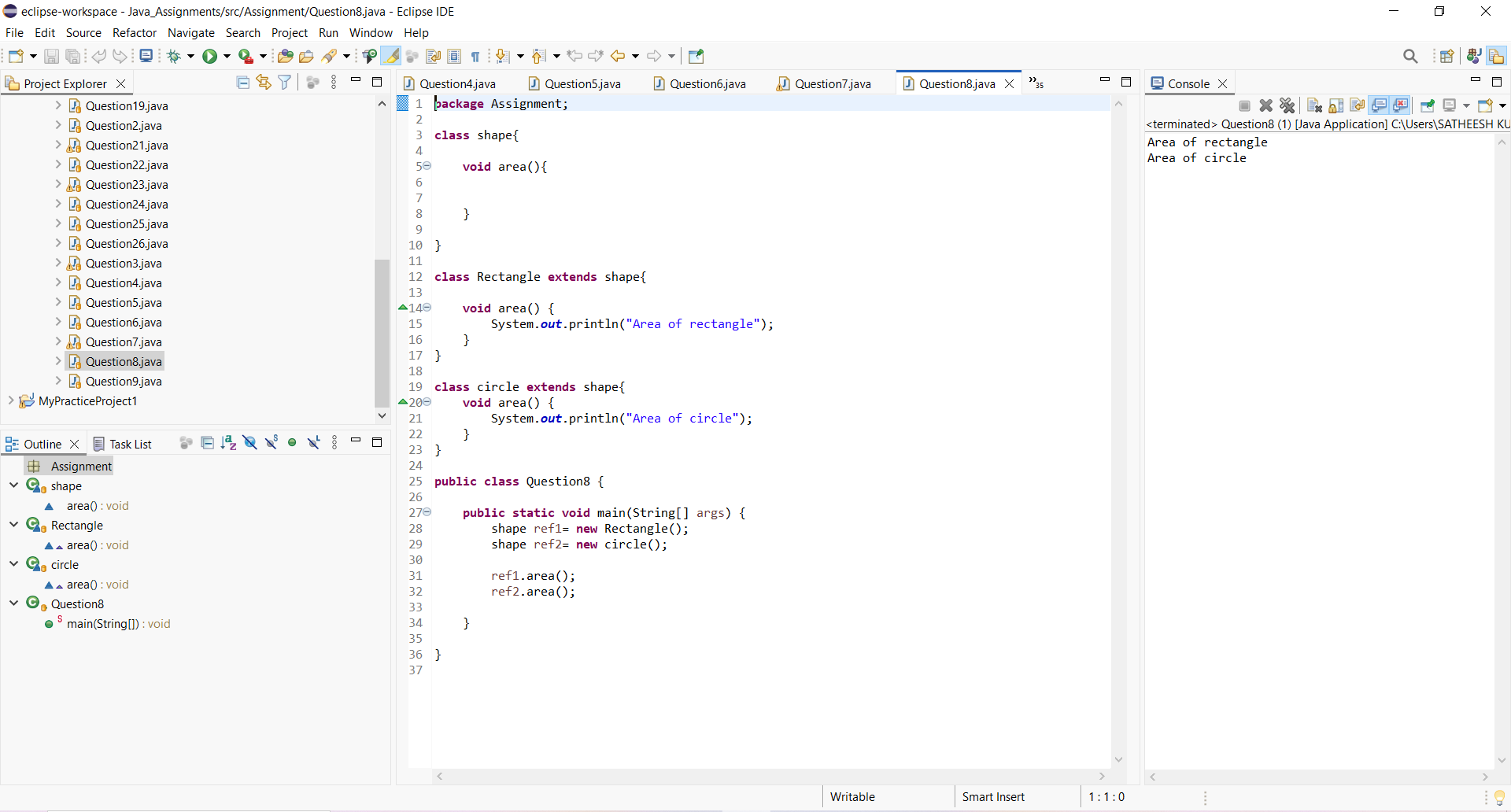
***8. Polymorphism (Dynamic Binding)***

Create a parent class Shape with a method area().

Create subclasses Rectangle and Circle and override the area() method.

Create a reference of Shape and assign objects of both subclasses one by one, calling area() each time.

***Program and output:***



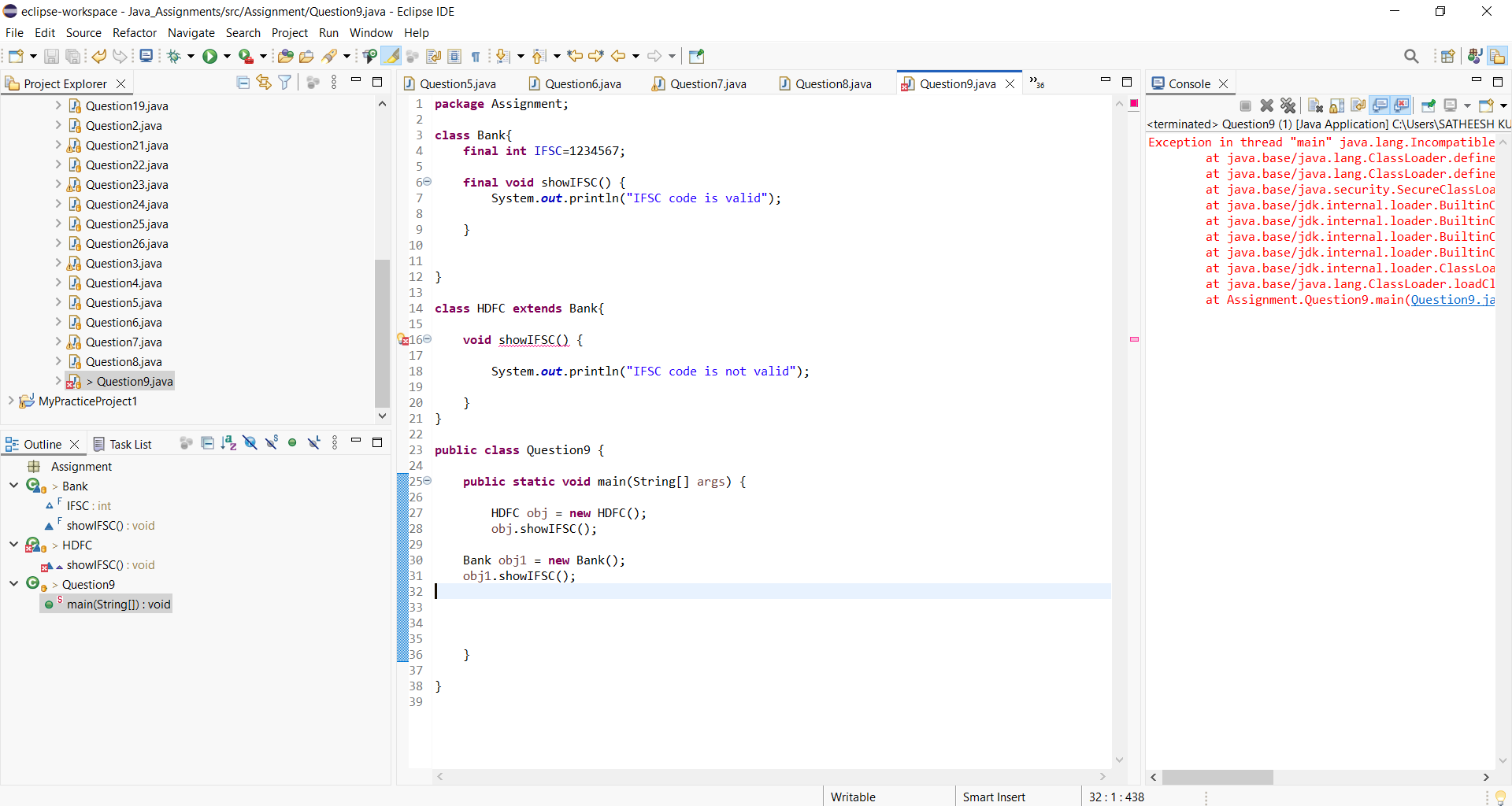
***9. Final Keyword***

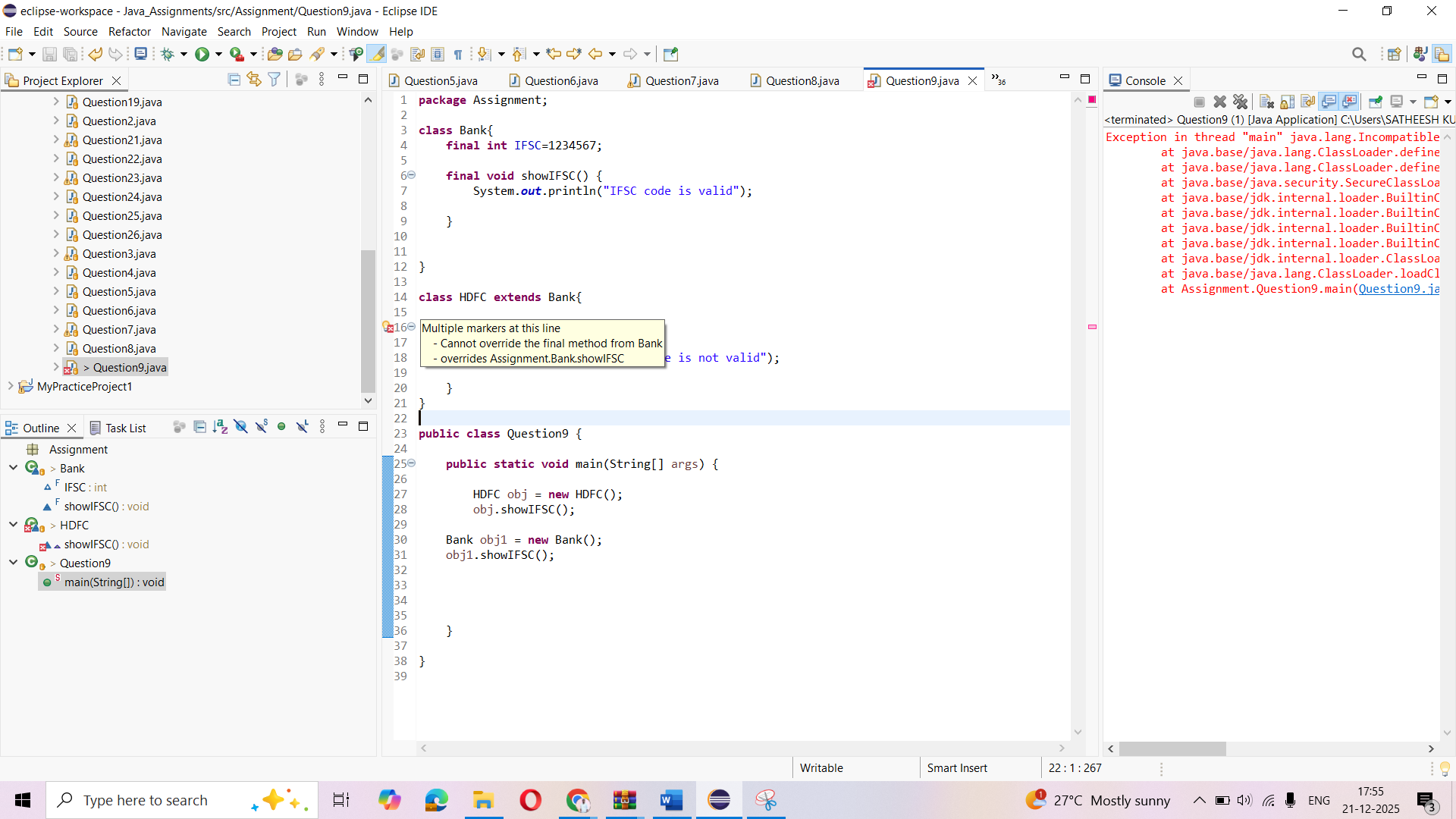
Create a class Bank with a final variable IFSC and final method showIFSC().

