# Project 04

## Project Idea Using OOP Concept

태아카

# Exam System with Dynamic Difficulty & Performance Analytics

**An adaptive, feedback-oriented exam platform for self-assessment**

## Project Overview

- **Purpose:** Create an offline exam system that dynamically adjusts question difficulty and provides performance insights

- **Key Features:**
  - **Dynamic Question Difficulty**: Adapts question levels based on user performance
  - **Analytics Dashboard**: Tracks Strengths, weakness, and trends over time

- **Technologies:**
  - C++ for core programming
  - Local Database for storing questions, answers, and results

## System Features

- **Core Features**
  - (1) Dynamic Question Difficulty:
    - Automatically adjust question difficulty based on the student’s accuracy and speed
    - Provides a personalized experience that helps students build condidence
  - (2) Analytics Dashboard:
    - Tracks test scores, time spent per question difficulty trends
    - Highlights strengths and areas for improvement
  - (3) Question Types:
    - Support **TRUE/FALSE** and **Multiple Choice** questions for self-assessment

## Functional flow

- **System Workflow**
  - (1) Exam Creation (Admin):
    - Admin adds questions to a pool with difficulty levels
  - (2) Exam Taking (Student):
    - Students answer qusestions that vary in difficulty
  - (3) Dynamic Adjustment:
    - Difficylty Engine recalibrates based on student performance
  - (4) Perfomance Analysis:
    - Dashboard provides a performance summary at the end

- **Visual:** Flowchart showing steps from exam creation to final analytics

# Class structure Overview

- **Class structure & OOP Concepts**
  - ***Question*** Base Class with *MultipleChoice* and *TrueFalse* subclasses for flexibility
  - ***DynamicDifficultyEngine***: Adjusts question difficulty level
  - ***PerformanceTracker***: Logs accuracy, time, and trends
  - ***Exam* Class**: Core controller that coordinates question delivery and analytics
  - ***User* Classes**:
    - *Admin*: Creates and manages exams.
    - *Student*: Takes exams and reviews performance.

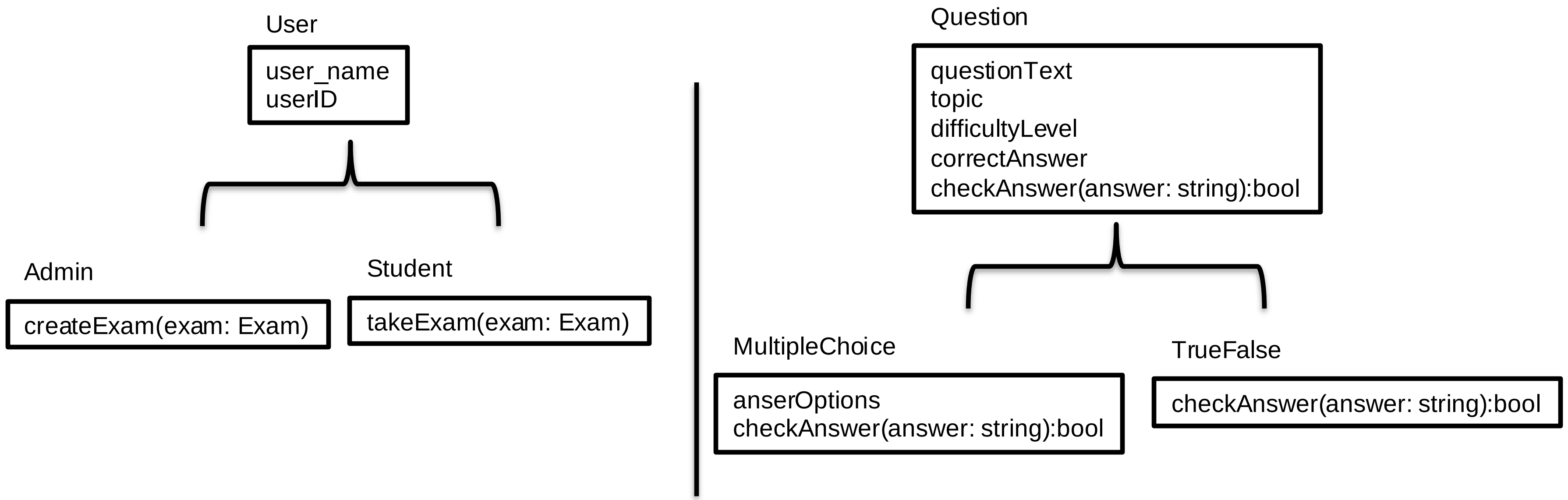
- **Visual:** Class Diagram (showing relationship between *Quesiton*, *Exam*, *PerformanceTracker*, and *DynamicDifficultyEngine*)

