Mid-Term Assessment: Café POS & Delivery Project (Weeks 2–6)

Purpose

The Mid-Term (Week 7) is your first major milestone in the Café POS & Delivery Project. This assessment evaluates the progress you have made from **Week 2 through Week 6**, where you incrementally built up the foundations of the system using object-oriented design principles and classic design patterns.

By the mid-term, you are expected to have a system that is cohesive, extensible, and testable. The goal is not only that the program runs, but that it reflects thoughtful design, clean structure, and evidence of weekly progress.

In particular, you must show integration of:

- **Week 2:** Money & Orders (core ordering system)
- **Week 3:** Payment Strategies (cash, card, wallet)
- **Week 4:** Observer Notifications (kitchen, delivery, customer)
- **Week 5:** Decorator + Factory for flexible product creation
- **Week 6:** Refactored Pricing (discount and tax policies, receipt printer, orchestrator)

Assessment Checklist (What Must Exist by Midterm)

1. **Core Order System (Week 2)**
   You must implement orders with line items, subtotal, tax, and total calculations using the Money type. This proves that your system can handle basic POS math.

2. **Payment Strategies (Week 3)**
   Your project must include cash, card, and wallet strategies. This proves that your design supports polymorphism and extensibility in handling payments.

3. **Observer Pattern (Week 4)**
   Kitchen, Delivery, and Customer observers must be notified at the right time. This proves that your system can broadcast events without hard-coding dependencies.

4. **Decorator & Factory (Week 5)**
   You must demonstrate recipe-driven product creation, such as ESP+SHOT+OAT or LAT+L. Add-ons must work through decorators, and clients should only need to call the factory. This proves you can combine patterns to achieve the Open/Closed Principle.

5. **Refactored Pricing (Week 6)**
   Pricing must be handled through discount policies, tax policies, and a receipt

printer, orchestrated by a checkout service. This proves that you can refactor God classes into clean, testable services while keeping outputs unchanged.

6. **Tests**
   You must provide JUnit tests for every week's work. Characterization tests must prove that refactoring did not change outputs, and unit tests must prove that new policies behave as expected.

Deliverables

– Code in your Git repository with meaningful weekly commits (not one bulk dump).
– A passing JUnit test suite covering all required features.
– A working demo class Week6Demo showing (the details will available in Week 6):

  - Creating decorated products via the factory
  - Paying with one of the payment strategies
  - Notifying observers
  - Applying discount and tax policies
  - Printing a correct receipt

Rubric (30 Marks)

Category	**Marks**	**What is expected**
Correctness of Features	10	All features from Week 2–6 implemented & working together (Money, Order, Payments, Observer, Factory, Decorators, Pricing Service).
Code Quality & Refactoring	6	Smells removed, SOLID applied, classes/interfaces well-defined.
Testing & Reliability	6	Comprehensive JUnit tests (unit + integration + characterization/equivalence). All pass.
Integration & Demo	5	Working Week6Demo proving full flow, with correct outputs.
Commit History & Effort Evidence	3	Meaningful commits per week, not one bulk dump; clear evidence of incremental work.

Note: The detailed rubric for assessing the project will be discussed 1 week prior to the submission deadline.

Final Note

By the mid-term you will have implemented the foundations of a professional Café POS system. You will have practiced clean design, refactoring, and the use of multiple patterns (Strategy, Observer, Decorator, Factory). You will also have proven that your design can evolve without breaking behavior through testing. This milestone demonstrates your ability to deliver a working, extensible software system incrementally and thoughtfully.