Week 9 - S9 - Advanced OOP - Object Class Methods, Inner Classes - Assignment Problem (HW)

**Name:** Ramesh Harisabapathi Chettiar

**Date of Submission:**15/10/25

**Problem Statement 1:**

Create a class Employee with fields id, name, and salary. Override the toString()

method to print employee details in a readable format. In the main method, create multiple

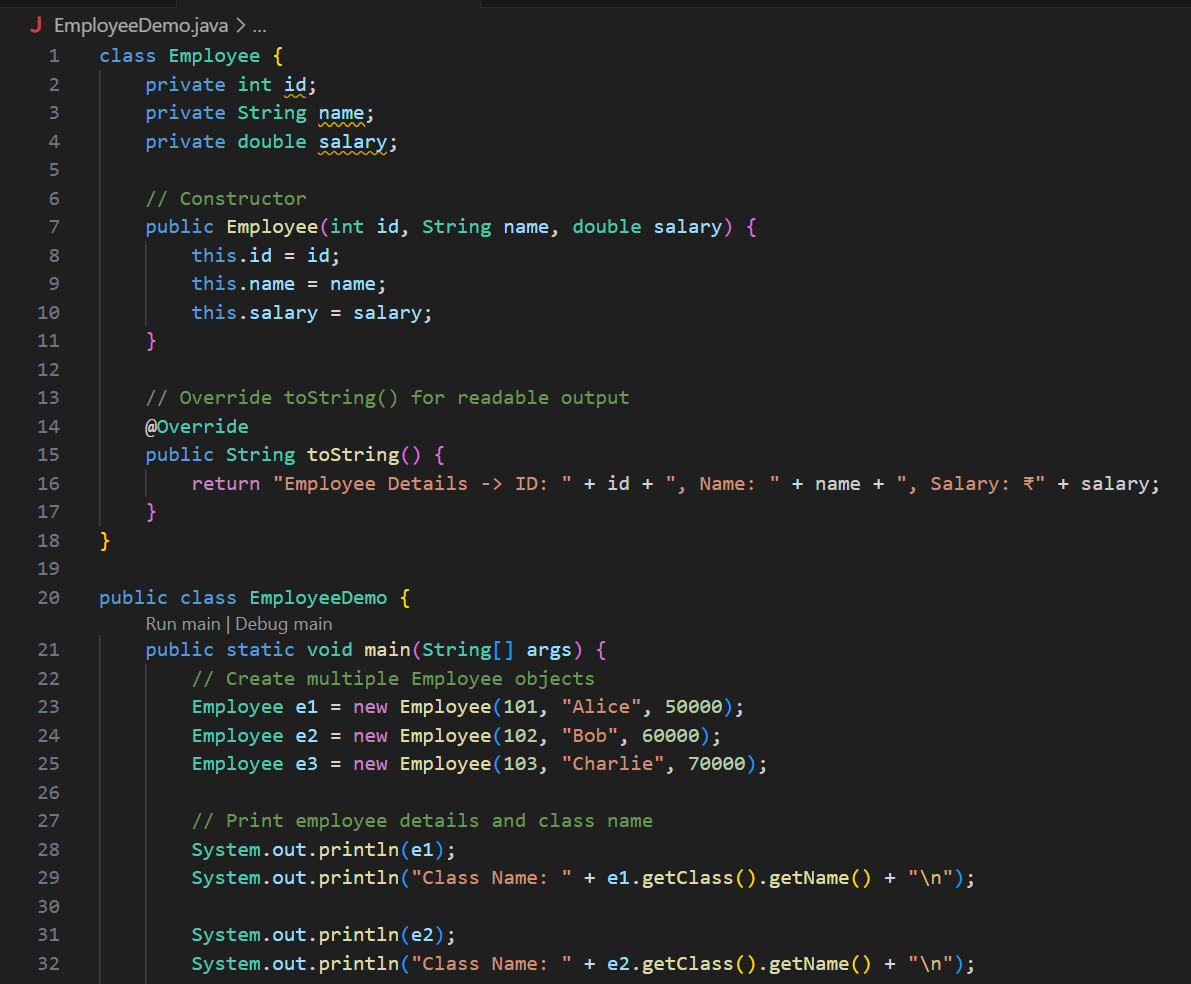
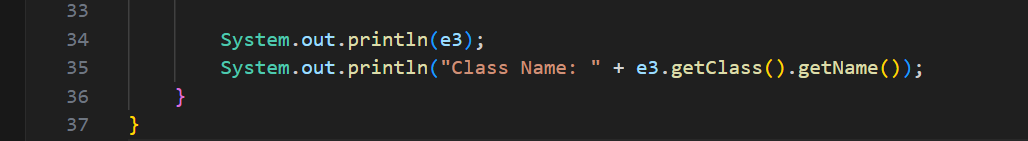
Employee objects and print their class name using getClass().getName().

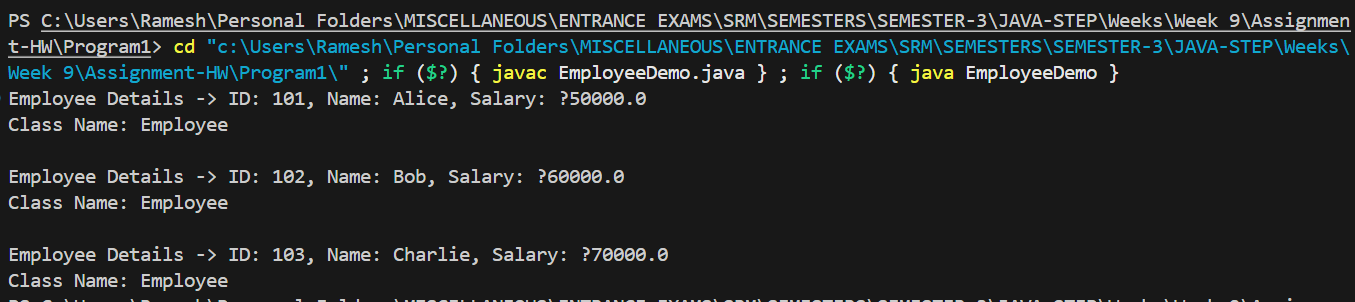
Hints:

● Override toString() to provide a meaningful string representation.

