Institute of Information Technology
University of Dhaka
Bachelor of Science in Software Engineering (BSSE)
2nd Year 4th Semester Final Examination, 2025
CSE 2215L : Design Patterns Lab
Marks: 30 # Duration: 1.5 hours

You are enhancing a Quiz System. The current system supports a few quizzes where each quiz contains multiple questions: MCQ, True/False, or Fill-in-the-blank. The GUI is fully functional and does not need any modification. Your task is to refactor the existing business logic (attached code in Google Classroom) to implement design patterns, improve maintainability, and make the system easily extensible.

Problems in the Current Code

1. **Centralized State Management**
   - Right now, the quiz state (current quiz, current question, score) is scattered in the *Main.java* class.
   - This makes the code hard to maintain and test.
   - We need a single place to manage the exam state consistently across the system.
2. **Question Creation**
   - Questions are currently created with direct new calls and type strings like "MCQ", "TF", "FILL".
   - If a new question type is added, you must modify multiple places in the code.
   - This violates the Open/Closed Principle.
3. **Scoring Logic**
   - The scoring logic is hardcoded with if-else or switch statements in the UI layer.
   - Adding a new scoring method (e.g., "Bonus Scoring") means modifying the main logic again.
4. **Score Updates**
   - Right now, the score label is updated directly in the UI logic.
   - This creates tight coupling between the business logic and the GUI.
   - The system needs a way for the score display to update automatically whenever the score changes, without direct dependencies.

Refactoring Tasks

Your job is to solve the above issues by refactoring with appropriate design patterns.

- Introduce a single centralized manager to handle the quiz state and score.
- Create a flexible way of constructing different question types without if-else checks.
- Move scoring logic into its own interchangeable components so that new scoring methods can be added without touching the main code.
- Ensure that the score display updates automatically when the score changes, without direct calls from the business logic.

Expected Outcomes After Refactoring

- New quizzes and question types can be added by creating new classes — no changes to existing code.
- Scoring logic is decoupled from the UI and can be switched dynamically.
- Score updates happen automatically whenever a question is answered, without UI needing to call business logic directly.
- Question creation is dynamic, avoiding if-else or type checks.
- All state management is centralized in a single manager.

Marking Rubric

- Identifying appropriate patterns (analysis & justification): **6 marks**
- Refactoring centralized state management: **6 marks**
- Refactoring question creation: **6 marks**
- Refactoring scoring logic: **6 marks**
- Decoupling UI from score updates: **6 marks**