Coding Task 5

Exception

---

5. Create the exceptions in package myexceptions and create the following custom exceptions and throw them in methods whenever needed. Handle all the exceptions in main method,

- **UserNotFoundException**: throw this exception when user enters an invalid user id which doesn't exist in db
- **ExpenseNotFoundException**: throw this exception when user enters an invalid product id which doesn't exist in db

---

Robust exception handling is crucial in any application to ensure smooth and predictable behavior when unexpected scenarios occur. In the Finance Management System, custom exceptions are created to handle specific application-level errors such as invalid user or expense IDs. These exceptions make the code more readable and maintainable by clearly defining the types of issues that can arise during execution.

Task 5.1 – UserNotFoundException

The UserNotFoundException class is a custom exception defined in the exception package. It is thrown when a user attempts to access or operate with a user ID that does not exist in the database. This exception provides a clear and descriptive way to indicate invalid user operations and can be used in services or repository methods to flag the issue.

```
package exception;

public class UserNotFoundException extends Exception {
    public UserNotFoundException(String message) {
        super(message);
    }
}
```

Example Usage:

```
if (!userExists(userId)) {
    throw new UserNotFoundException("User ID " + userId + " not found in the database.");
}
```

Task 5.2 – ExpenseNotFoundException

Similarly, ExpenseNotFoundException is another custom exception created in the exception package. It is thrown when a user provides an expense ID (or product ID) that cannot be found in the database records. This helps to handle expense-related errors gracefully and inform the user of invalid input with meaningful messages.

```
package exception;

public class ExpenseNotFoundException extends Exception {
    public ExpenseNotFoundException(String message) {
        super(message);
    }
}
```

Example Usage:

```
if (!expenseExists(expenseId)) {
    throw new ExpenseNotFoundException("Expense ID " + expenseId + " not found in the database.");
}
```

Conclusion :

The implementation of custom exceptions such as UserNotFoundException and ExpenseNotFoundException enhances the reliability and clarity of the Finance Management System. By explicitly handling specific error scenarios, the system becomes more user-friendly and maintainable. These custom exceptions not only improve code readability but also ensure that meaningful feedback is provided to the user when invalid operations are performed. Proper exception handling plays a critical role in building robust applications, and this task effectively integrates it into the system's workflow.