● Use getClass() to obtain runtime class information.

● Display both the object details and its class name.

EmployeeDemo.java  

**OUTPUT🡪** ****

**Problem Statement 2:**

Create a class Product with productId and productName fields. Compare two Product

objects using both == and .equals() to demonstrate the difference between reference and

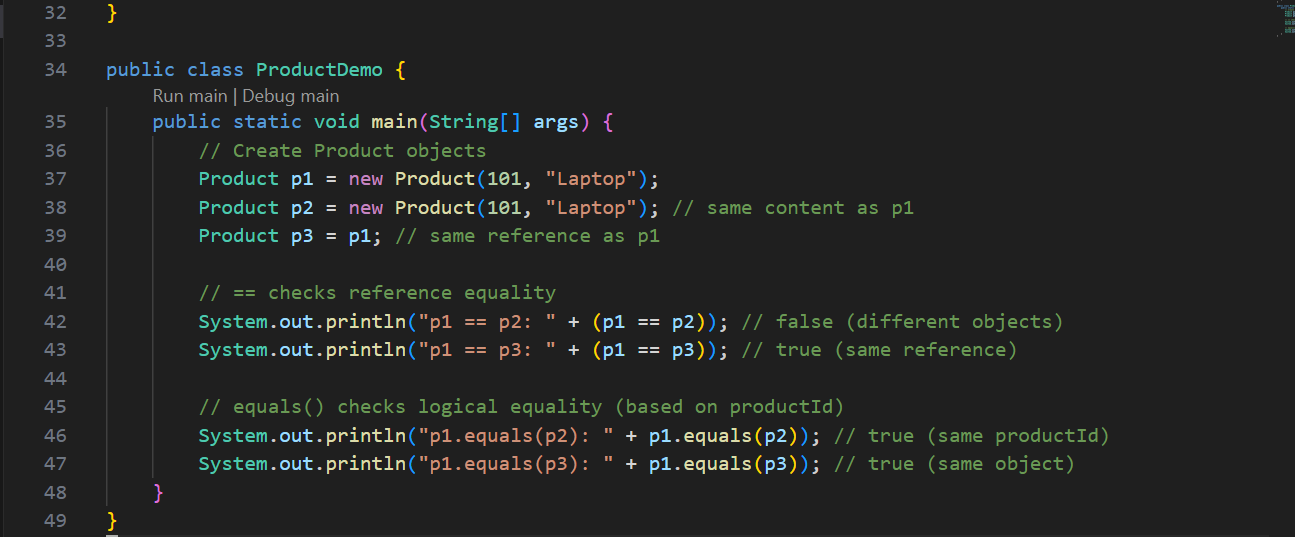
content comparison. Override the equals() method to compare objects by productId.

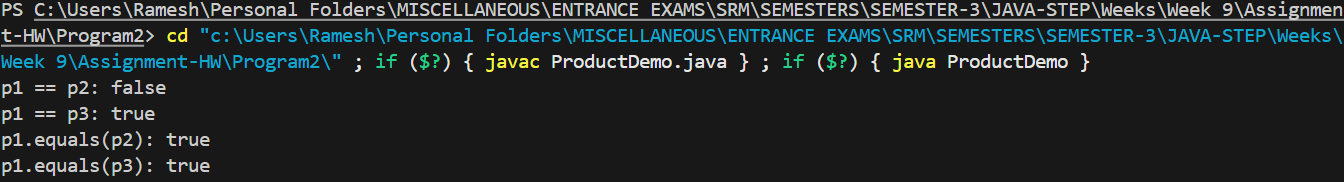
Hints:

● == checks reference equality, .equals() checks logical equality.

● Override equals() properly using the @Override annotation.

● Print results of both comparisons for clarity.

ProductDemo.java  

**OUTPUT🡪** ****

**Problem Statement 3:**

Create a Student class with rollNo and name fields. Override both equals() and

hashCode() so that two students with the same roll number are considered equal.