# How OOP Principle Enhance the System

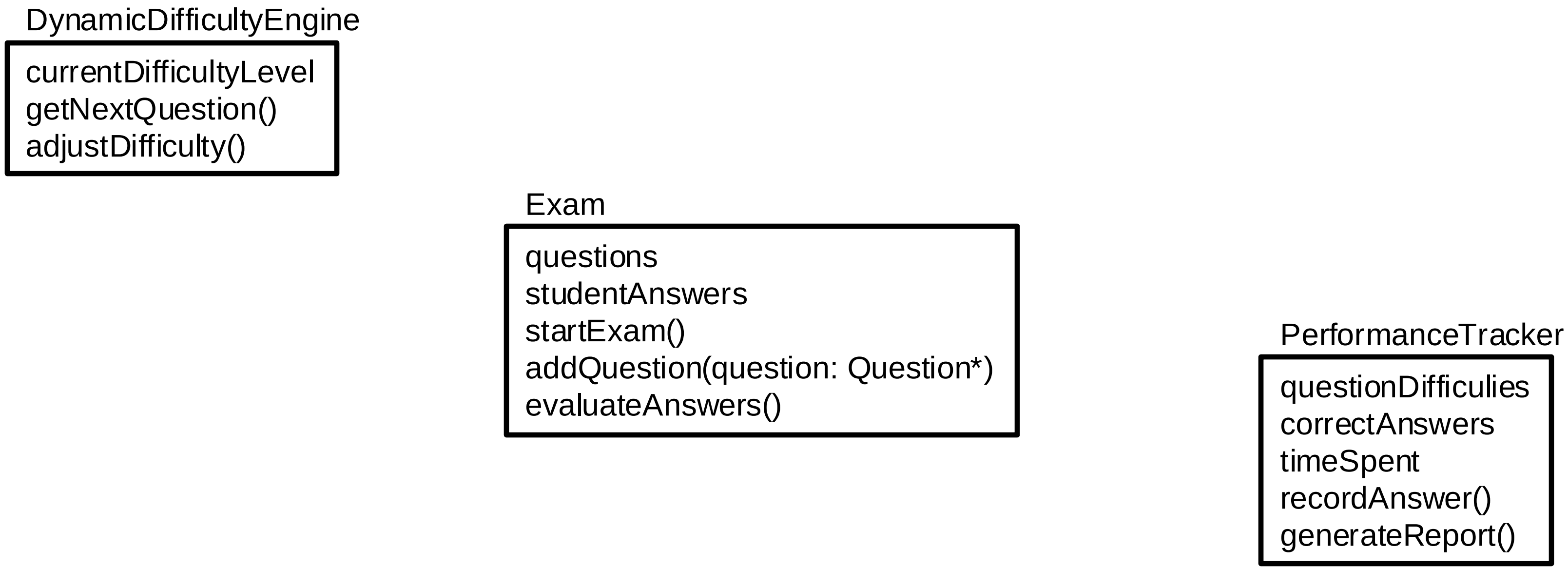
- **Inheritance:**
  - *Question* subclasses (*MultipleChoice* and *TrueFalse*) allow flexible question handling
- **Polymorphism:**
  - Different question types are handled uniformly, thanks to polymorphic *checkAnswer()* methods
- **Encapsulation:**
  - Self-contained components like *DynamicDifficultyEngine* and *PerformanceTracker* simplify maintenance
- **Composition:**
  - *Exam* uses *PerformanceTracker* and *DynamicDifficultyEngine* to manage question adaptation and results

# Basic & Derived Classes

## Exam System



## Core Classes



## Key Benefits

- **For Students:**
  - Adaptive testing builds confidence and helps with targeted learning
  - The dashboard provides a clear view of strengths and weakness

- **For Developers:**
  - Modular design simplifies extension (e.g. adding new question types)
  - Real-world OOP principles applied to build a structured system

## Thank You!

**HTET ARKAR (hak3601@cau.ac.kr)**