Title:

Store Front CLI Application

Presenter's Name:

Salida Maharjan

Date:

August 3, 2025

Introduction to Store Front Application

- The StoreFront application is a command-line Java program that simulates a simple shopping experience. Users can:
- View available products.
- Add or remove products from a shopping cart.
- See the total price of their cart.
- Complete or cancel purchases.
- The application loads inventory from a JSON file.
- It runs an AdminService in a separate thread, allowing real-time inventory updates via local network commands.
- The application also includes a suite of JUnit tests to verify core functionalities of the ShoppingCart.

Goals and Design Decisions

- Create a simulated storefront with an interactive user experience.
- Implement core object-oriented principles such as inheritance and encapsulation.
- Allow dynamic inventory handling and runtime updates.
- Provide reliable unit tests to ensure code quality.
- Use of SalableProduct, the superclass for Weapon, Armor, and Health, allowing for polymorphism.
- JSON based inventory loading to simulate real world data loading from files.
- Introduce multithreading to handle administrative functions concurrently.
- Create detailed unit tests to verify correctness and robustness.

Challenges Encountered During Development

- Using Eclipse IDE was quite challenging
- The instruction for the application development was not clear, however with the help of instructor it made it clearer.
- Reading a malformed or missing JSON file initially caused application crashes; added a fallback to default products.
- Running AdminService in a separate thread was complex and required managing shared data safely.
- Console input required strict input validation to avoid crashes from invalid entries.
- Hardcoded product names led to redundant logic and difficulty scaling for new products

Pending Bugs or Issues Remaining

- The app can throw InputMismatchException if the user enters text where numbers are expected.
- Product addition/removal uses hardcoded names rather than dynamic lists from inventory.
- Need to refine the application.
- Proper front end design for better UI and UX.
- Need the use of database for dynamic data.

Demo

Five Things Learned That Can Be Reused in Future Projects

- Unit Testing with JUnit: Writing comprehensive tests for classes like ShoppingCart ensures reliable code and fewer bugs.
- File I/O & JSON Parsing: Working with Jackson's ObjectMapper to read/write JSON is practical for real-world applications.
- OOP Best Practices: Encapsulating product types and using polymorphism made the design flexible and easy to extend.
- Multithreading Basics: Spawning and managing a background service thread gave experience in concurrency and synchronization.
- User Interaction Design: Designing a logical and intuitive CLI interface sharpened skills in usability and flow control.

Thank You