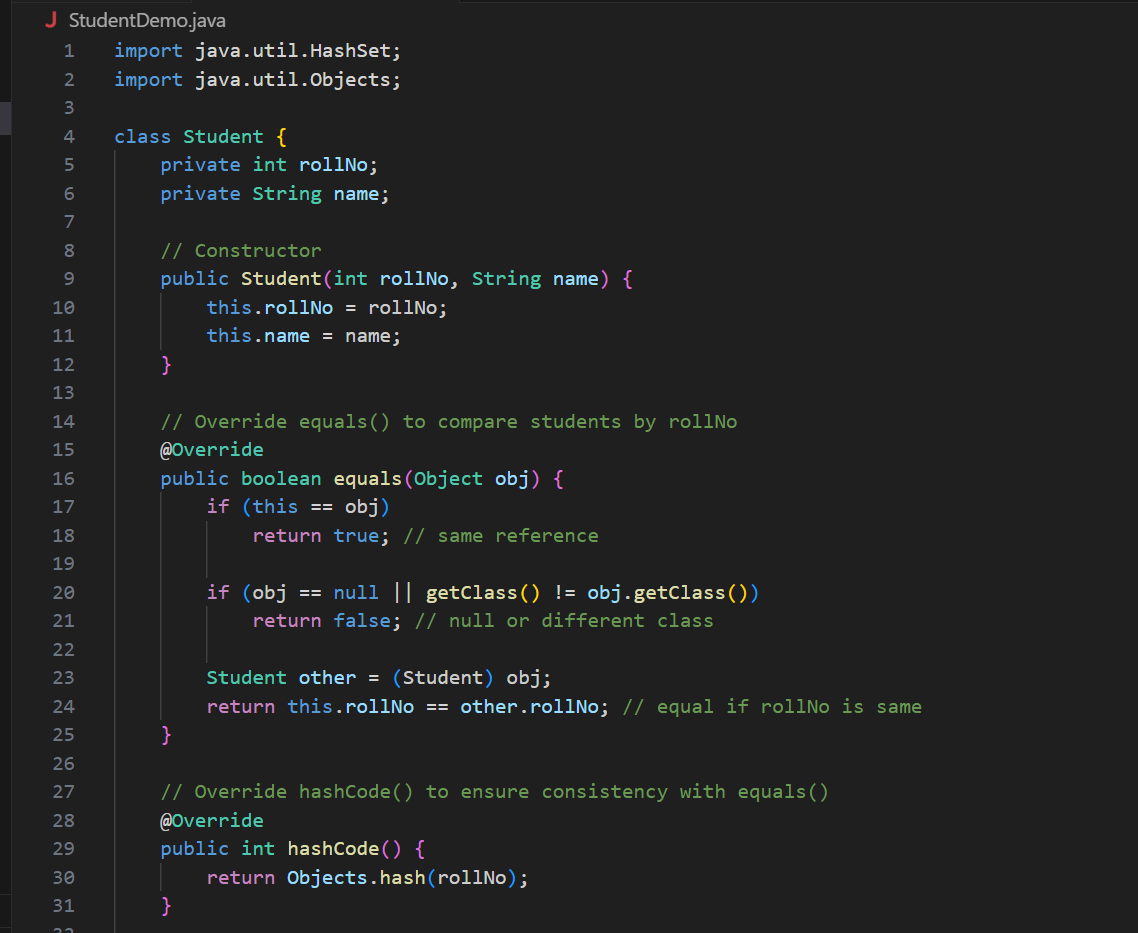
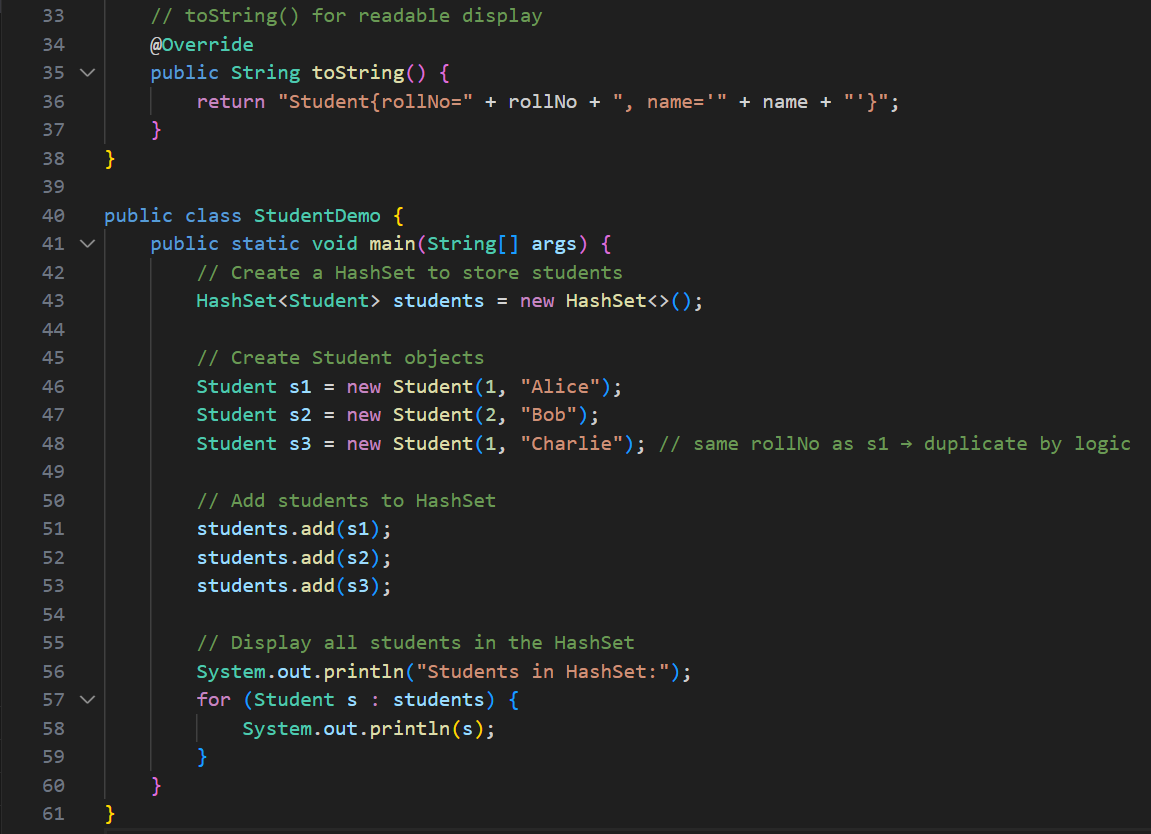
Demonstrate how these methods affect object storage in a HashSet.

