SE116 – LAB#7 2024-2025 SPRING

Aim: Exceptions

In this lab, you will create a **library management system** that demonstrates how to use **seven custom exceptions** to handle a variety of error conditions in Java. You will write classes that represent **members**, **items**, and a **system** that manages them, while also creating **seven distinct exception types**, four that extend `Exception` (checked exceptions) and three that extend `RuntimeException` (unchecked exceptions). Through this project, you will practice how to **throw** exceptions at the point of error detection and **catch** them where recovery or error handling must occur, ensuring a robust and maintainable codebase.

You must begin by defining seven separate Java classes for your custom exceptions**.** You will create `UserNotFoundException`, `ItemNotFoundException`, `DuplicateMemberException`, and `DuplicateItemException` as checked exceptions by extending `Exception`. You will also create `OverLimitException`, `InvalidMemberNameException`, and `InvalidItemTitleException` as unchecked exceptions by extending `RuntimeException`. Each exception class will have a constructor that accepts a `String message` and passes it to `super(message)`, allowing you to produce meaningful error descriptions.

After defining these exceptions, you will write a class called `LibraryMember`, which represents a user of the library. It will include private variables for the `String name`, an `int memberId`, an `int borrowedCount`, and an `int borrowLimit`. The `borrowLimit` can default to a value such as 5, indicating how many items the user may borrow at once. If the constructor or any setter method detects an empty or invalid name, it will throw the unchecked exception `InvalidMemberNameException`. By performing these checks, you ensure that no `LibraryMember` is created or modified without a proper name.

Next, you will create a class called `LibraryItem` to represent an individual cataloged item in the library. It will have private variables for a `String title`, a `String itemId`, and a `boolean isBorrowed`, defaulting to `false`. If someone tries to construct a `LibraryItem` with an empty or invalid title, it will throw an InvalidItemTitleException. The class will provide methods to mark an item as borrowed or returned, and a way to display its current information.

After defining these data classes, you will design a `LibrarySystem` class that manages two `ArrayList`s, one containing `LibraryMember` objects and another containing `LibraryItem` objects. You will add a method `registerMember` that checks for duplicate member IDs and throws `DuplicateMemberException` if you try to register a member whose ID already exists. Similarly, a method `addNewItem` will throw a DuplicateItemException if you try to add an item with a duplicate item ID. Two search methods, `findMemberById` and `findItemById`, will either return the matching object or `null` if none is found. You will write a `borrowItem` method that looks up both a user and an item, throwing `UserNotFoundException` and `ItemNotFoundException` if either is missing. If the user's borrowed count is already at or above their borrow limit, the method will throw an OverLimitException, which is unchecked. Another method, `returnItem`, will also declare `throws UserNotFoundException, ItemNotFoundException`, since it similarly must ensure that the user and item exist. This

method will allow you to mark items as returned and decrement the user's borrowed count. Additional helper methods such as `printAllMembers` and `printAllItems` will aid in debugging and demonstration.

Finally, you will write a `Main` class to show how your system and exceptions interact. In the `main` method, you will instantiate a `LibrarySystem`, register a few members, and add some items to illustrate various operations. You will trigger exceptions by, for example, attempting to register the same member twice to cause a `DuplicateMemberException`, or trying to add a new `LibraryItem` with the same ID to cause a `DuplicateItemException`. You will also make calls to `borrowItem` with invalid user IDs or item IDs to throw `UserNotFoundException` or `ItemNotFoundException`, and attempt to exceed the user's borrow limit to trigger `OverLimitException`. Each scenario will be enclosed in a `try-catch` block to handle the thrown exceptions. You will also create at least one `LibraryMember` and one `LibraryItem` with empty names or titles, thus throwing the `InvalidMemberNameException` or `InvalidItemTitleException`.