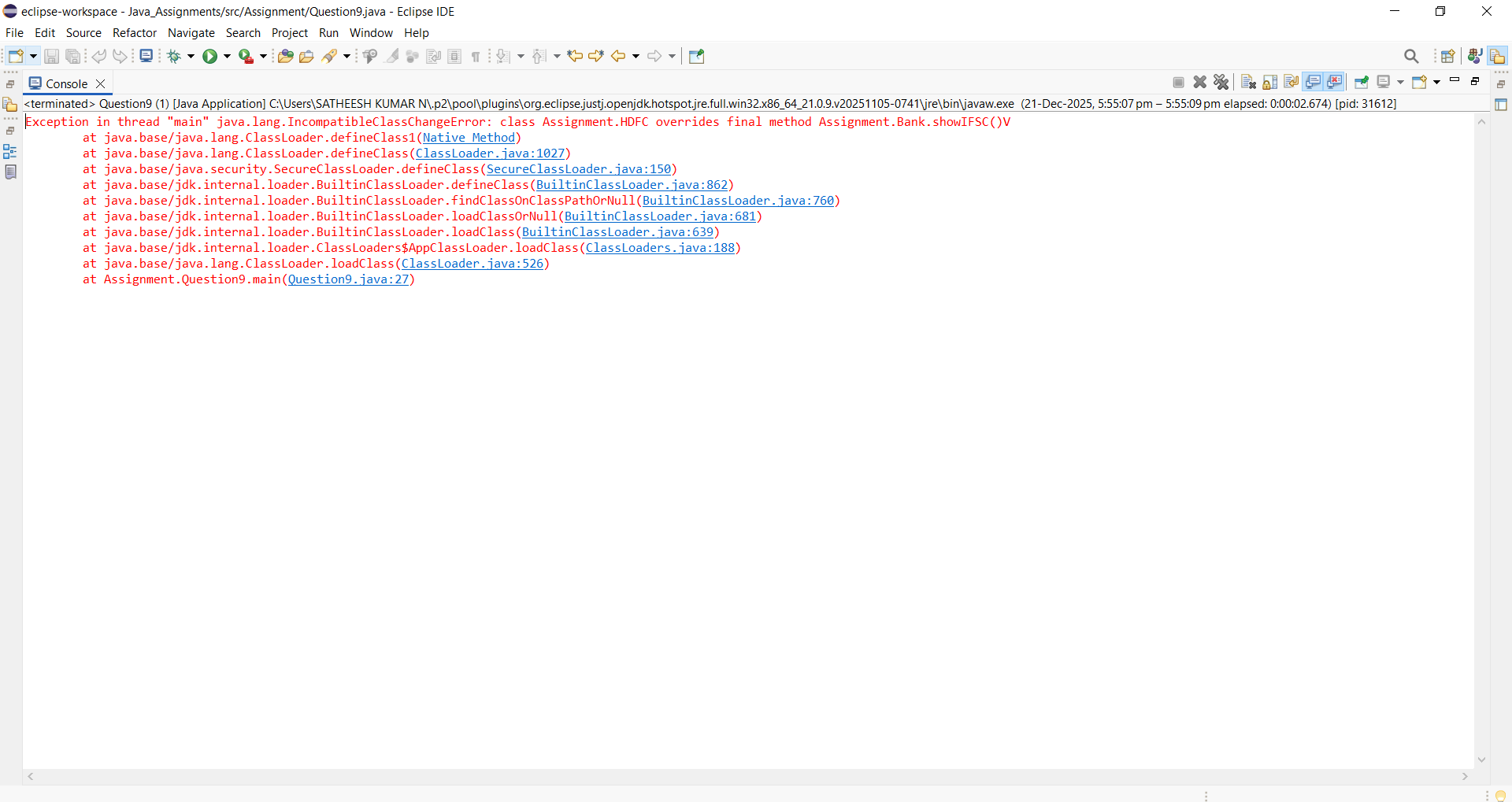
Try creating a subclass HDFCBank and attempt overriding the final method (should show compile-time restriction).

Create a main method to demonstrate usage.

***Program and Output:***







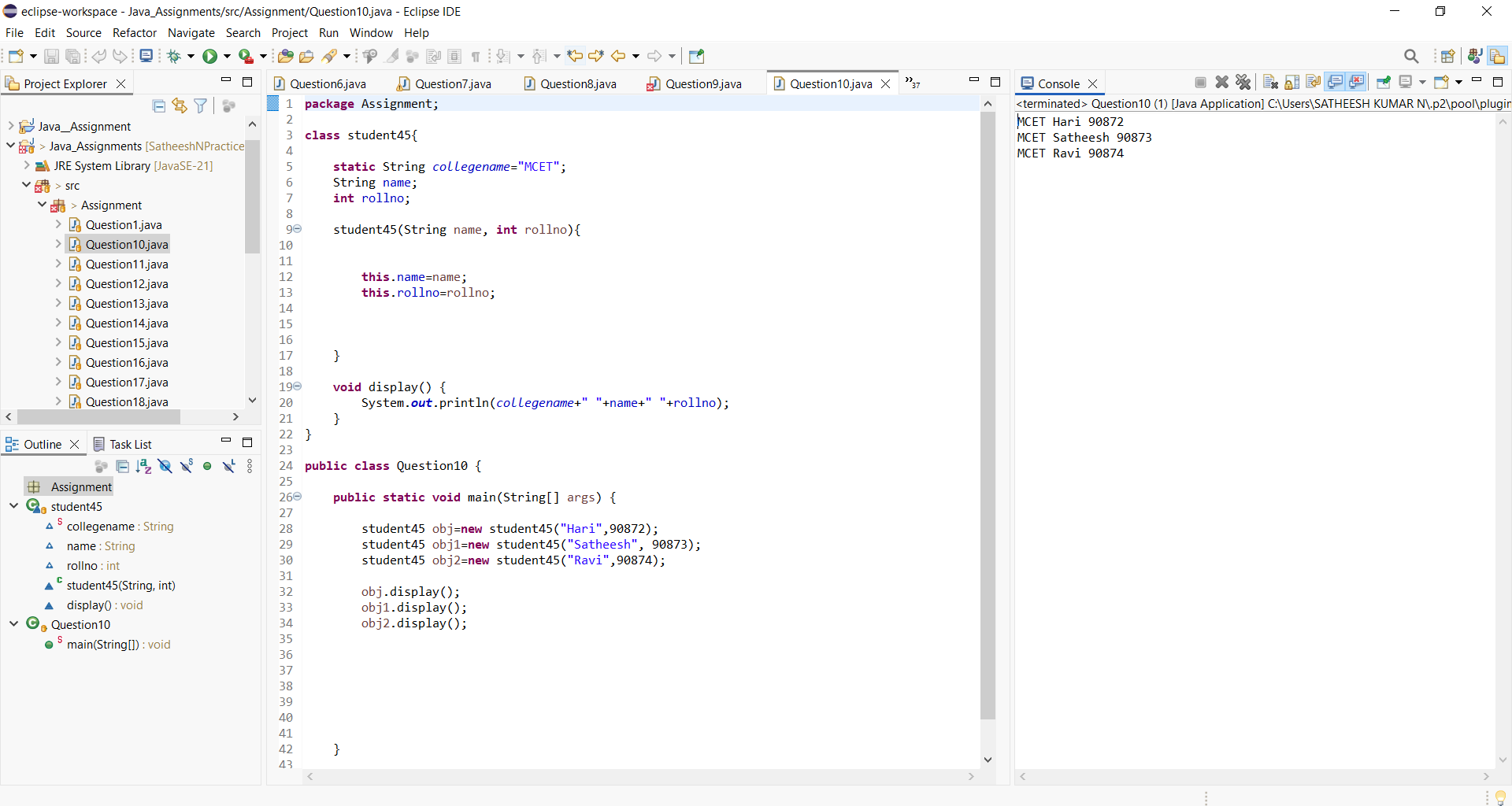
***10. Static Keyword***

Create a class Student having static variable collegeName and instance variables name and rollNo.

Write a method to print both static and instance data.

Create multiple objects to show static value remains constant.