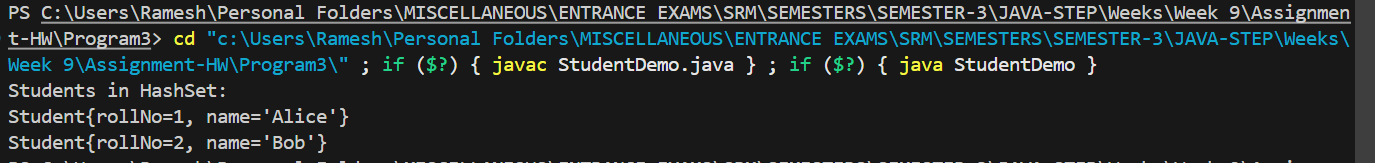
Hints:

● Use Objects.hash() to generate hash codes.

● Ensure equals() and hashCode() produce consistent results.

● Add duplicate objects to a HashSet and observe the output.

StudentDemo.java  

**OUTPUT🡪** ****

**Problem Statement 4:**

Create a class Library containing a list of Book objects. Implement cloning such that shallow

cloning only copies object references while deep cloning copies the entire list with individual

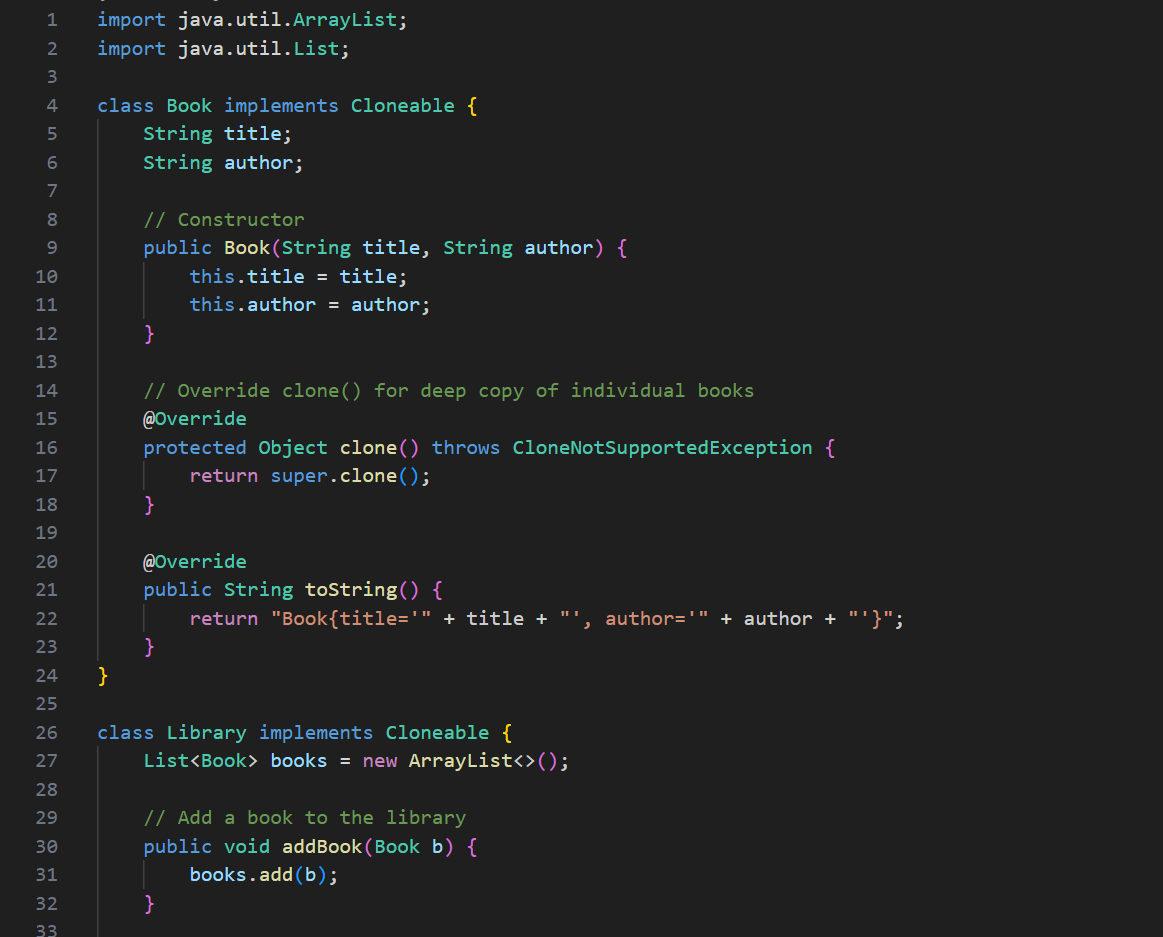
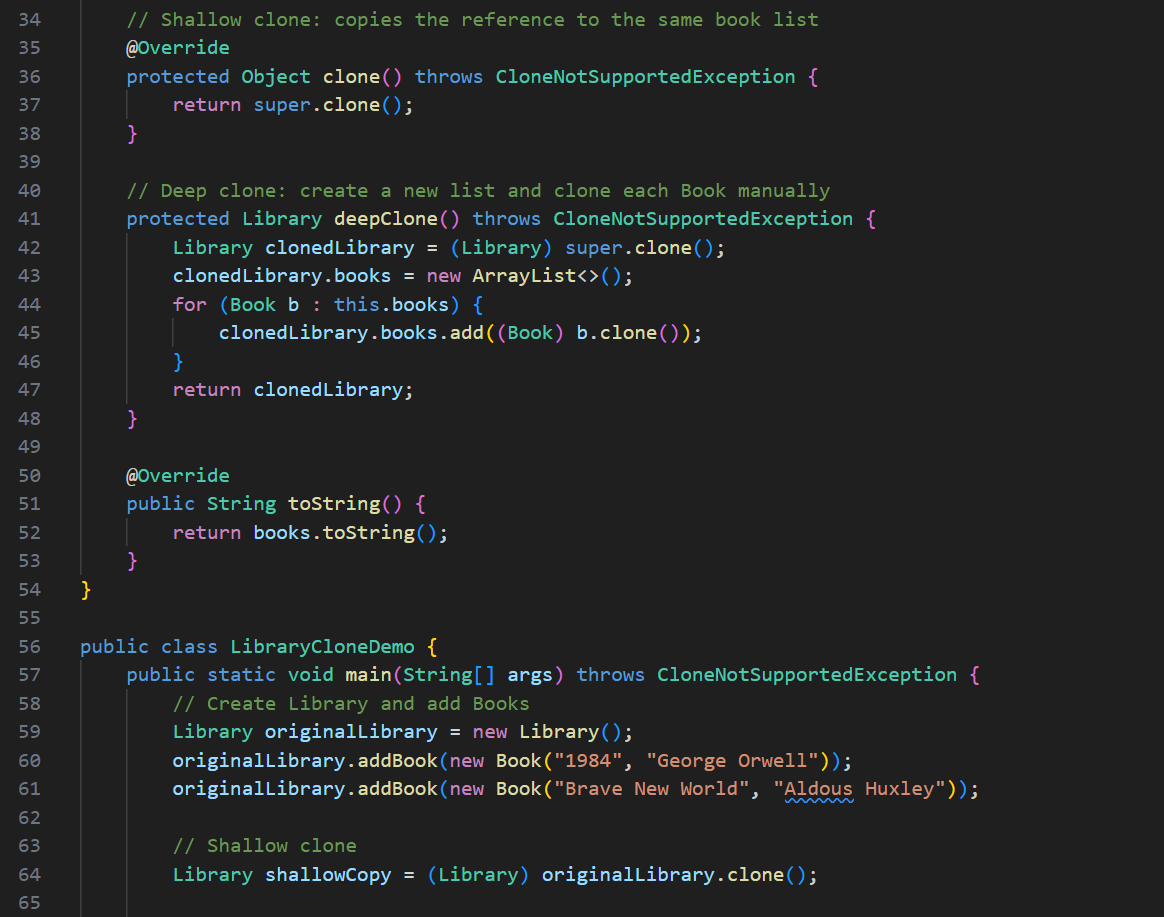
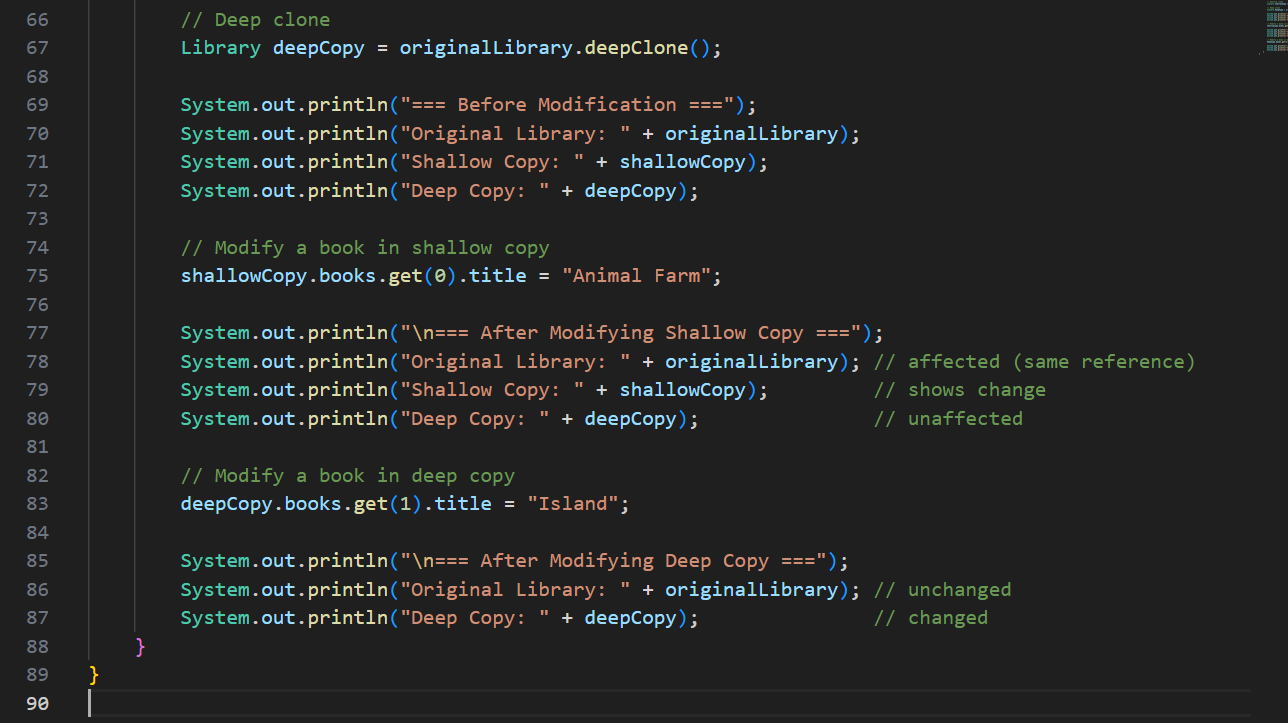
book data. Modify one book in the cloned object and observe its effect on the original.

