Week 7 - S7 - Core OOP - Polymorphism - Lab Problem

Name:Ramesh Harisabapathi Chettiar

Date of Submission:24/09/25

**PROBLEM 1: Food Delivery App**

**Concept: Method Overloading**

**You're creating a food ordering system. Design a class that can calculate delivery**

**charges in different ways:**

**● Basic delivery (just distance)**

**● Premium delivery (distance + priority fee)**

**● Group delivery (distance + number of orders discount)**

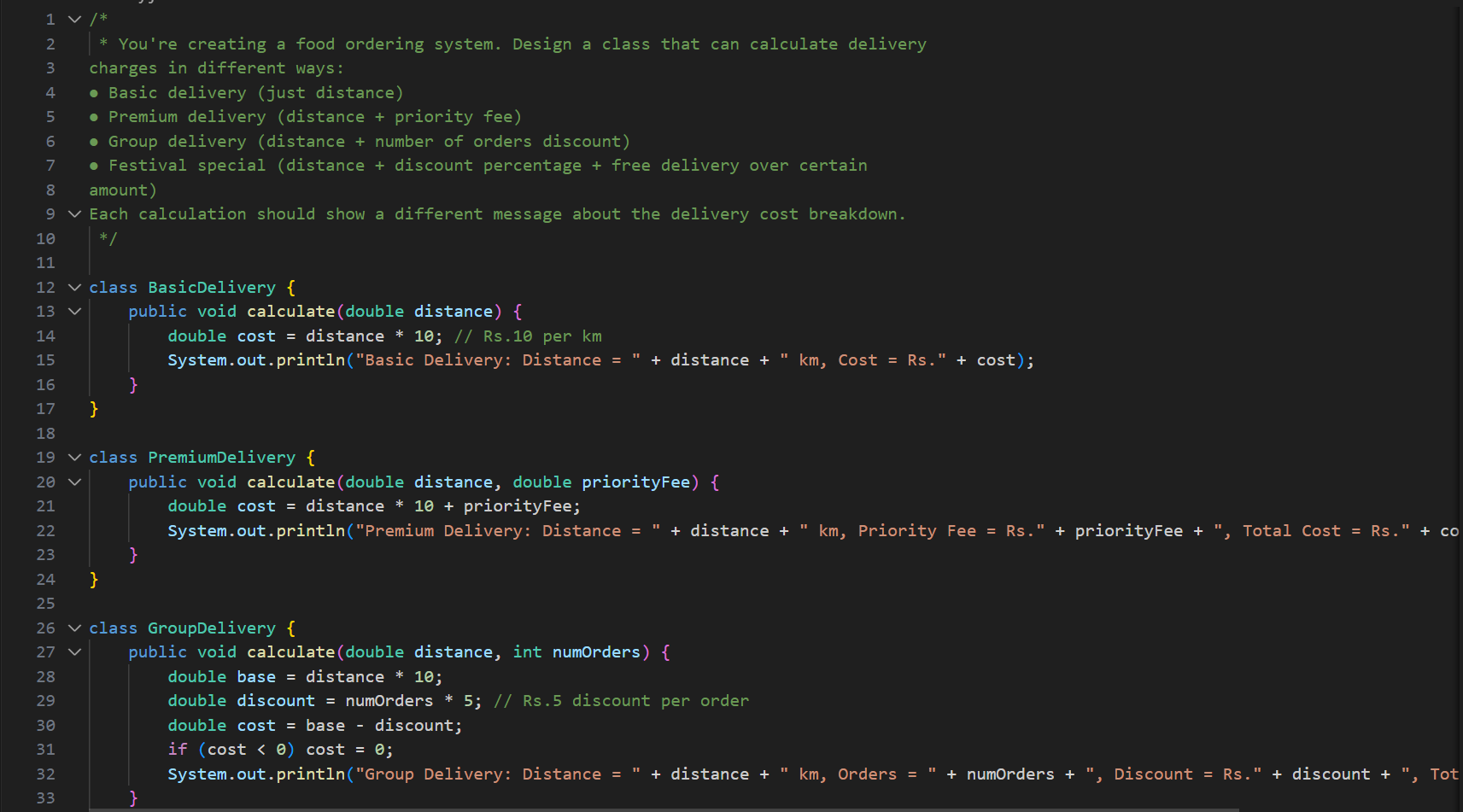
**● Festival special (distance + discount percentage + free delivery over certain**

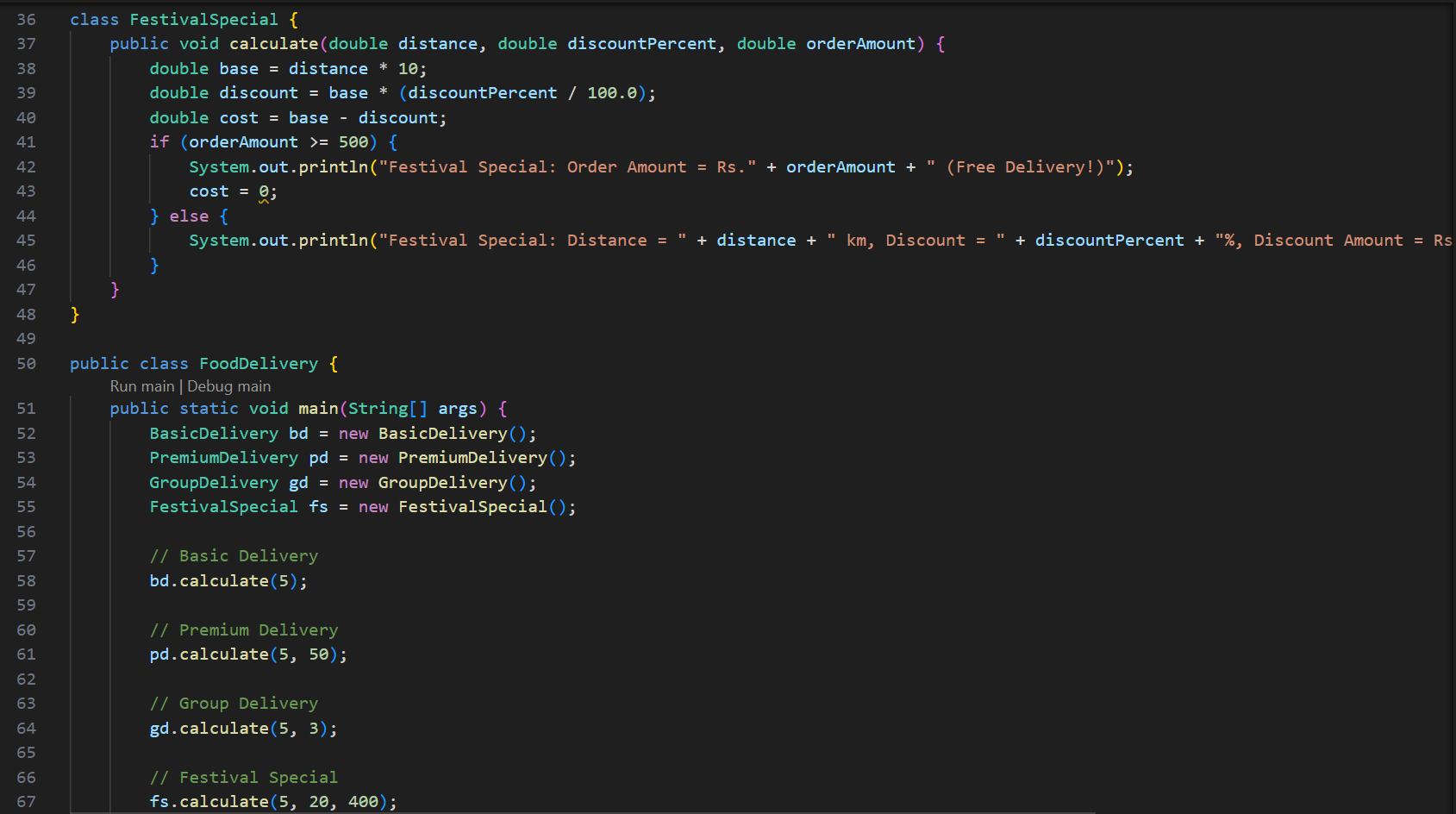
**amount)**

**Each calculation should show a different message about the delivery cost breakdown.**

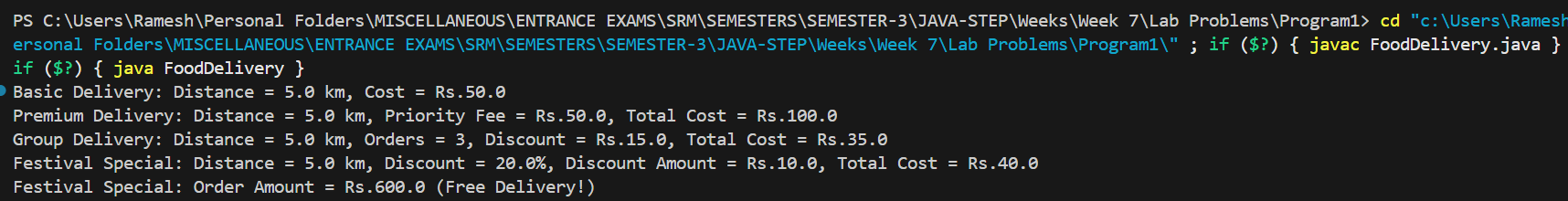
**Hint: Same method name, different parameters - let Java pick the right one!**