***Program and Output:***



***11. Class + Object + Method***

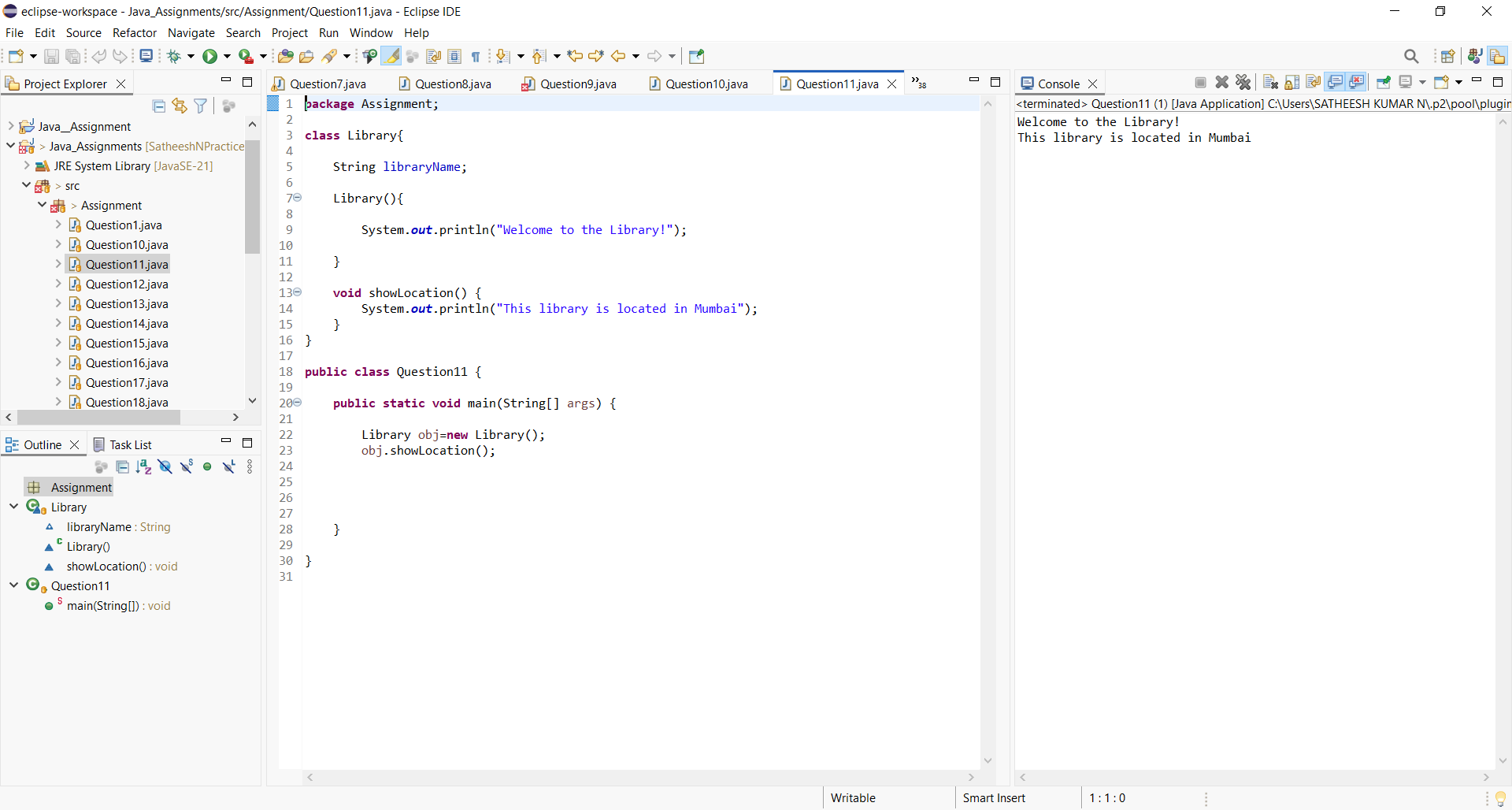
Create a class Library with an instance variable libraryName.

Create a default constructor to print "Welcome to the Library!".

Create a method showLocation() which prints "This library is located in Mumbai".

Create an object in main() and call both.

***Program and Output:***



***12. Encapsulation + Validation Logic***

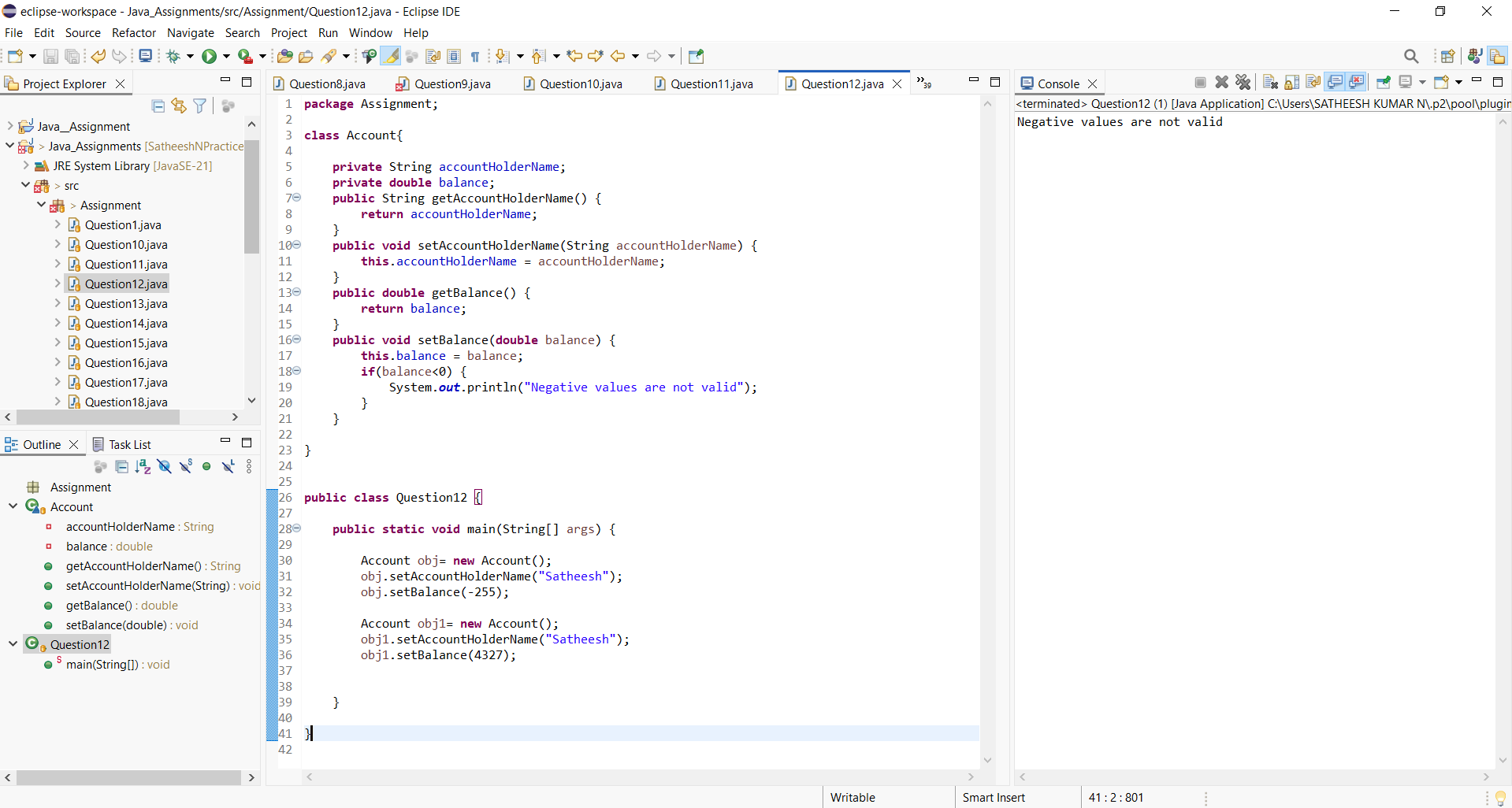
Create a class Account with private variables accountHolderName and balance.

Provide setters and getters, where:

setBalance() should not accept negative values (print a warning).

Create an object and update values through setters only.

***Program and Output:***



***13. Inheritance (Multilevel)***

Create three classes:

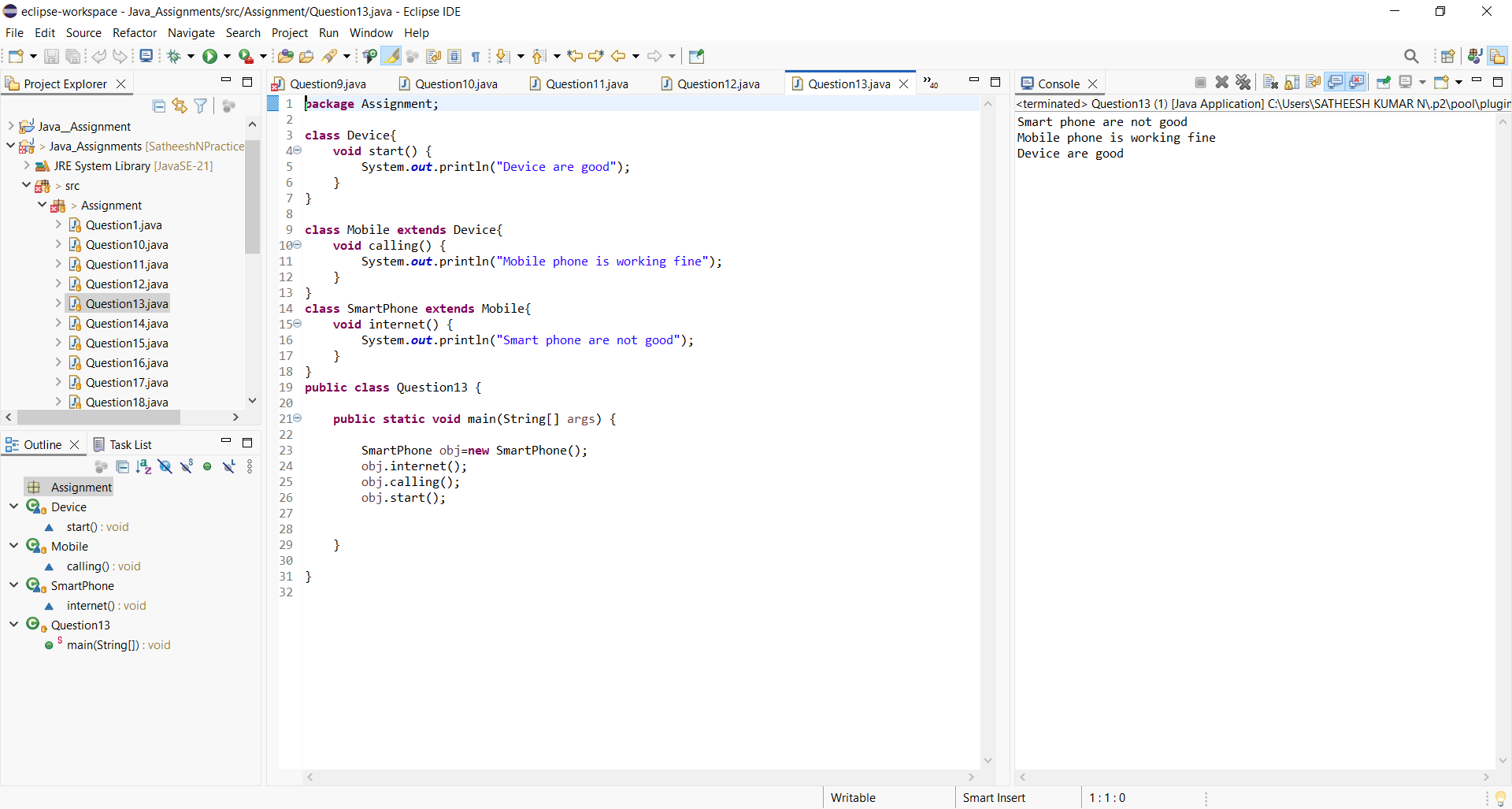
Device â†’ method start()

Mobile extends Device â†’ method calling()

SmartPhone extends Mobile â†’ method internet()

Create object of SmartPhone and call all methods.