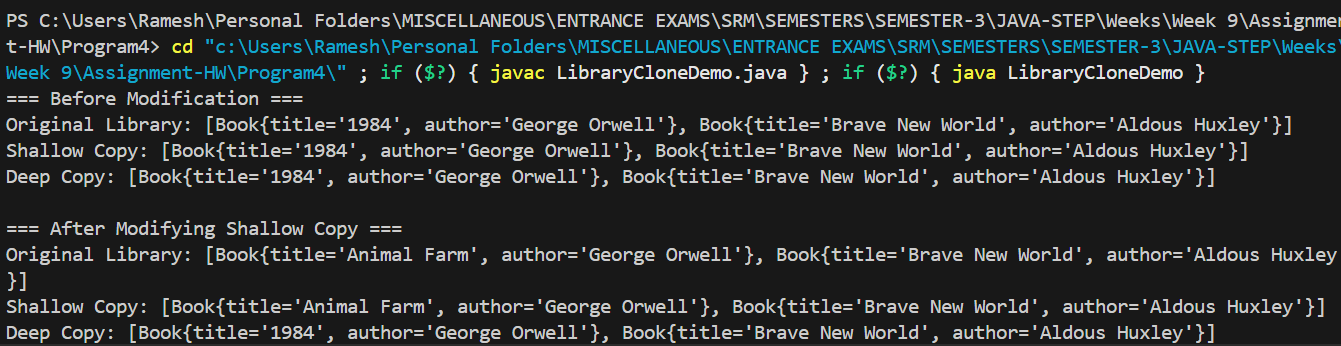
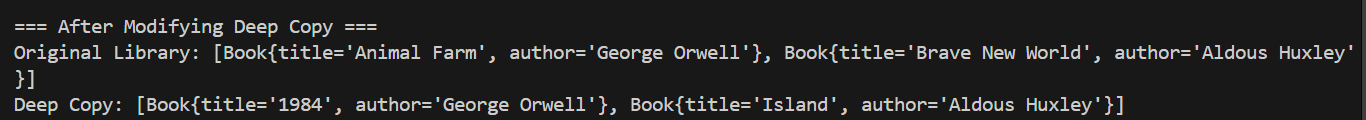
Hints:

● Use Cloneable interface and override clone().

● For deep cloning, clone each Book object inside the list manually.

● Use loops or streams to copy nested objects.

LibraryCloneDemo.java   

**OUTPUT🡪** **** 

**Problem Statement 5:**

Create an University class with a non-static inner class Department and a static nested

class ExamCell. The Department class should access outer class data, while the ExamCell

performs general exam operations. Demonstrate access of both inner types from the main

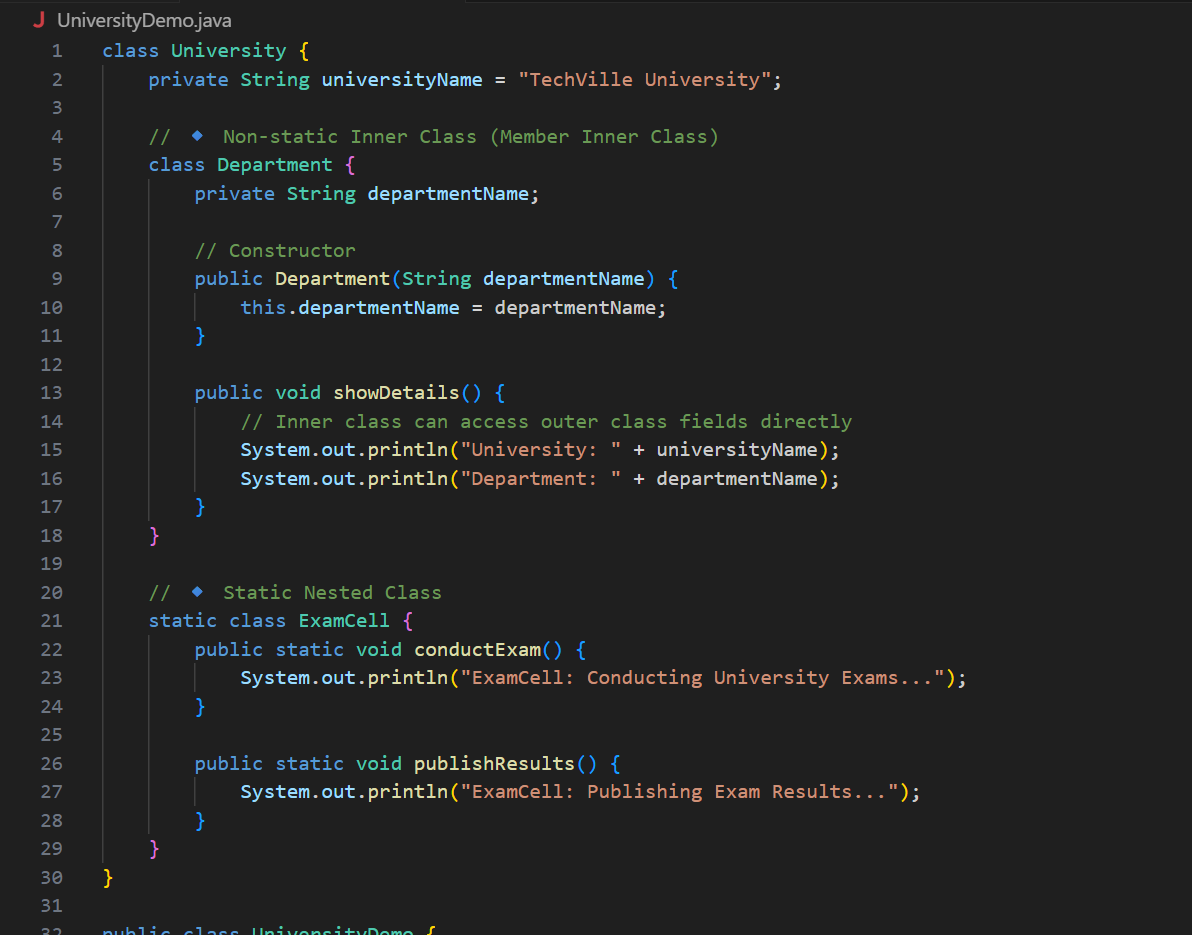
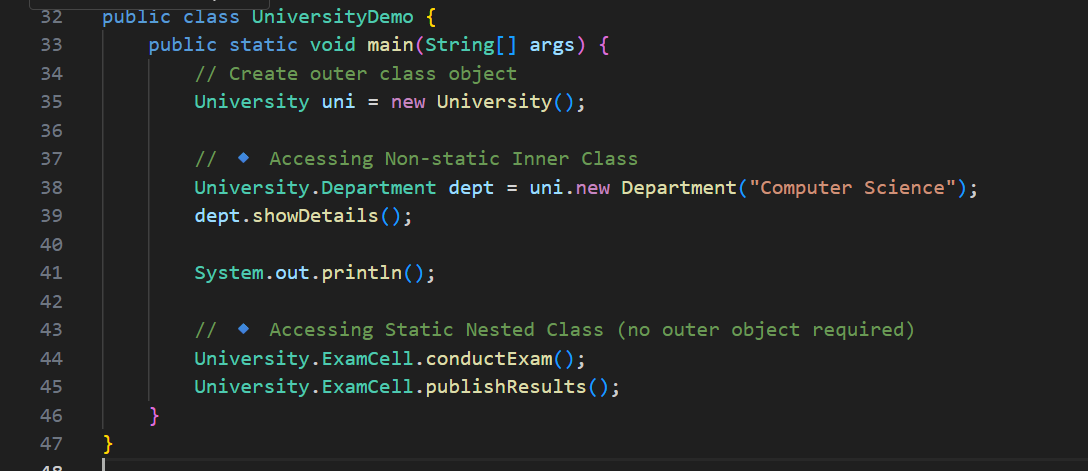
method.