**PROGRAM🡪**

****

****

**OUTPUT🡪**

****

**PROBLEM 2: Social Media Feed**

**Concept: Method Overriding**

**Build a social media post system where different platforms display posts differently:**

**● Instagram posts show with hashtags and likes**

**● Twitter posts show with character count and retweets**

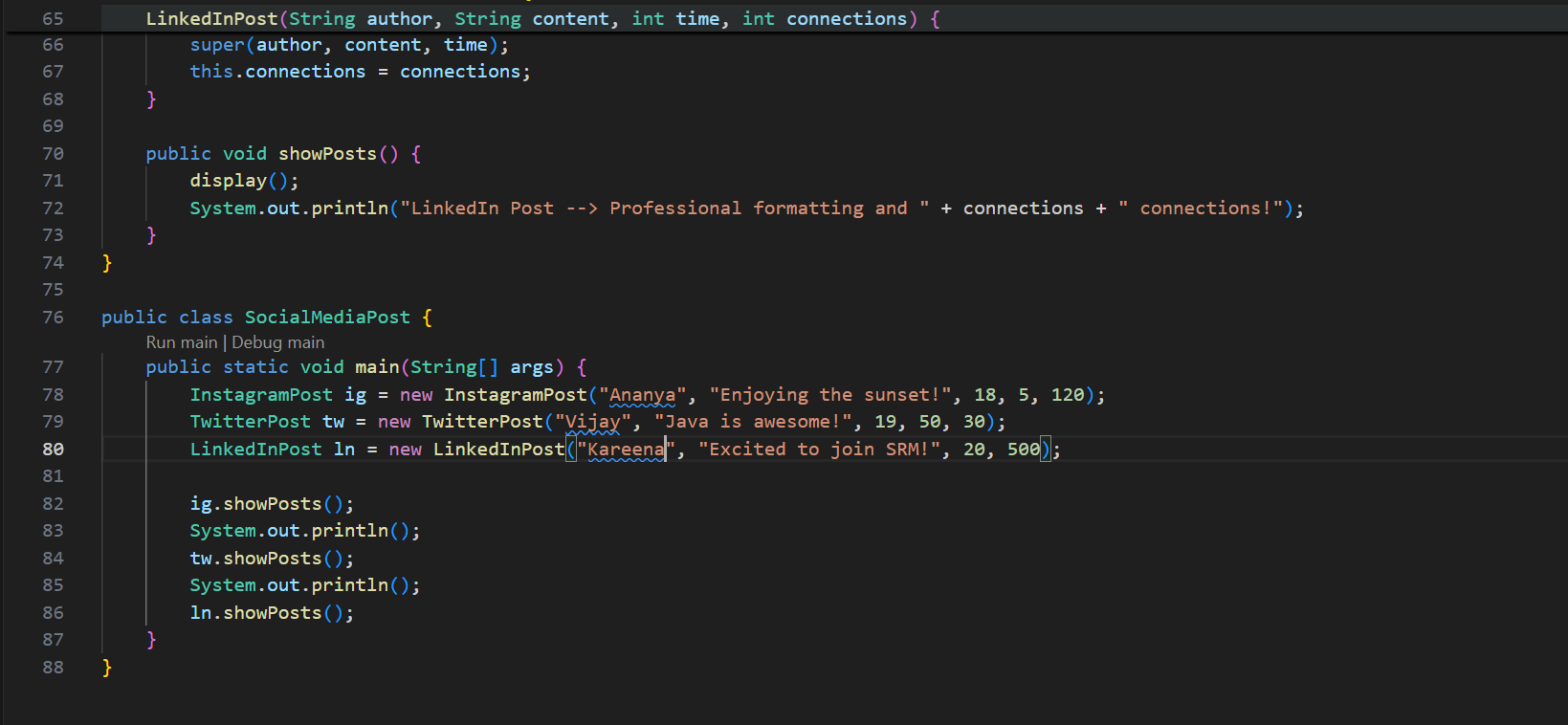
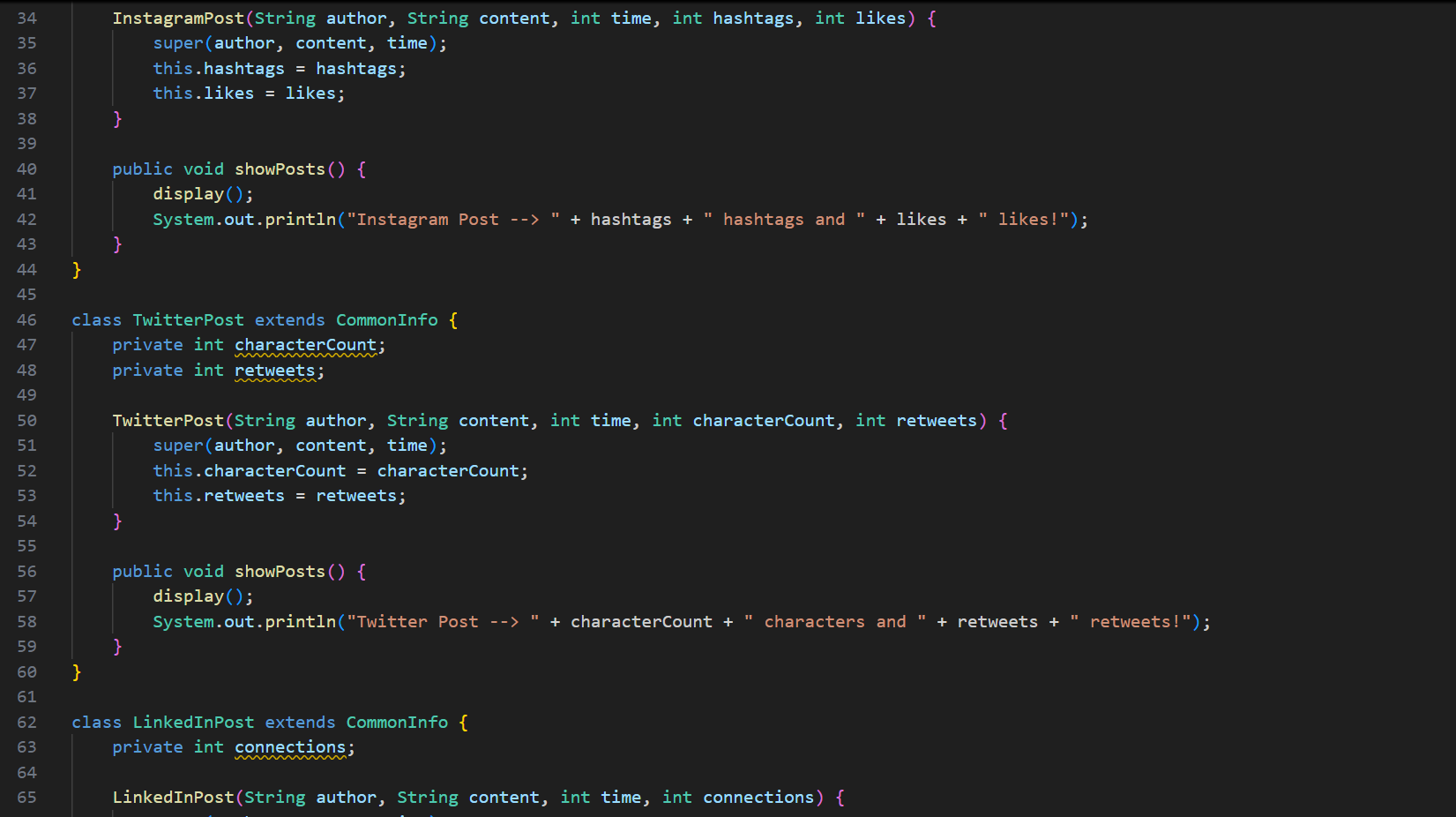
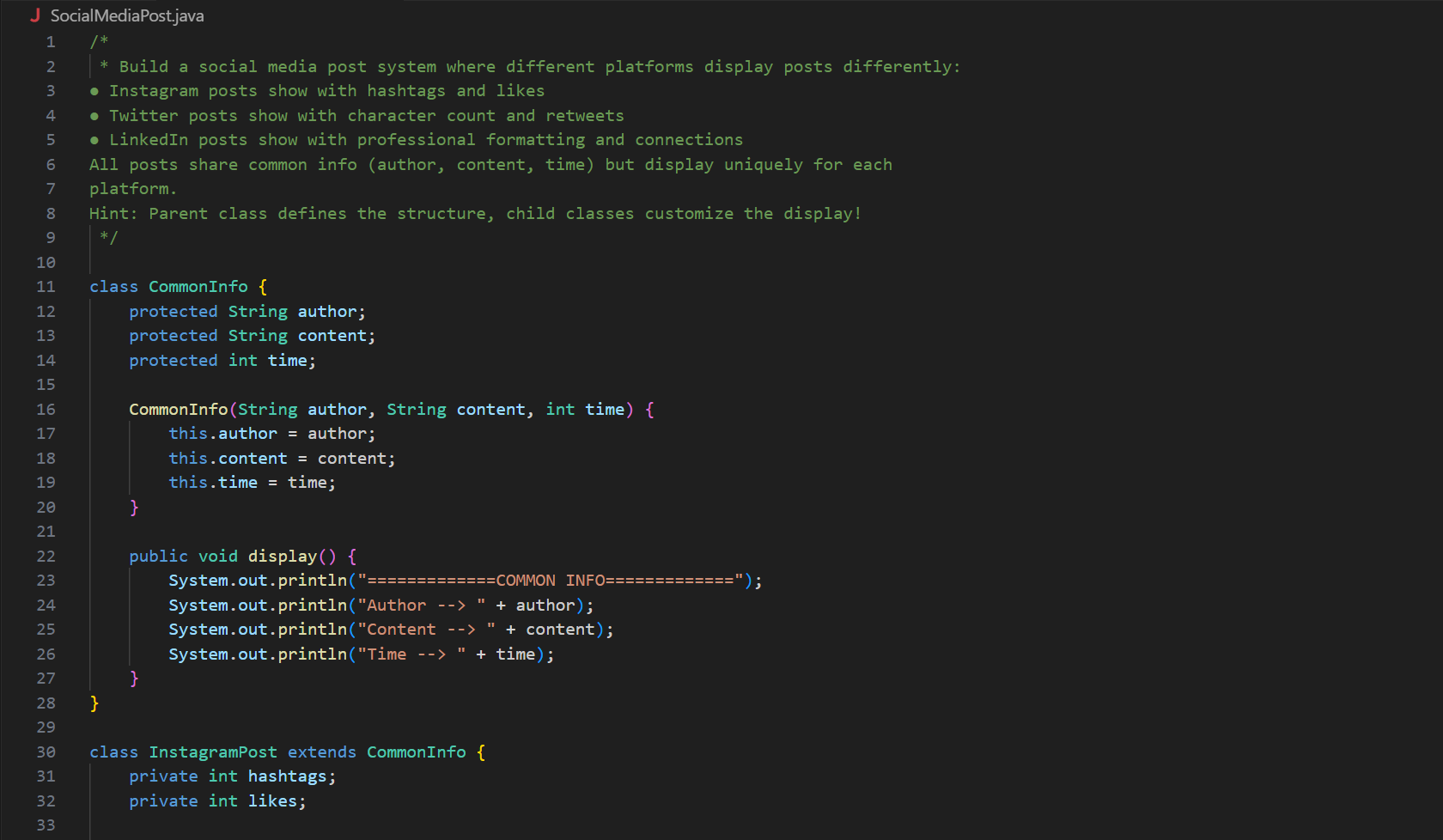
**● LinkedIn posts show with professional formatting and connections**