***Program and Output:***



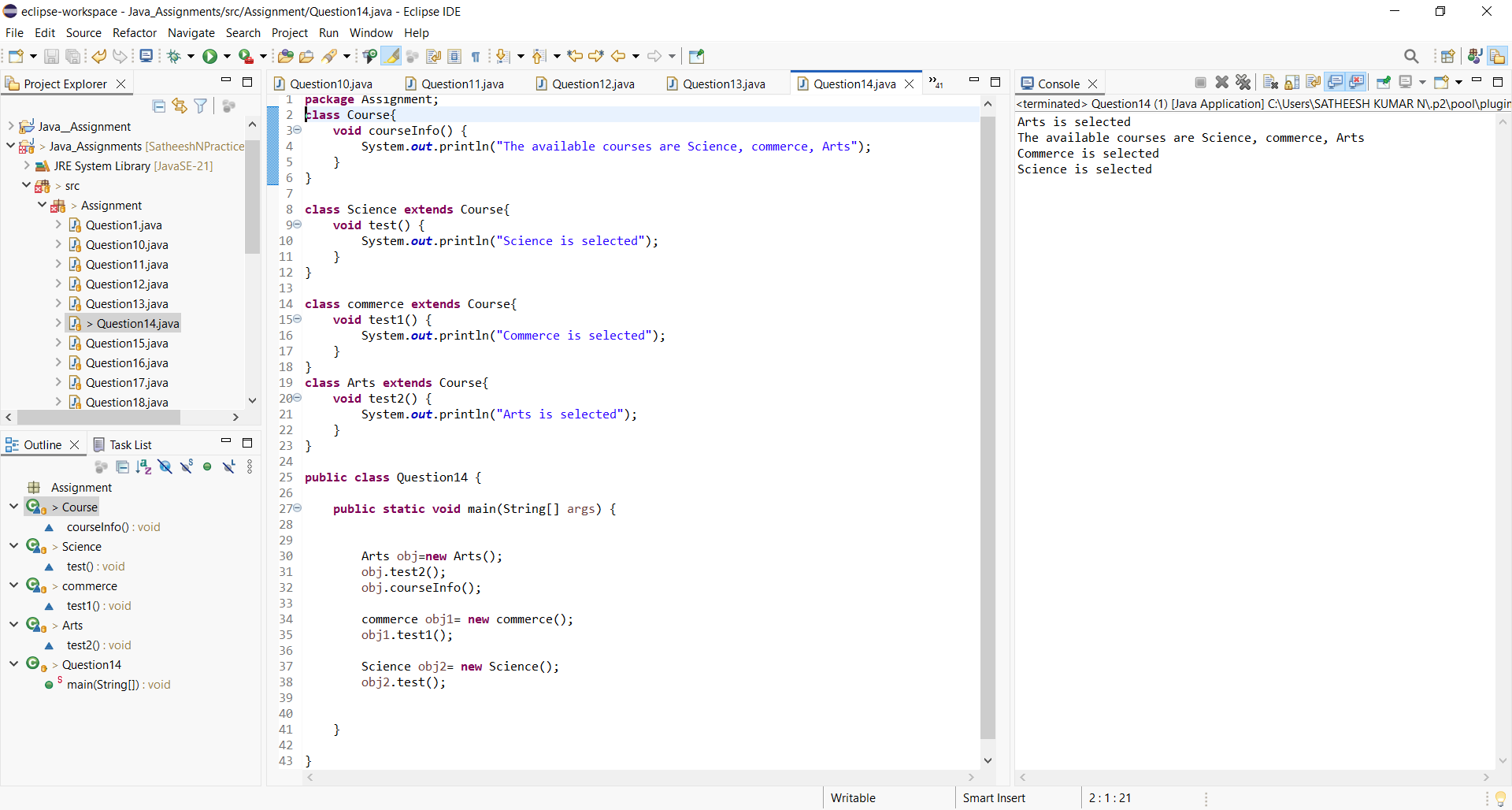
***14. Hierarchical Inheritance***

Create a class Course with a method courseInfo().

Create subclasses Science, Commerce, and Arts each with their own method.

Create objects of each and call methods to show hierarchy.

***Program and Output:***



***15. Method Overloading (Bank Scenario)***

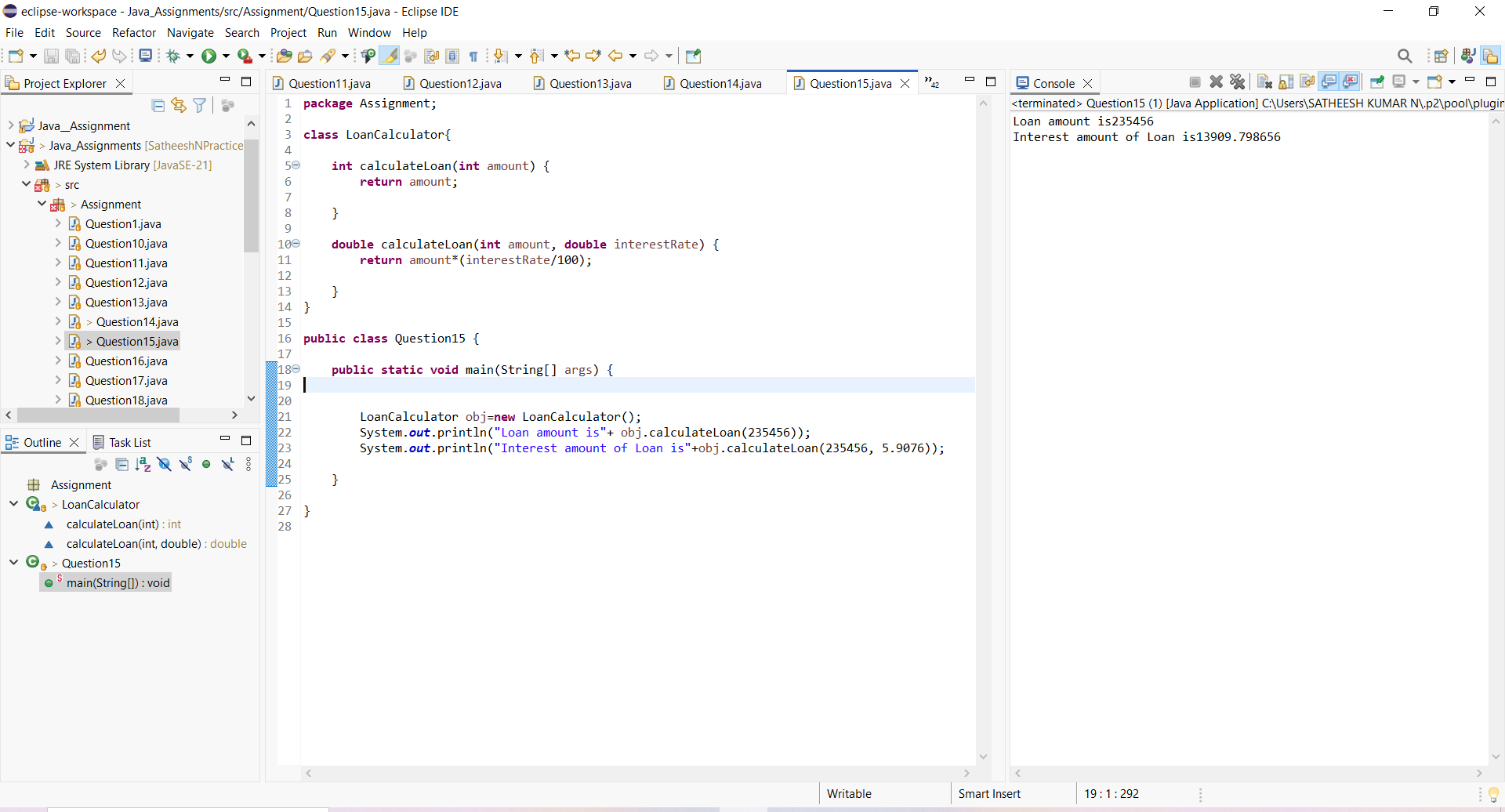
Create a class LoanCalculator with two overloaded methods:

calculateLoan(int amount)

calculateLoan(int amount, double interestRate)

Print loan details accordingly. Call both methods from main.