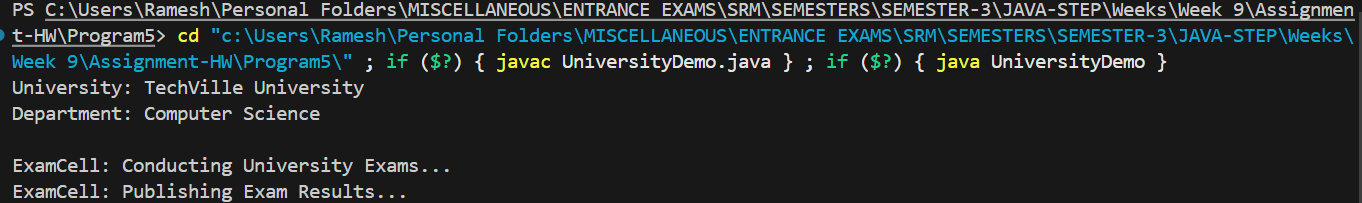
Hints:

● Use Outer.Inner syntax to create a member inner class object.

● Access outer class fields directly from member inner class.

● Use class name to access static nested class methods.

UniversityDemo.java  

**OUTPUT🡪** ****

**Problem Statement 6:**

Create a Payment class with a method processTransaction(). Inside it, define a local

inner class Validator that checks if payment amount is valid. Also, create an anonymous