**All posts share common info (author, content, time) but display uniquely for each**

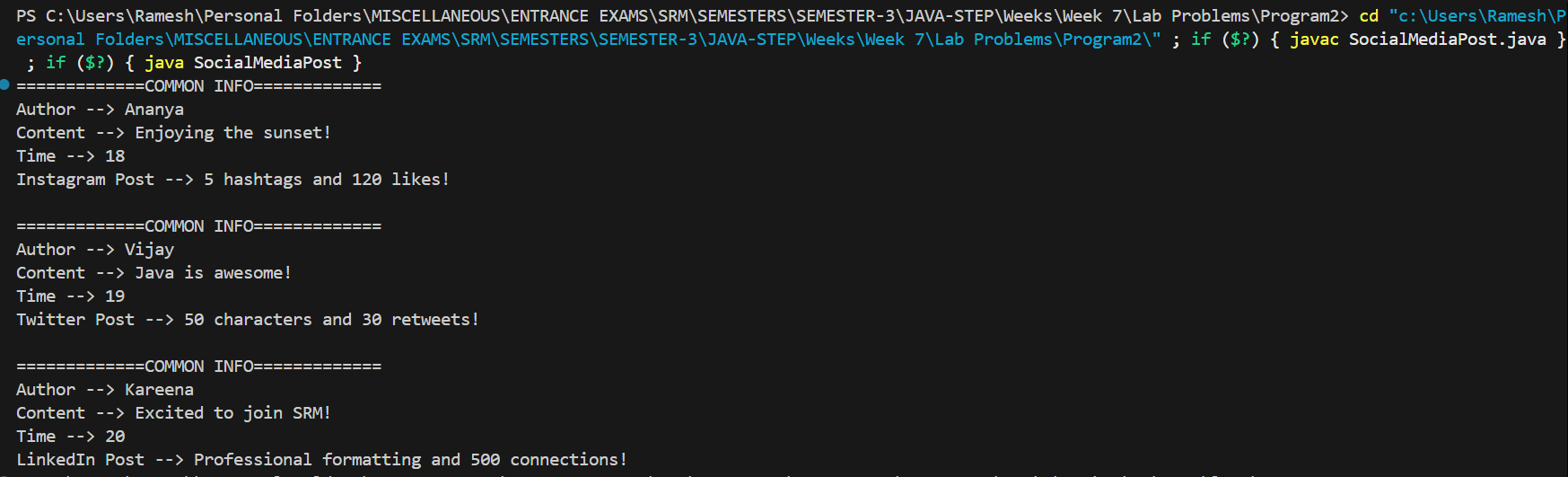
**platform.**

**Hint: Parent class defines the structure, child classes customize the display!**

**PROGRAM🡪**

****

**OUTPUT🡪**

****

**PROBLEM 3: Gaming Character System**

**Concept: Dynamic Method Dispatch**

**Create a battle system with different character types:**

**● Warriors attack with weapons and have high defense**

**● Mages cast spells and use mana**

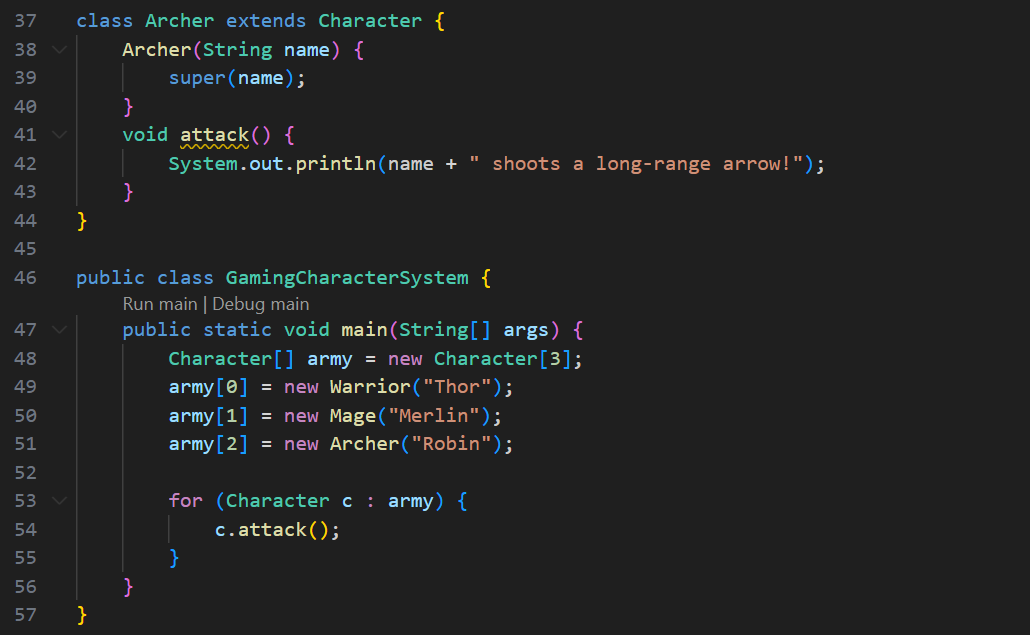
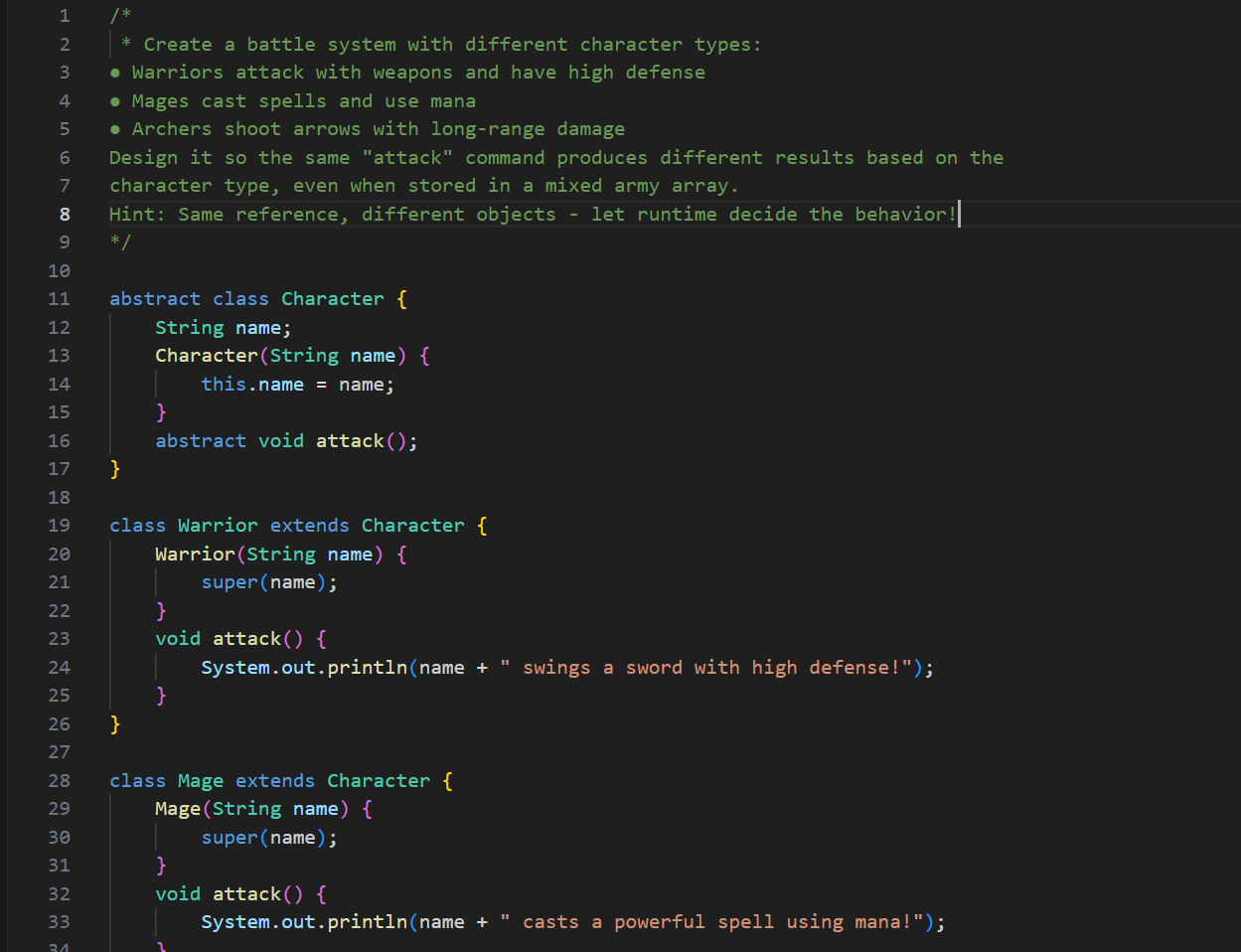
**● Archers shoot arrows with long-range damage**

**Design it so the same "attack" command produces different results based on the**

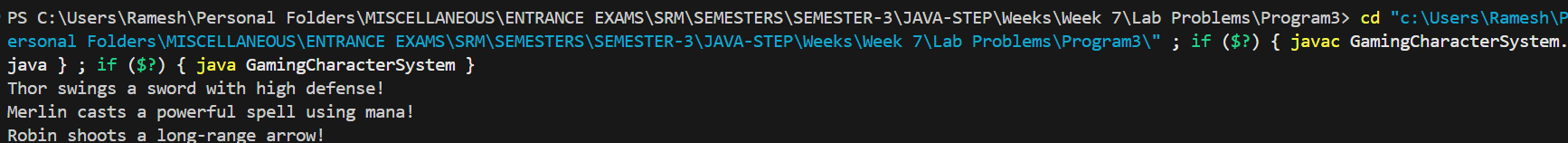
**character type, even when stored in a mixed army array.**