***Program and Output:***



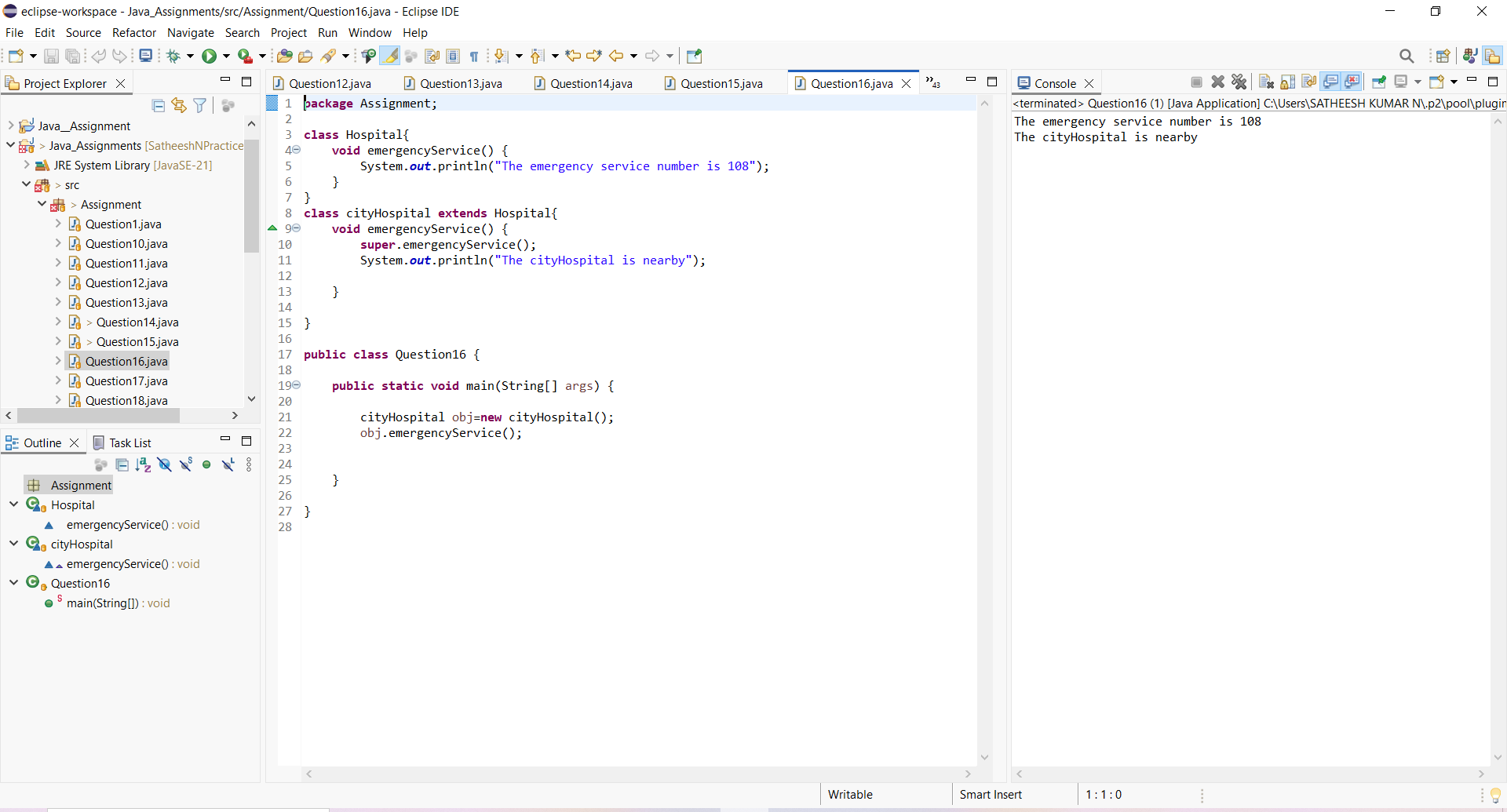
***16. Method Overriding with super***

Create a base class Hospital with a method emergencyService().

Create a subclass CityHospital that overrides the method and calls parent method using super.emergencyService().

Demonstrate overriding in main.

***Program and Output:***



***17. Abstract Class + Real Usage***

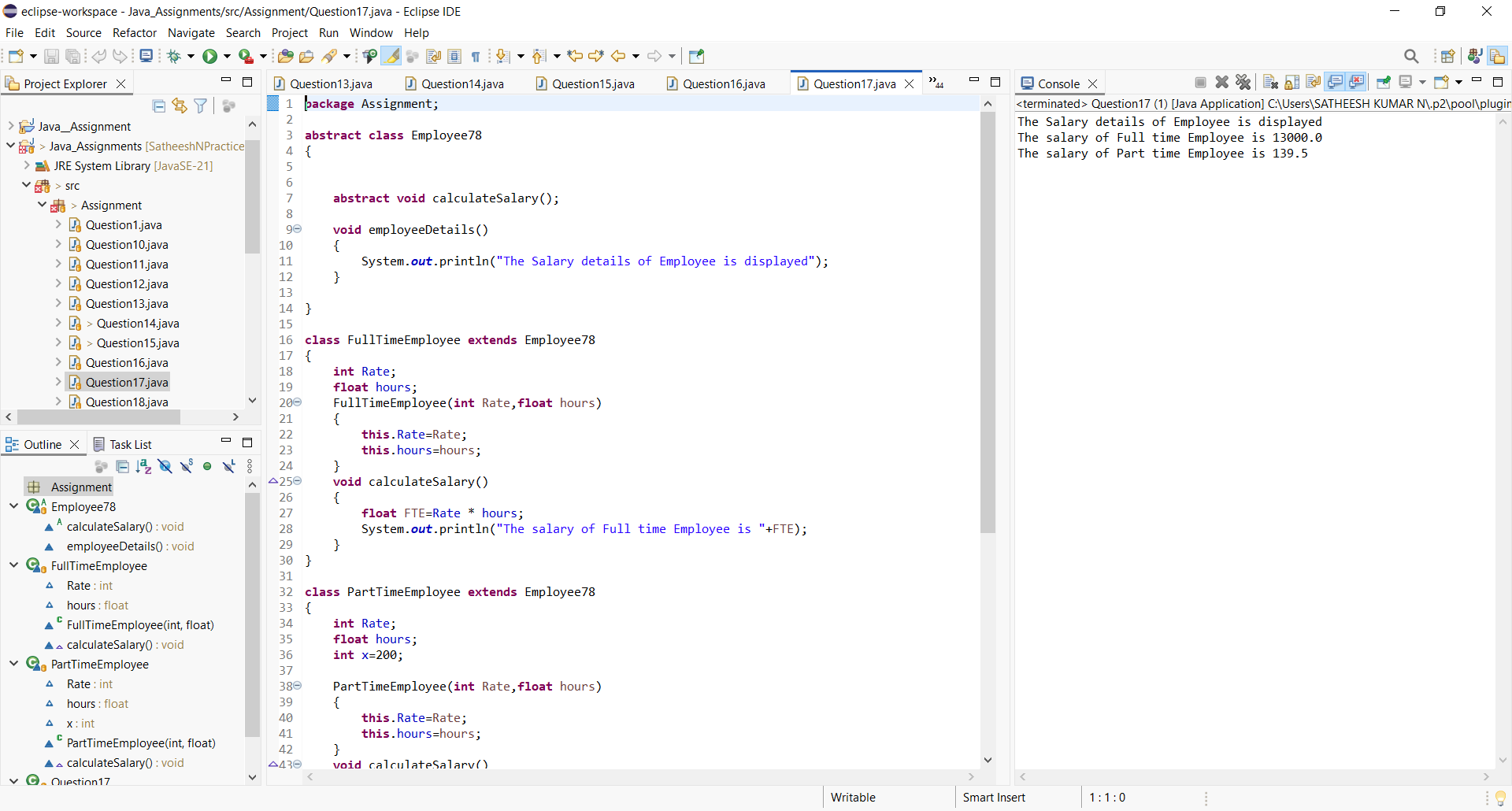
Create an abstract class Employee with:

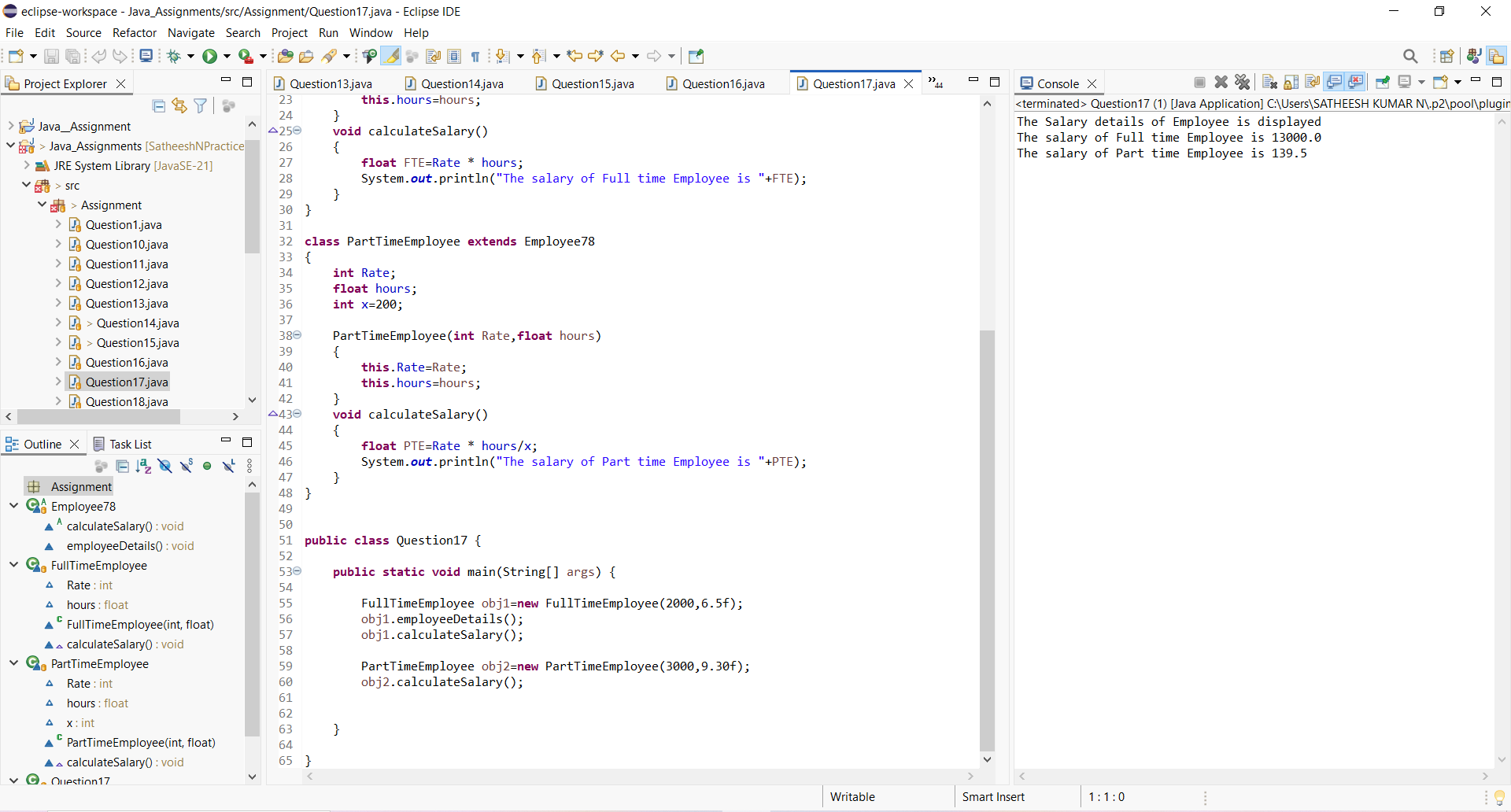
abstract method: calculateSalary()

concrete method: employeeDetails()

Subclass FullTimeEmployee and PartTimeEmployee implementing salary calculation logic differently.

***Program and Output:***





***18. Interface with Multiple Implementations***

Create an interface Transport with method booking().

Implement it in Bus and Flight classes.

Call using interface reference.