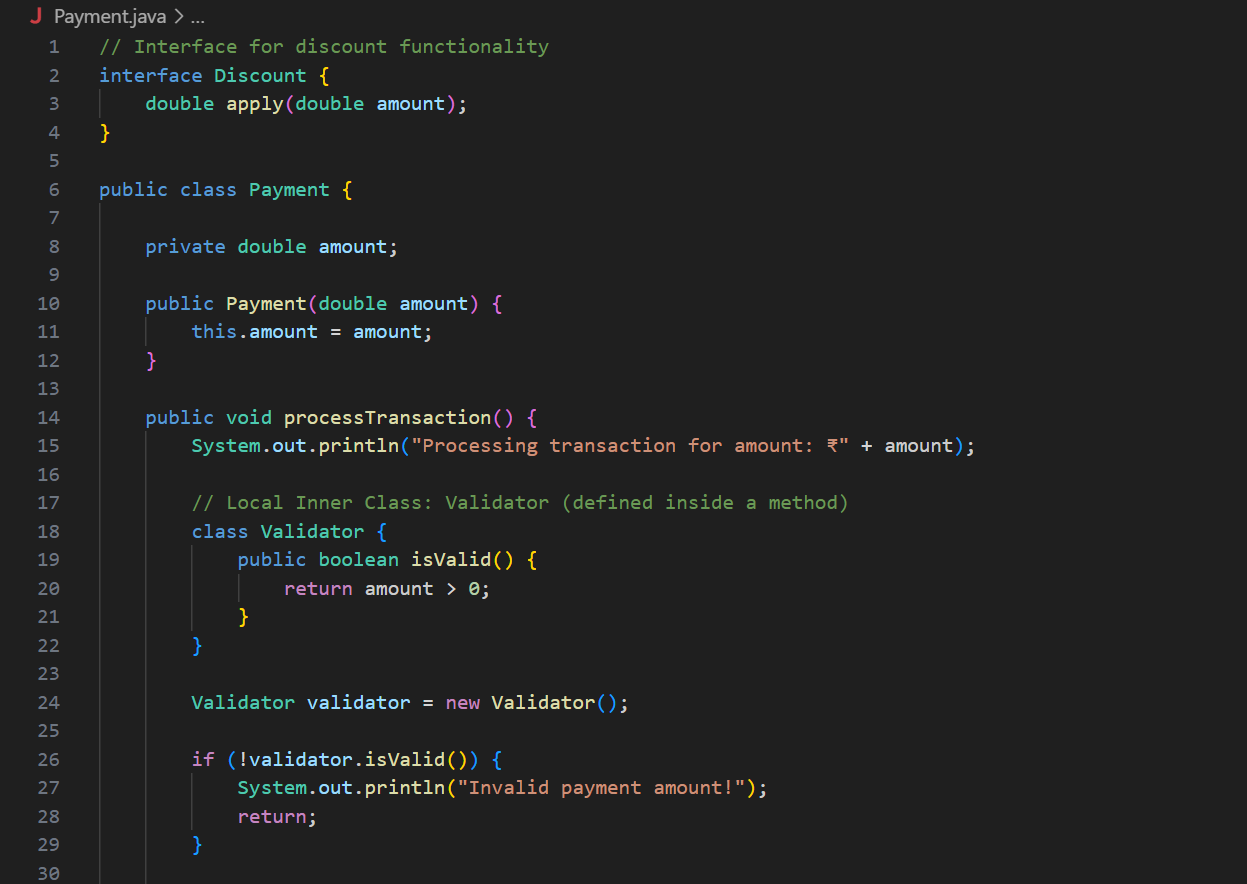
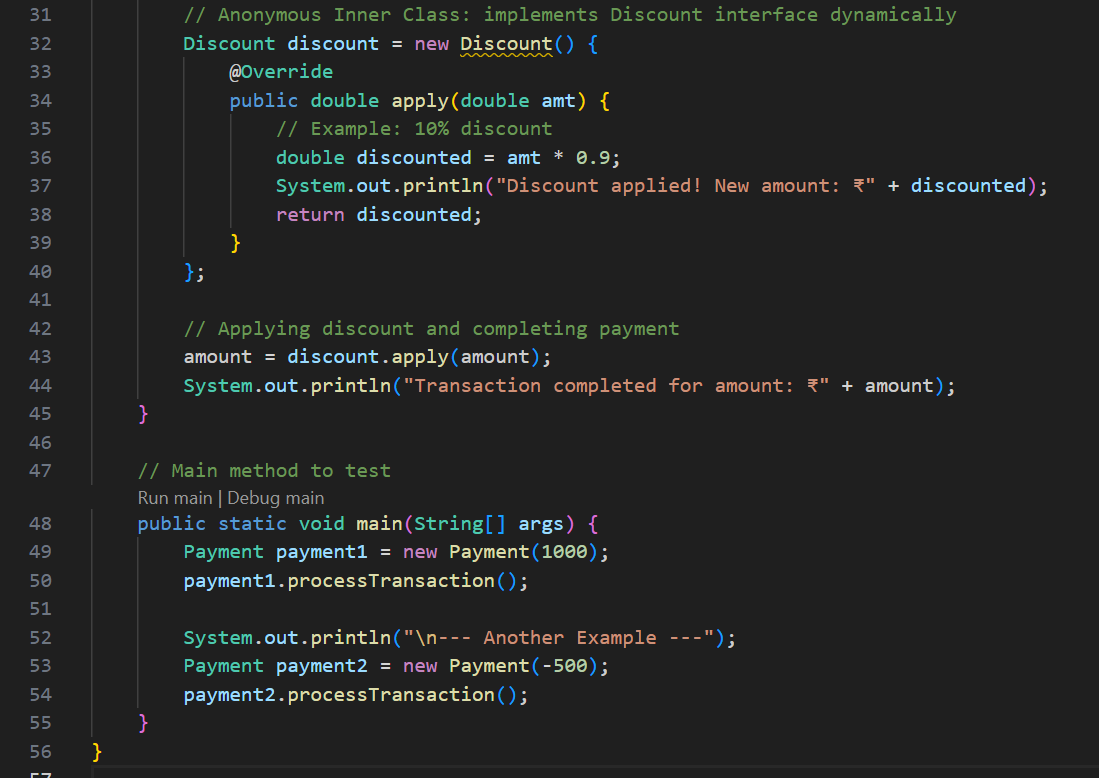
inner class implementing an interface Discount to apply discount dynamically.

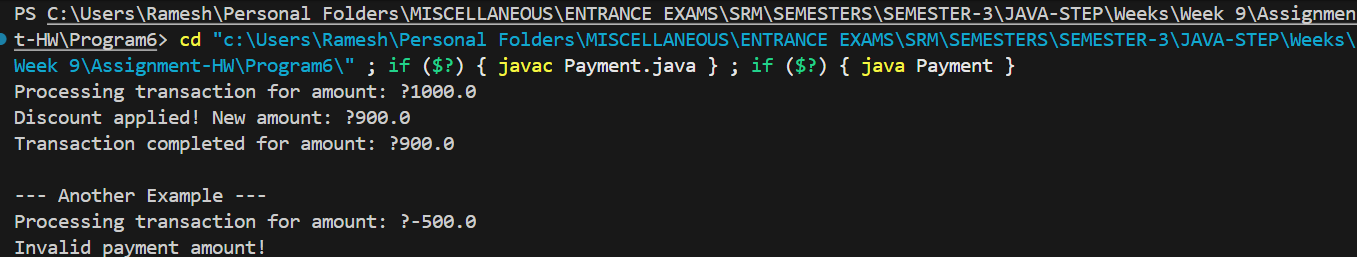
Hints:

● Define local inner class inside a method body.

● Use anonymous inner class for one-time interface implementation.

● Call methods of both classes inside processTransaction().

Payment.java  

**OUTPUT🡪** ****