**Hint: Same reference, different objects - let runtime decide the behavior!**

**PROGRAM🡪**

****

**OUTPUT🡪**

****

**PROBLEM 4: University Library System**

**Concept: Upcasting**

**Design a library system with different types of users:**

**● Students can borrow books and access computers**

**● Faculty can reserve books and access research databases**

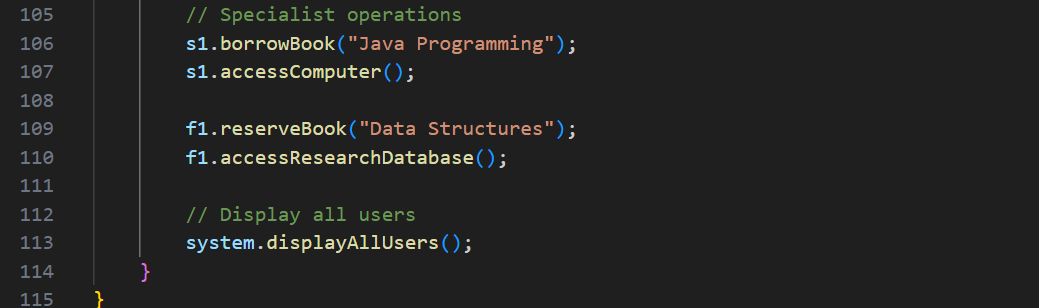
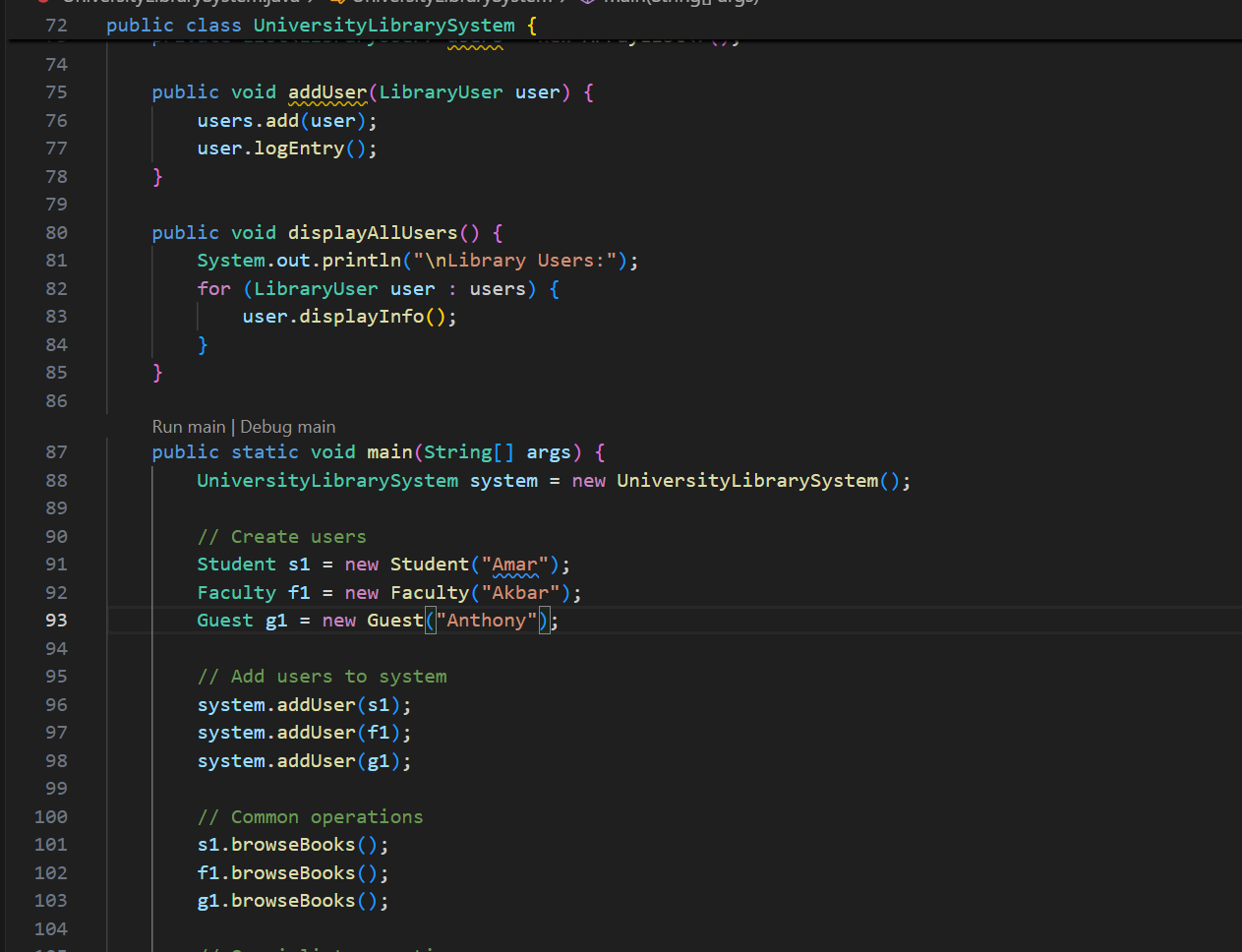
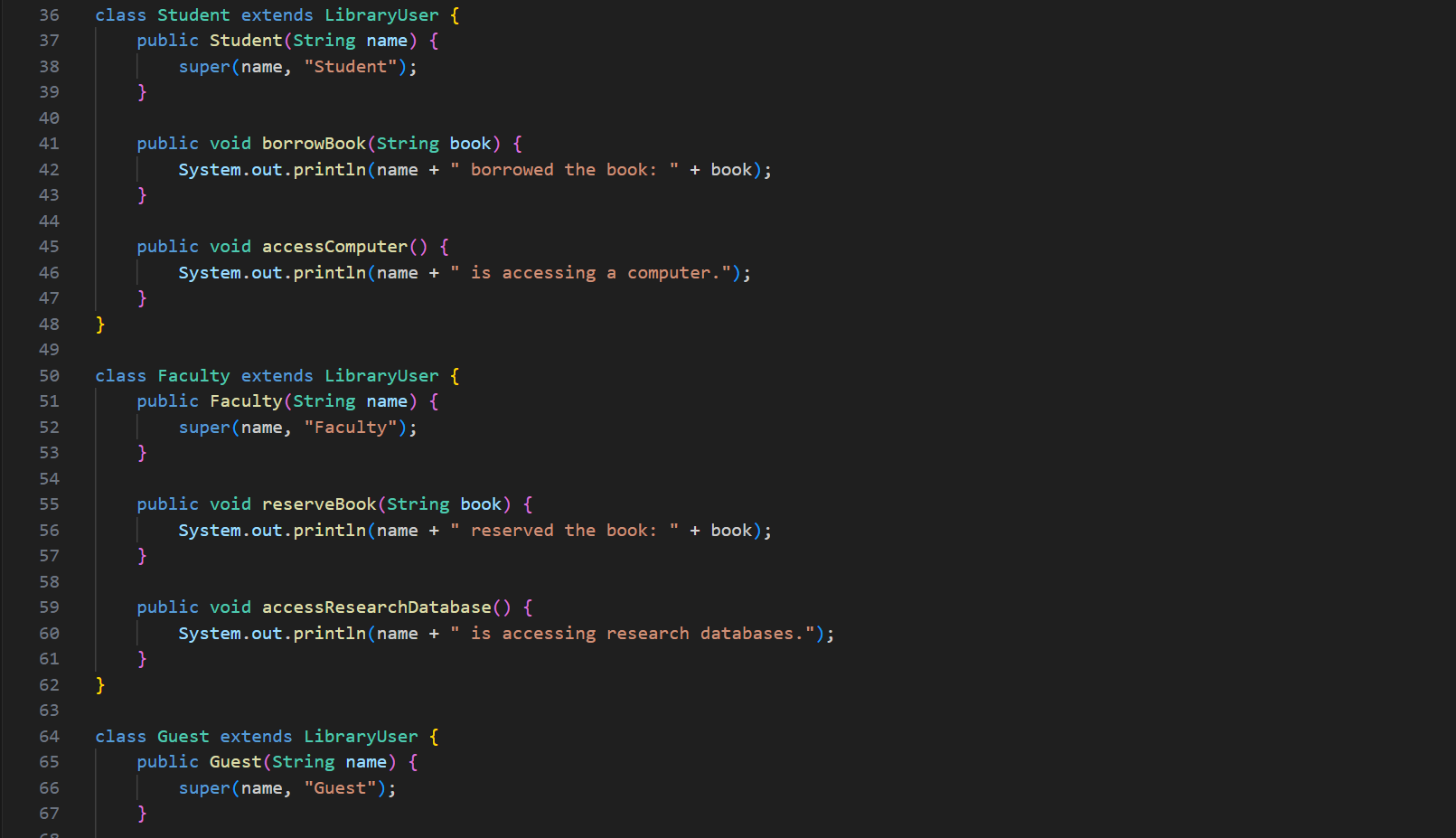
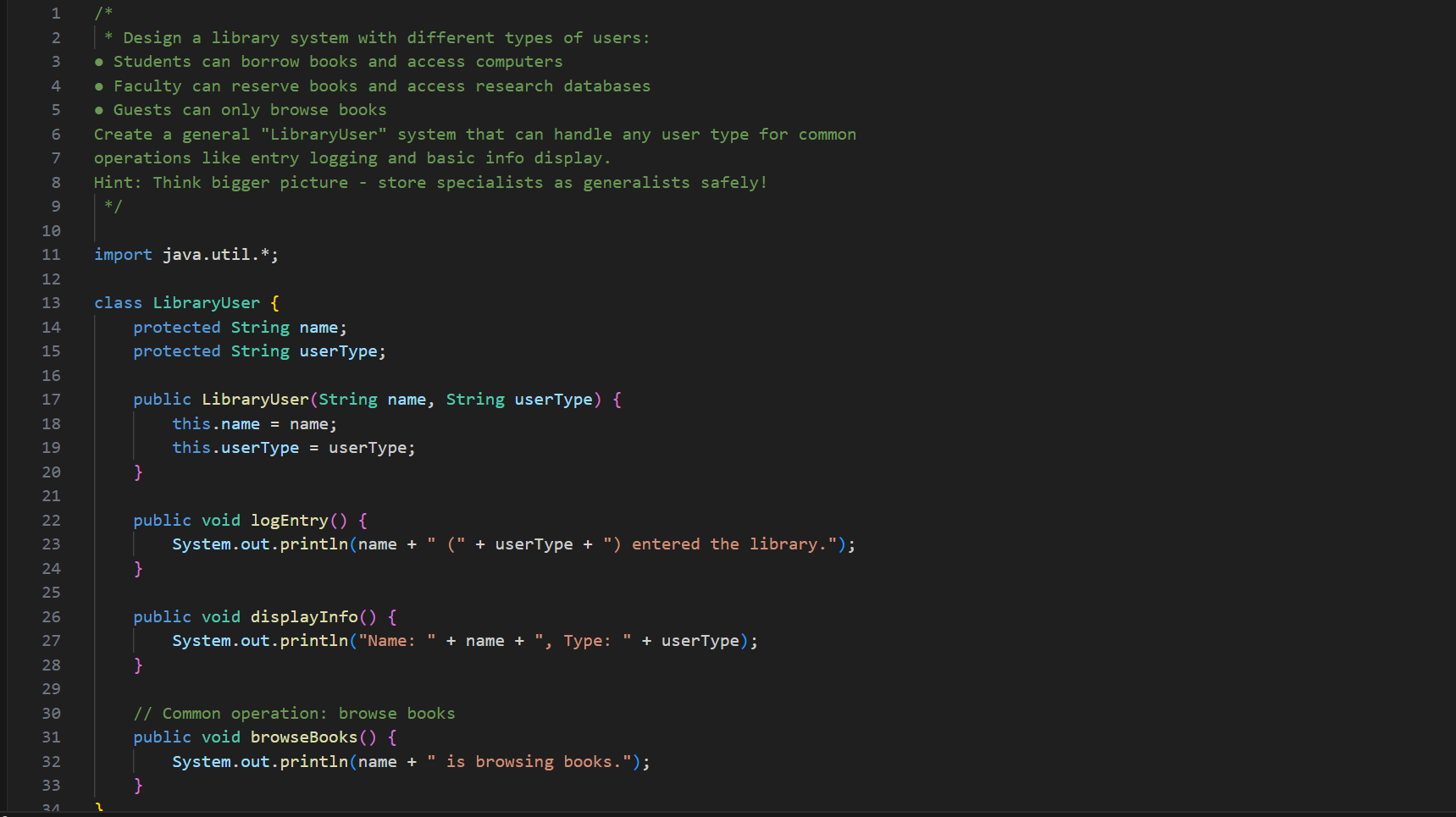
**● Guests can only browse books**