***Program and Output:***



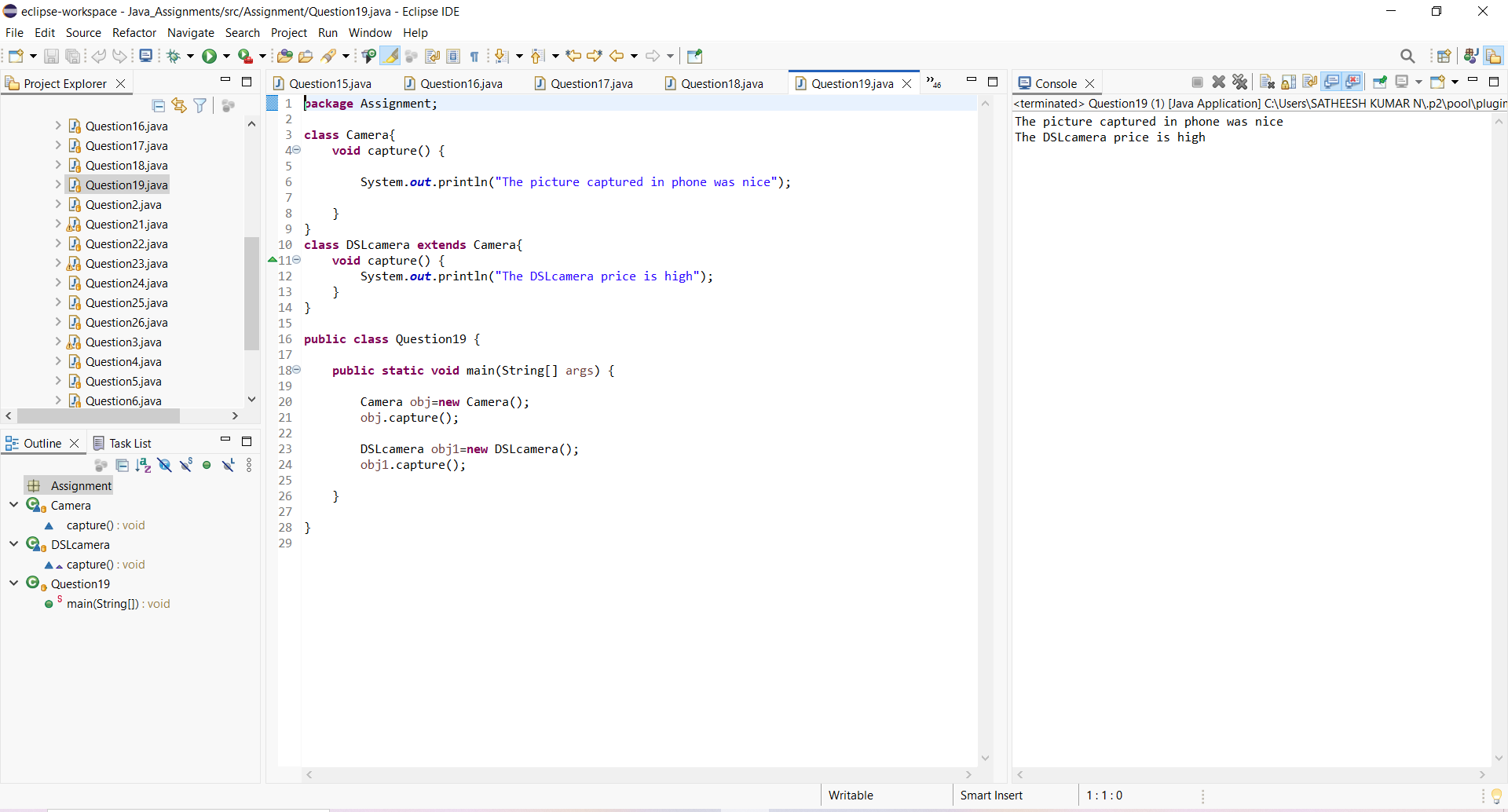
***19. Polymorphism (Runtime + Upcasting)***

Create a class Camera with a method capture().

Create a subclass DSLCamera that overrides the method.

Use parent reference to call child object method (dynamic polymorphism).

***Program and Output:***



***21. Static Concepts***

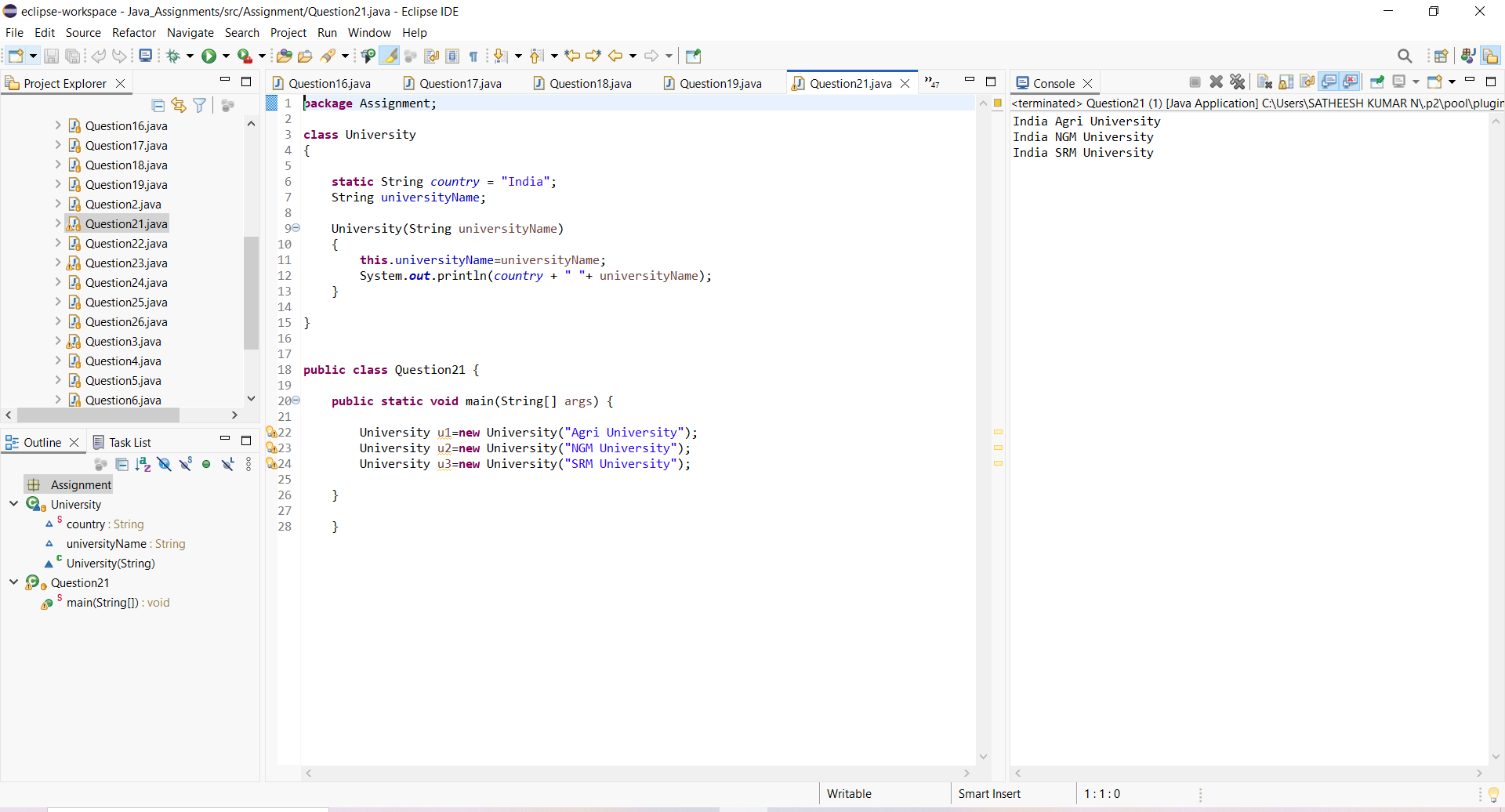
Create a class University with:

static variable country = "India"

instance variable universityName

Print values using different objects to show static effect.