**Create a general "LibraryUser" system that can handle any user type for common**

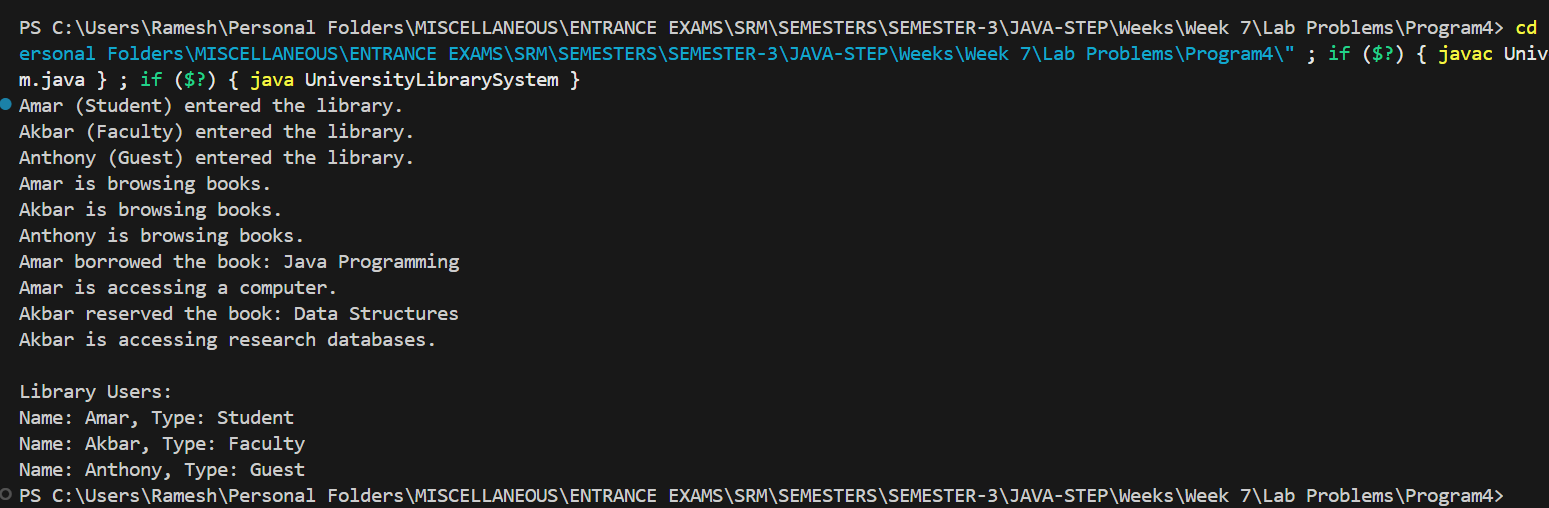
**operations like entry logging and basic info display.**

**Hint: Think bigger picture - store specialists as generalists safely!**

**PROGRAM🡪**

****

**OUTPUT🡪**

****

**PROBLEM 5: Movie Streaming Platform**

**Concept: Downcasting**

**Build a streaming service that handles different content types:**

**● Movies have ratings, duration, and subtitle options**

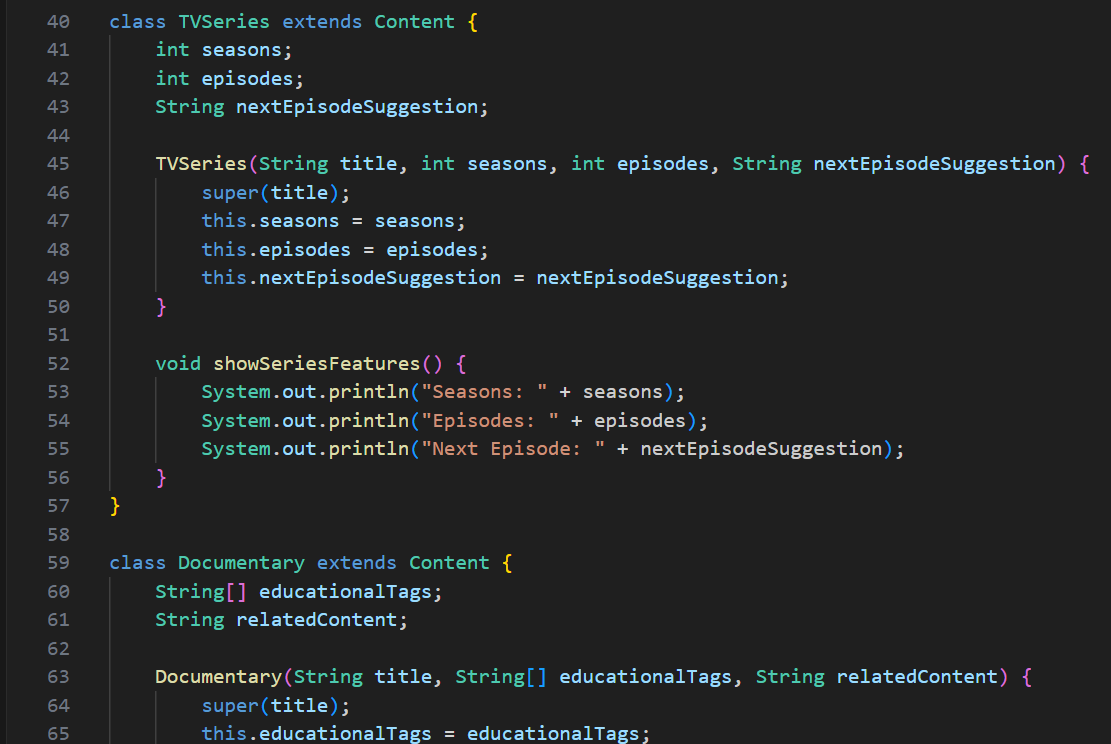
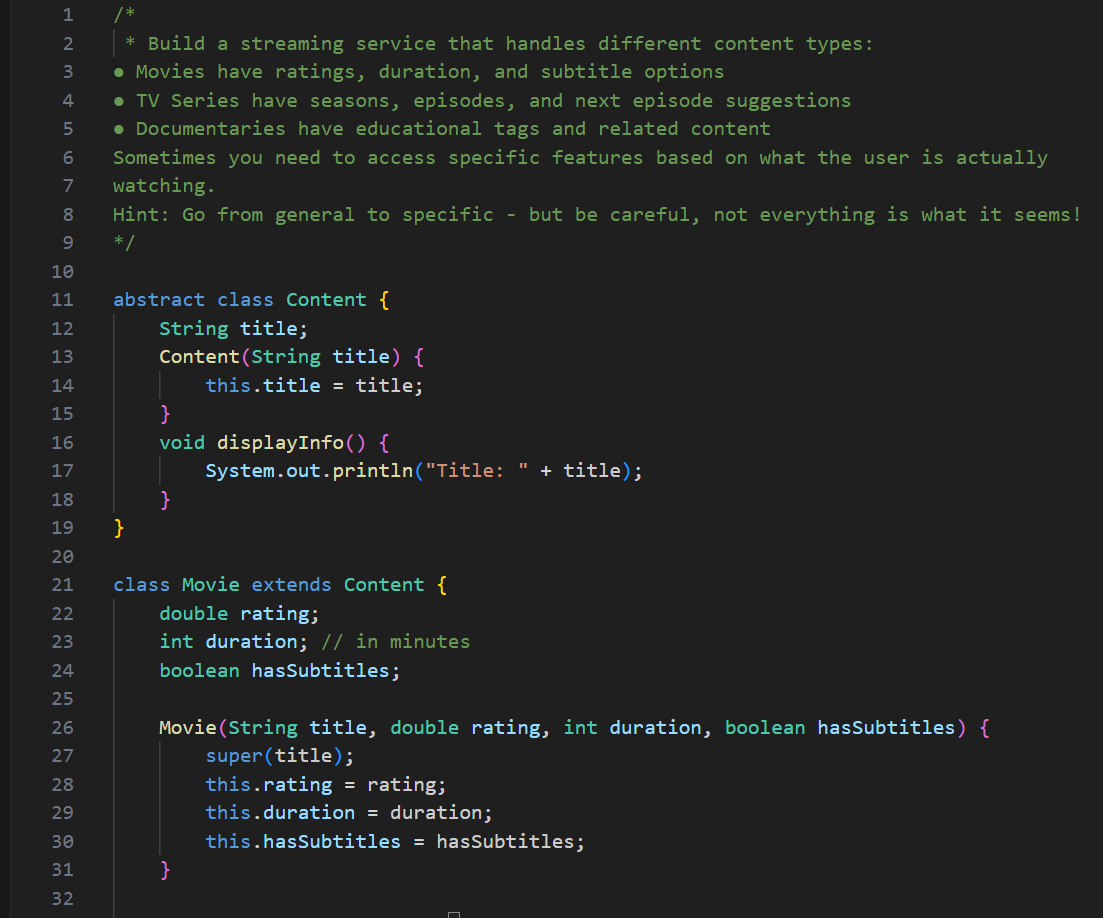
**● TV Series have seasons, episodes, and next episode suggestions**

**● Documentaries have educational tags and related content**

**Sometimes you need to access specific features based on what the user is actually**

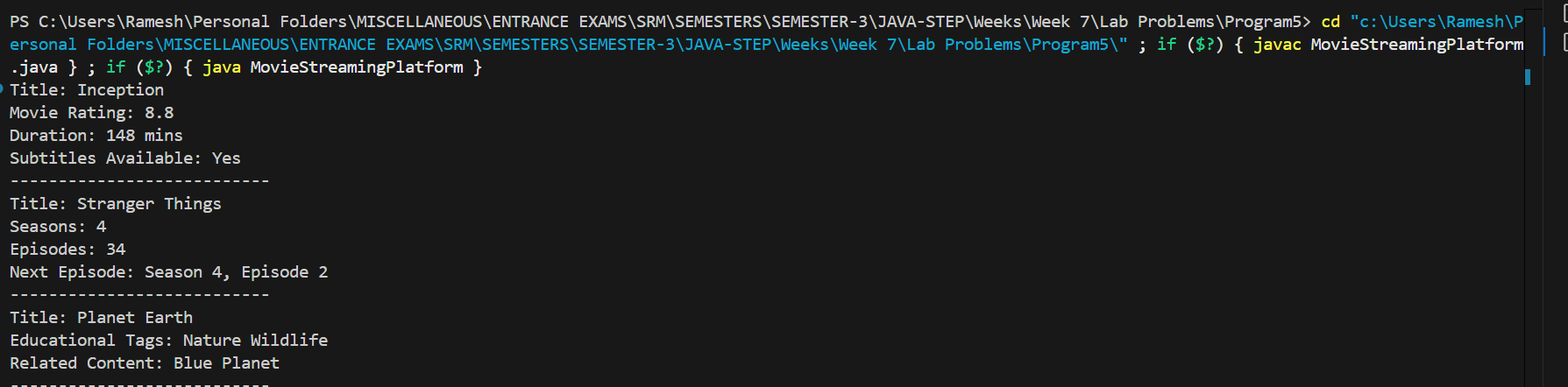
**watching.**

**Hint: Go from general to specific - but be careful, not everything is what it seems!  
PROGRAM🡪**

****

****

**OUTPUT🡪**

****

**PROBLEM 6: Smart Campus IoT System**

**Concept: Safe Downcasting with