***Program and Output:***

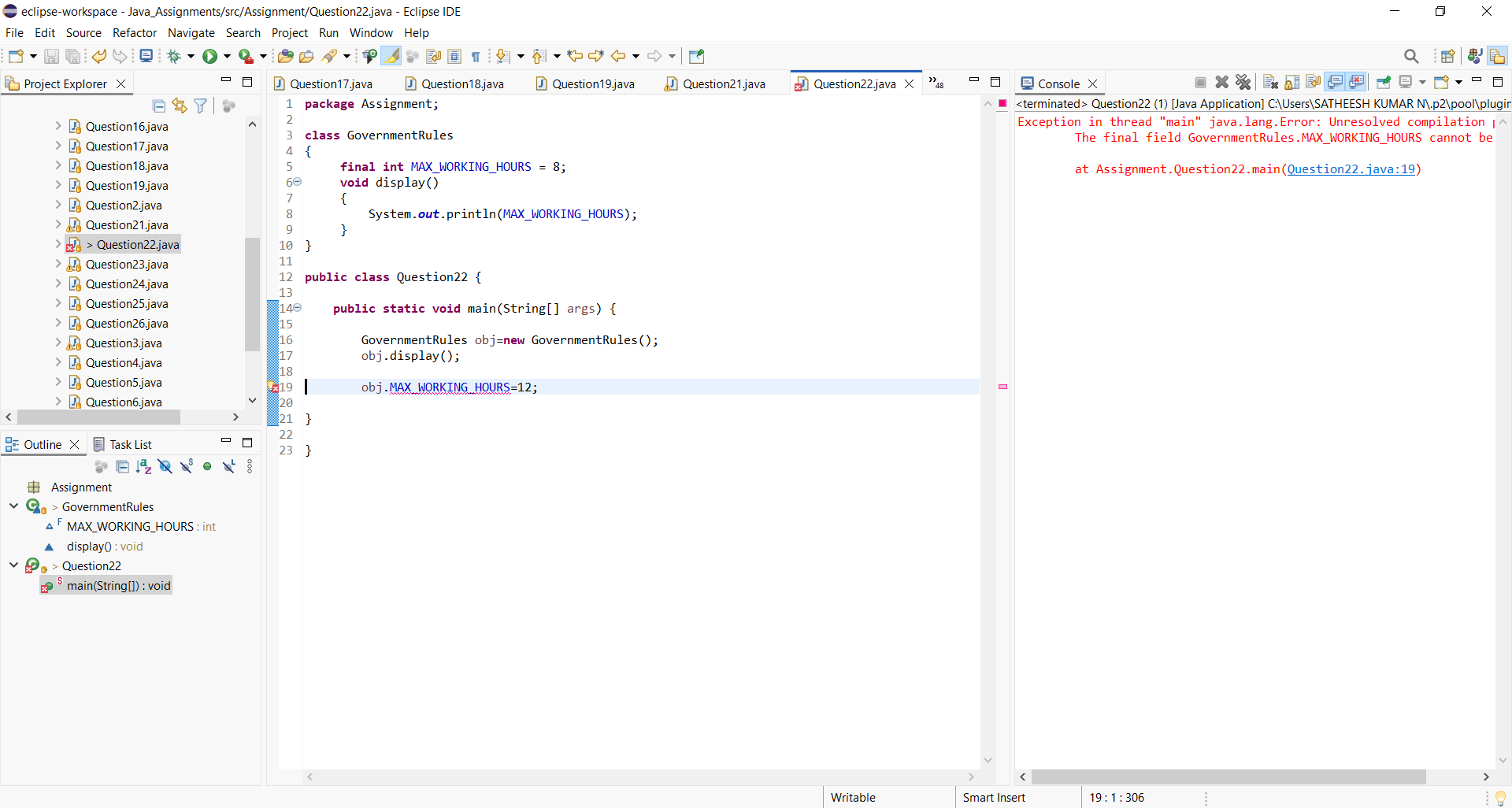


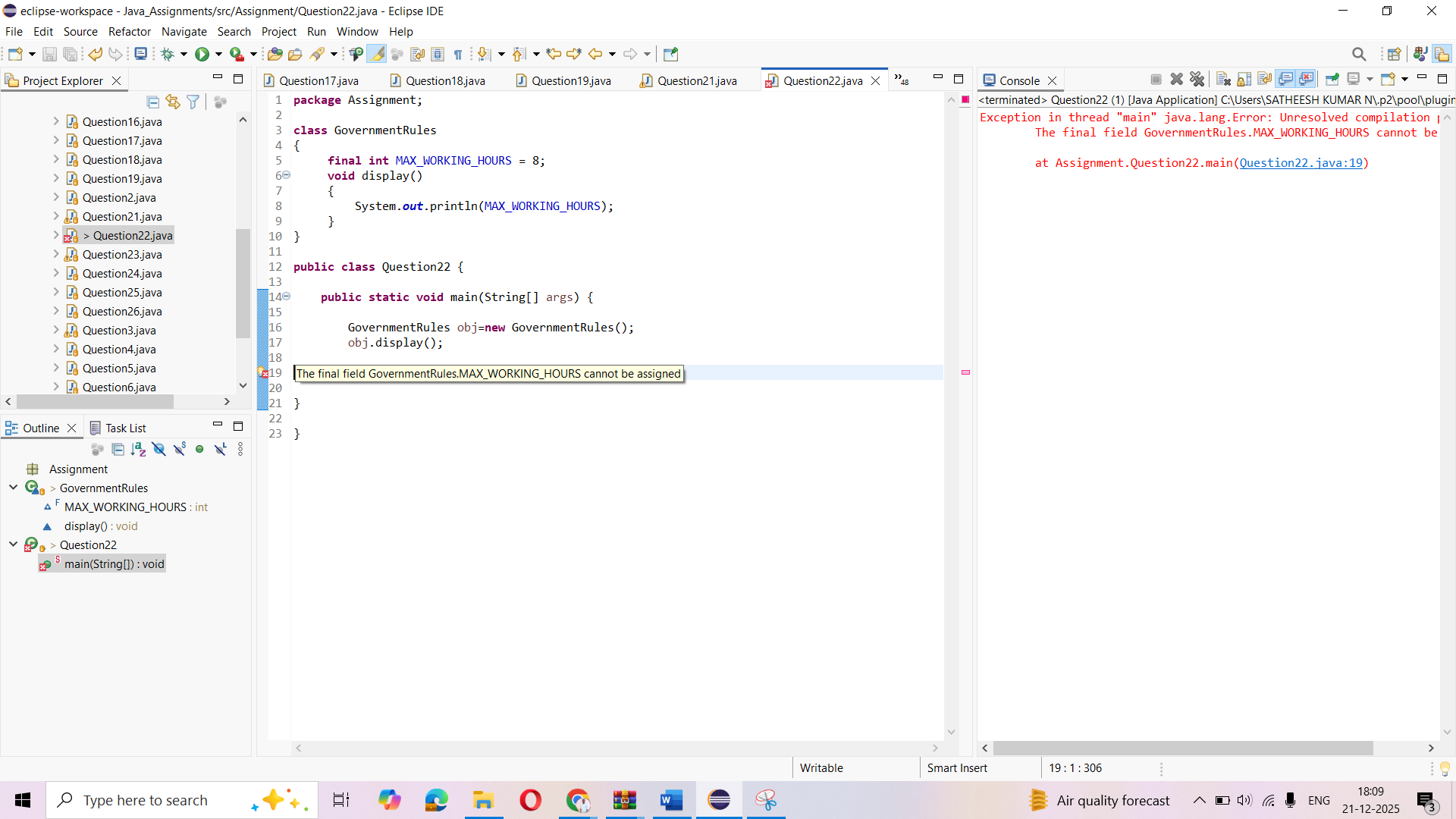
***22. Final Keyword + Constant***

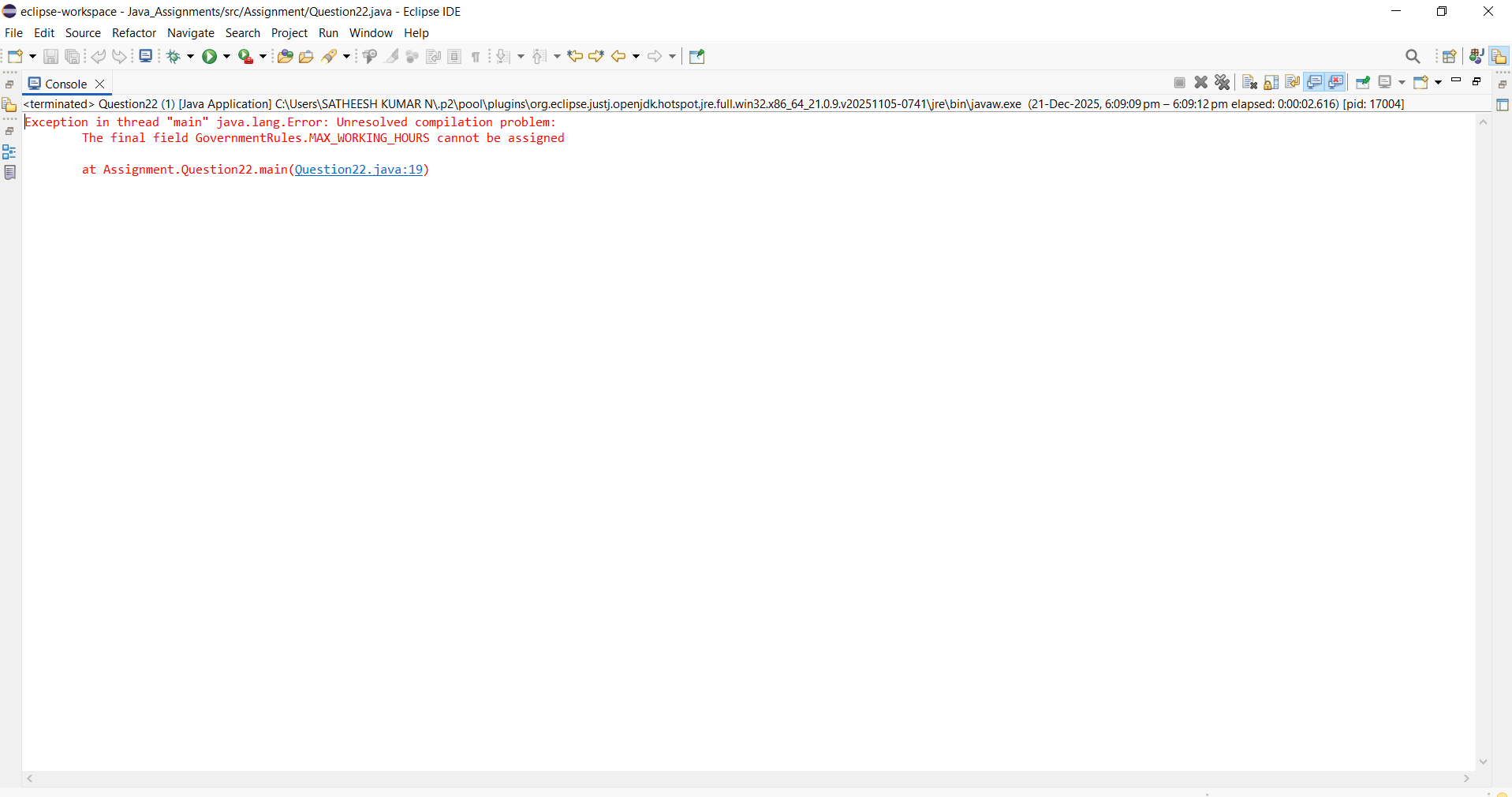
Create a class GovernmentRules with a final variable MAX\_WORKING\_HOURS = 8

Try modifying it inside main and observe compile-time restriction.

***Program and Output:***







***23. Constructor Chaining***

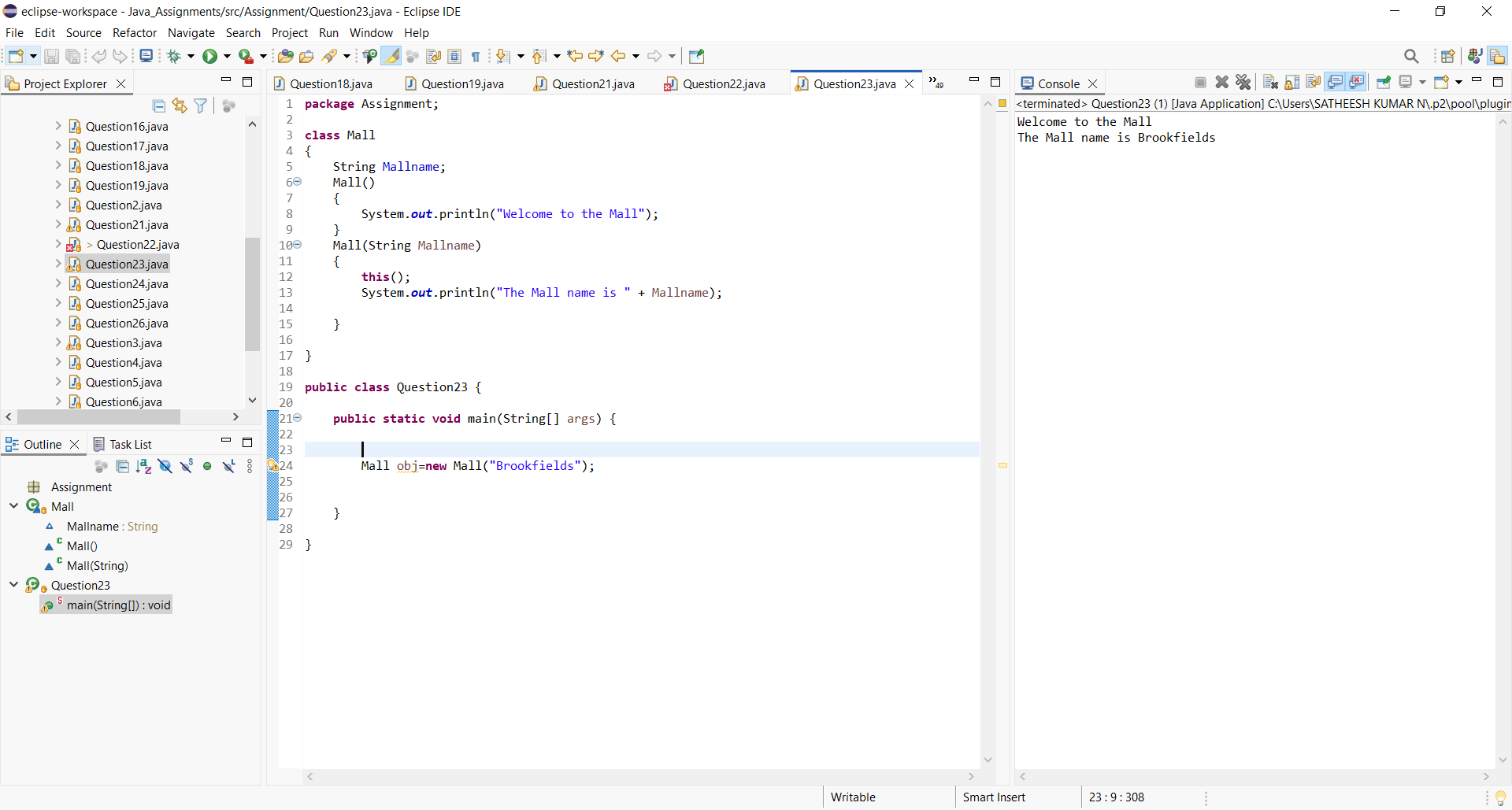
Create a class Mall with:

Default constructor printing "Welcome to the Mall"

Parameterized constructor calling default constructor using this()

Demonstrate constructor chaining in main.

***Program and Output:***



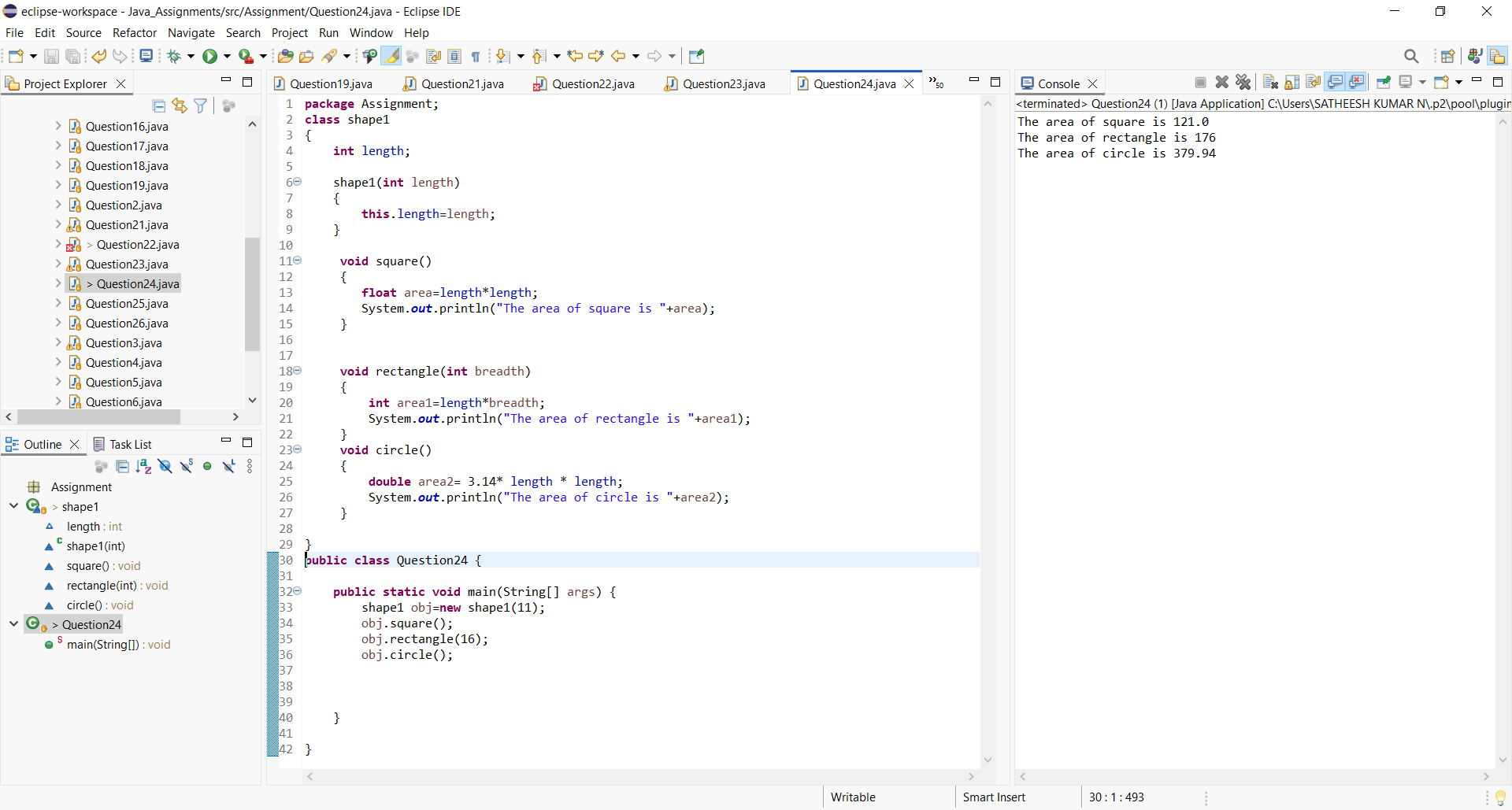
***24 > WAP in Java***

Create a Class named Shape with length as instance variable , create three methods as square , rectangle , circle

and find out their respective areas

Create a object in main method and call these different methods with the instance of object

***Program and Output:***



***25> WAP in Java***

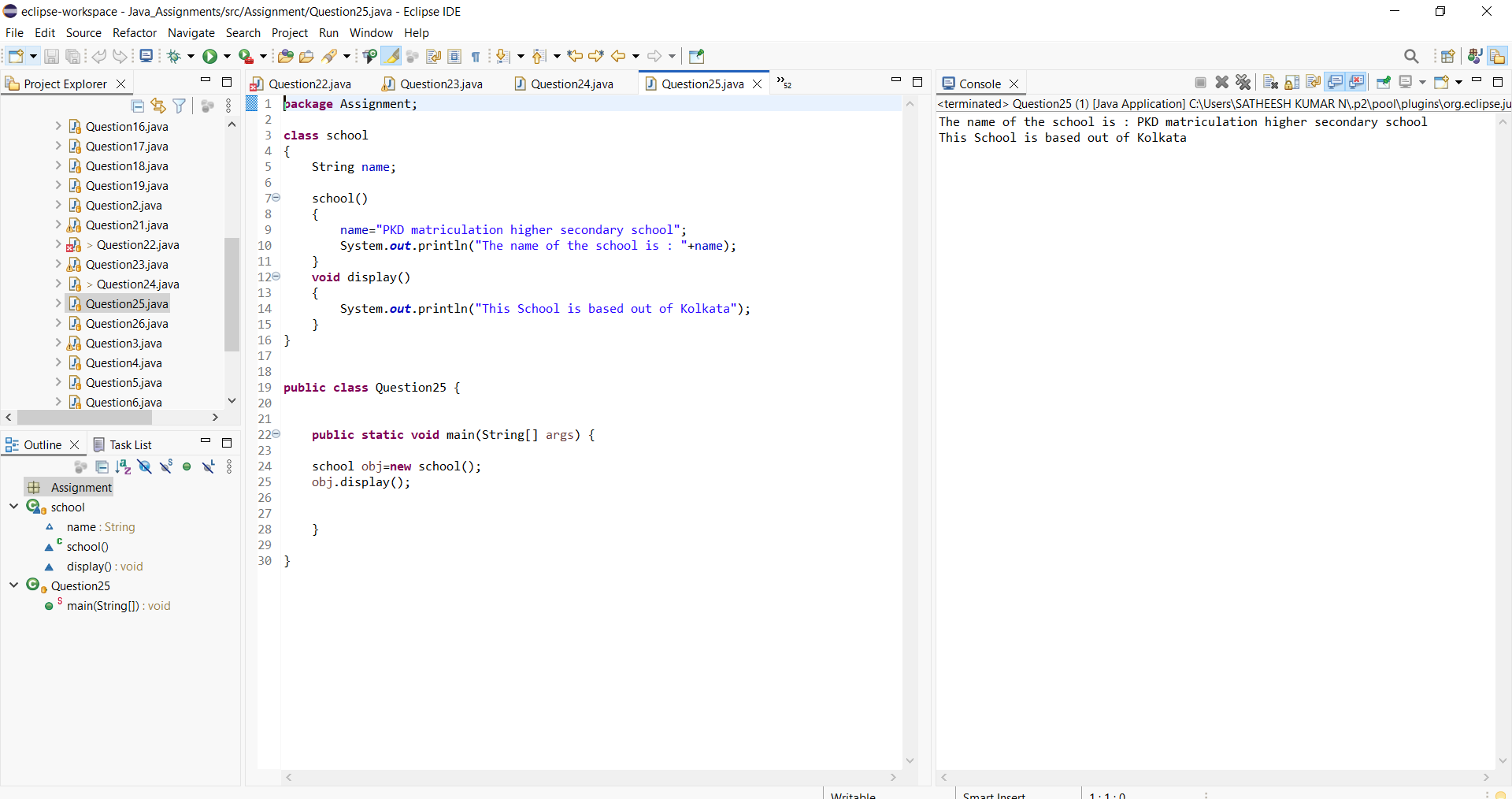
Create a class named school ,create name as their instance variables

Create a default constructor of this class which will have a print statement to display the name of school

Create a method inside the class which will display a message as "This School is based out of Kolkata"

Create a object under main method and call the constructor and the method

***Program and Output:***



26> WAP in Java to create a class named school

create name, address,strength as their instance variables

Create two constructor one with two variables and one with all the three variables

Create a method that will display all the three parameters

create two object of this class and call the respective methods

***Program and Output:***