instanceof**

**Create a campus management system with different smart devices:**

**● Smart classrooms control lighting, AC, and projectors**

**● Smart labs manage equipment and safety systems**

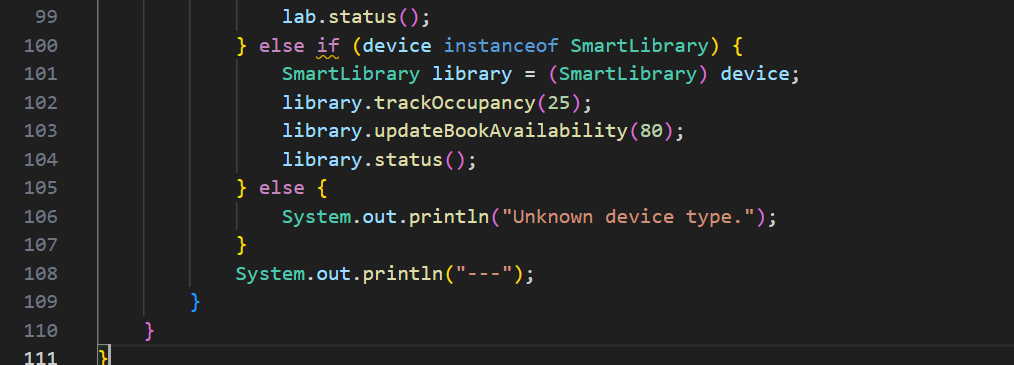
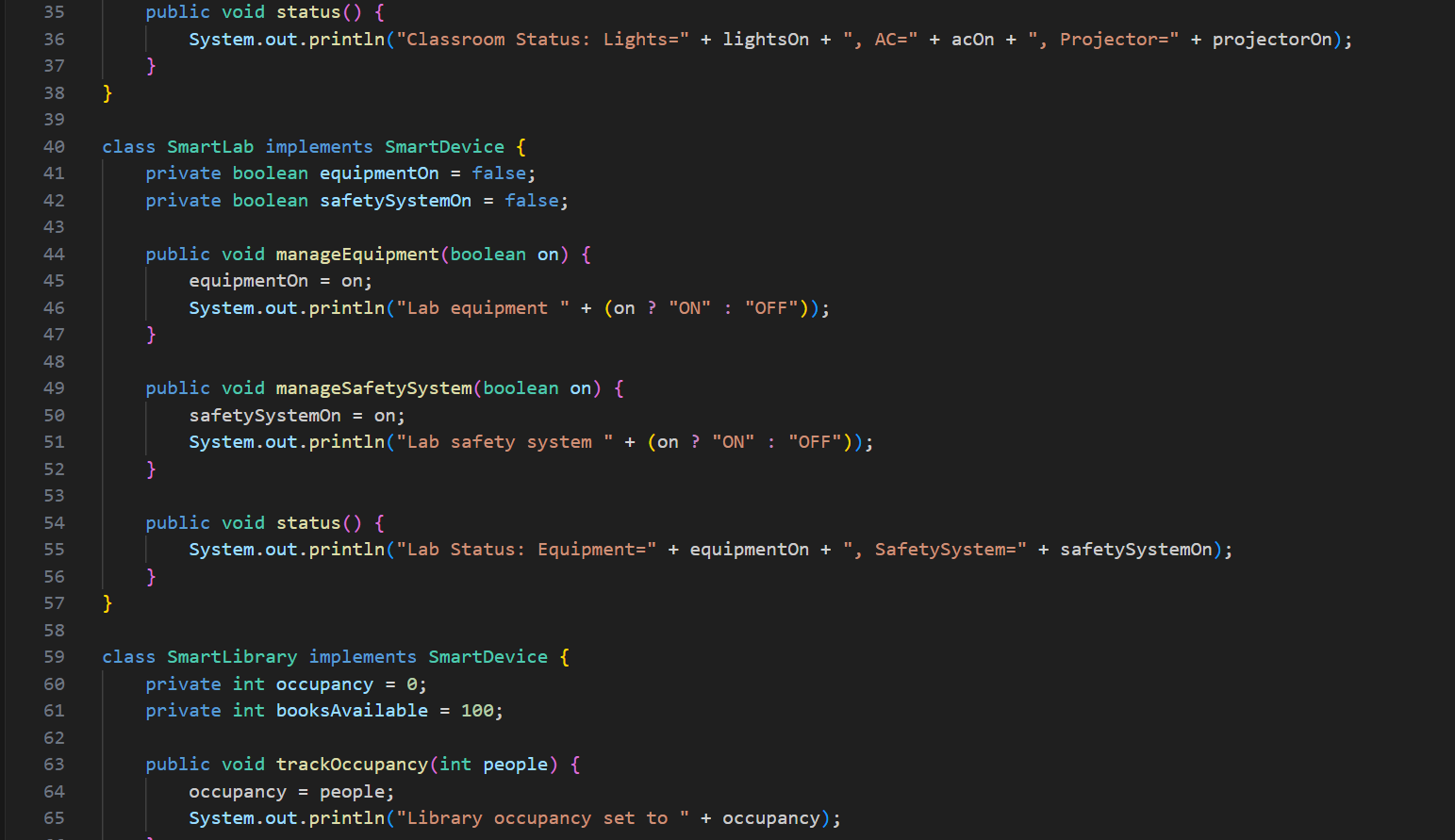
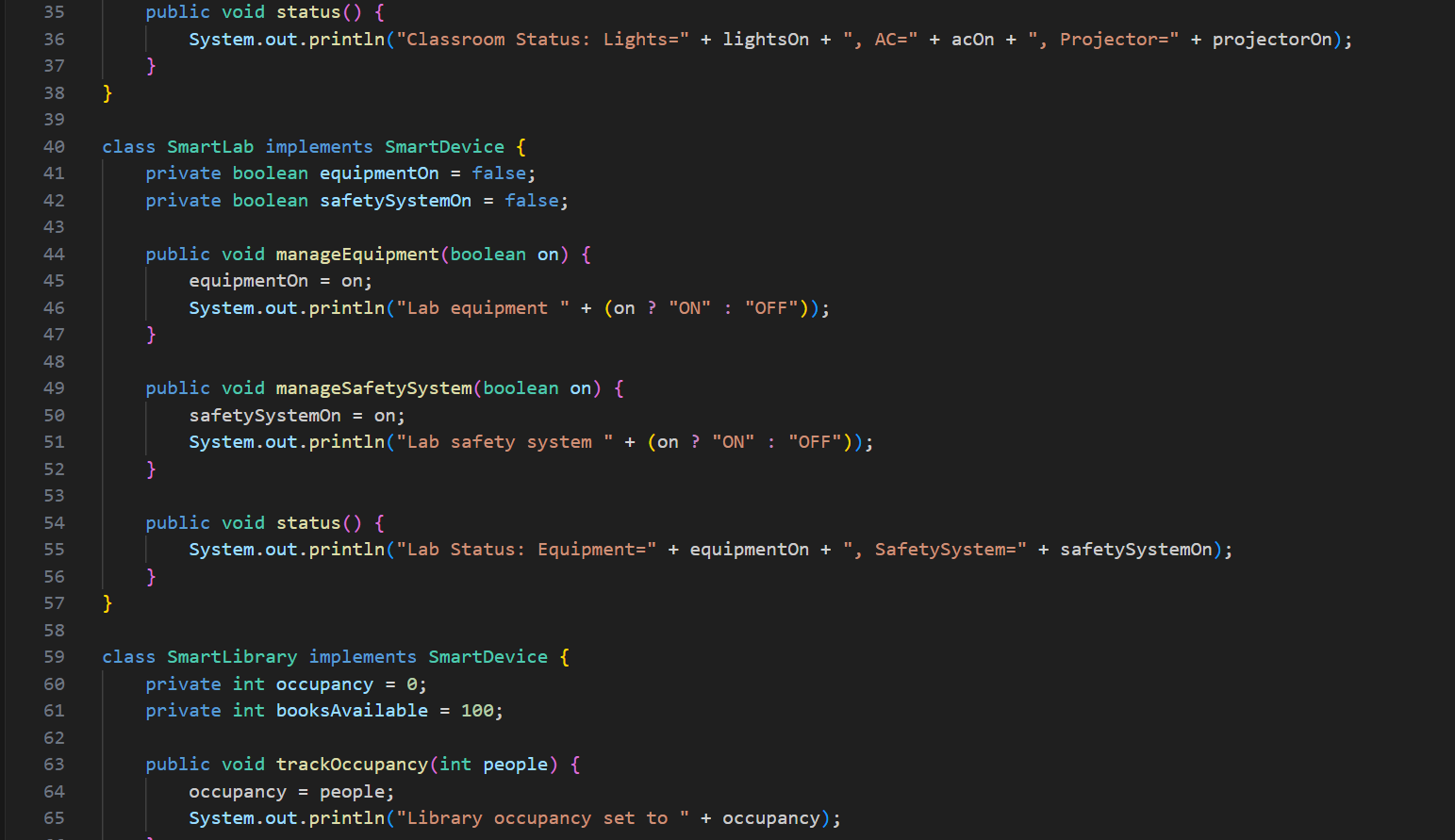
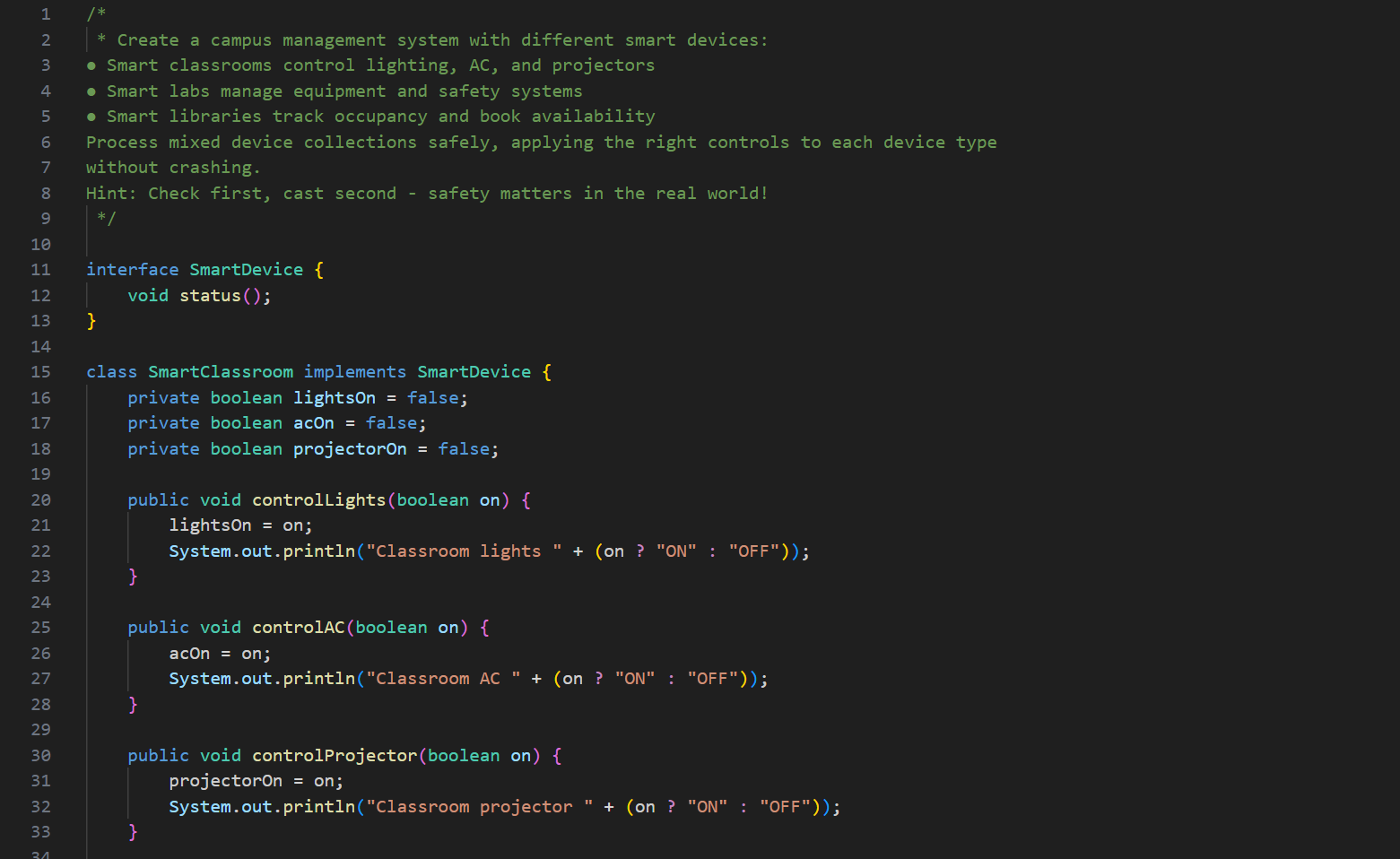
**● Smart libraries track occupancy and book availability**

**Process mixed device collections safely, applying the right controls to each device type**

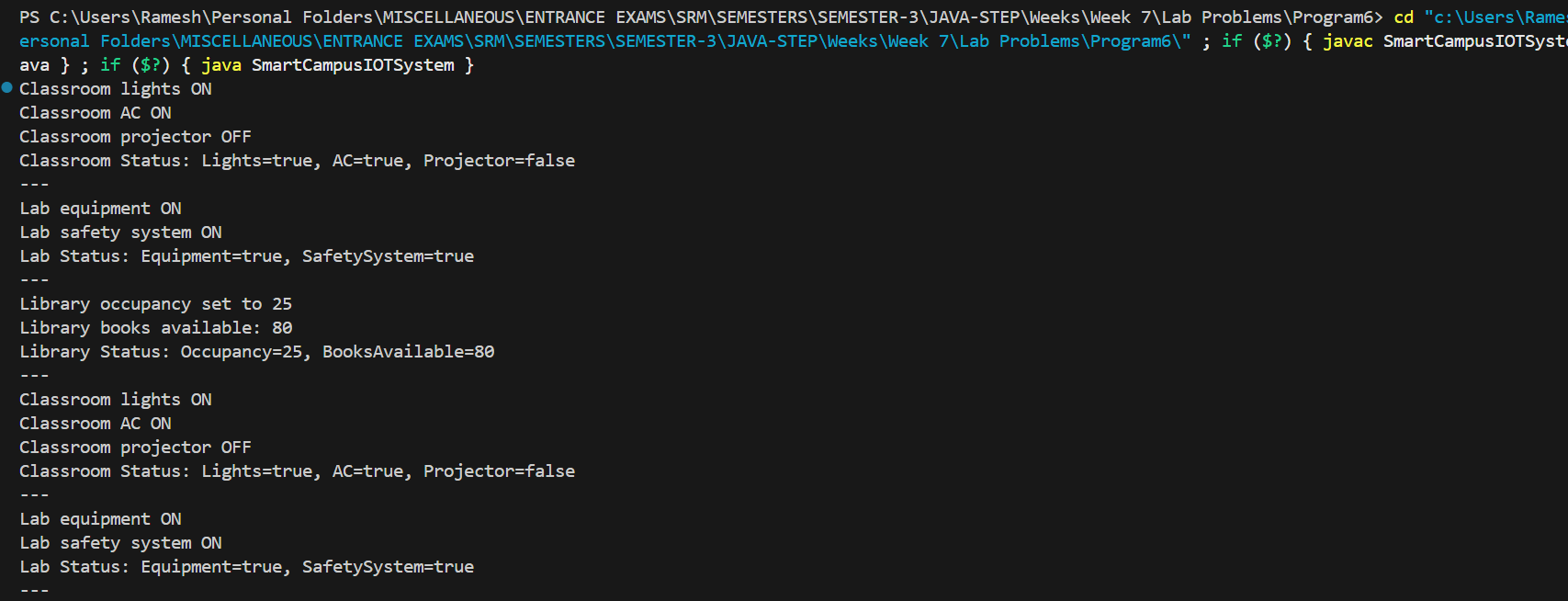
**without crashing.**

**Hint: Check first, cast second - safety matters in the real world!**

**PROGRAM🡪**

****

**OUTPUT🡪**

****