COMP3211 Software Engineering

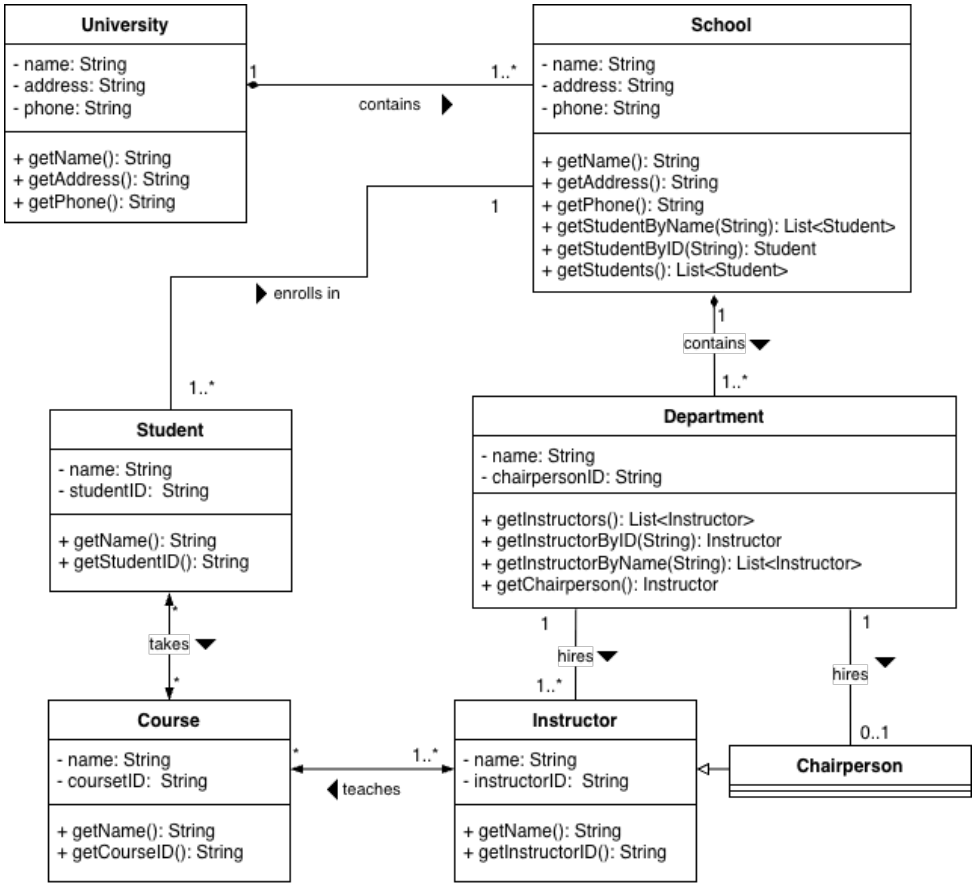
Assignment 2

Author
Wang Yuqi
[redacted]

Lecturer
Prof. Max Yu PEI

BsC in Computer Science
Academic year: 2025/2026 Sem 1

Q1: Class Diagram (8 marks)



Design Rationale:

Relationships	Type	Multiplicity	Evidence
University vs. School	Strong Aggregation	1, 1..*	*University comprises more than one school*
School vs. Department	Strong Aggregation	1, 1..*	*Each department ... is affiliated with a school*
Department vs. Instructor	Association	1, 1..*	*instructor join and leave departments (**not aggregation**), each department has at least one instructor. Multiple contracts ... strictly forbidden*
Department vs. Chairperson	Association	0...1,1	chairperson if exists (hence 0...1)
Chairperson vs. Instructor	Generalization	-	chairperson ... is an instructor
Course vs. Instructor	Association	*, 1..*	course taught jointly by several instructors
Student vs. Course	Association	*, *	Certain course are not aken by any students
School vs Student	Association	1, 1..*	Student join leave schools (**not aggregation**)

Q2: Architectual Design (10 marks)

1. **User Interface**:
   - (e) database browser
   - (h) browser UI
   - (i) mobile UI
2. **User Communications / Authentication and Authorization**:
   - (a) mobile device management
   - (g) forms management
3. **Information Retrieval and Modification**
   - (b) database search
   - (f) database query management
4. **Transaction Management Database**
   - (d) equipment database
   - (c) buildings database
   - (j) vehicle database

Q3: Software Testing

Let's denote:

1. `if(x <= 0 || y <= 0 || z <= 0)` as condition **`C1`**
2. `if(x + y <= z)` as condition **`C2`**
3. `else if (x + z <= y)` as condition **`C3`**
4. `else if (y + z <= x)` as condition **`C4`**
5. `else` as condition **`C5`**

x	y	z	ExpectedResult	BranchesCovered
−1	0	−5	`false`	`C1==true`
1	1	2	`false`	`C2==true`
2	8	3	`false`	`C3==true`
6	3	2	`false`	`C4==true`
5	5	5	`true`	`C5==true`

Note: when we say `Ci==true` we automatically assume `C1==false`, `C2==false`, ..., `Ci-1==false`.

Q4: Software Maintenance (8 marks)

Q4.1: Centralized VC systems have a single master project repository. Each dev has a private workspace containing only components checked out from the project repoistory. In distributed VC, each dev's private workspace is a full clone of the project repositor (entire project history + working directory), meaning multiple repositories exist simutaneously.

Q4.2:

1. In VCS, later versions of a code is derived from earlier versions via *codelines*. The system labels versions of components and baselines so particular system release can be traced back and reconstructed loselessly.
2. The system records history: what chagned, when, by whom, in the form of a sequence of deltas and commits. This allows for debugging, contribution analysis, blames, etc.

Q5: Software Evolution (8 marks)

If I were to add an "auto" mode to the jungle featuring a random computer agent, a wide range of changes needs to be made. Firslty, I need to refactor the turn-taking logic so the `Game` class no longer asks the TUI directly for a move. Rather, each `Player` now expose a method like `getNextMove(Board board)`; for normal players ,htis delegates to a `HumanMoveProvider`, but when in "auto" mode, this switches to an internal strategy handled by `RandomMoverProvider`. From then on, `getNextMOve` always delegates to the random provider and never returns to human provider. This makes it structually impossible to exit auto mode.

To implement computer behavior, I'd extend existing move-validation logic. Introduce a `MoveGenerator` service that can enumerate all legal moves for a given player based on current `Board` and `Piece` information, compile them into a list of moves, then using a PSRNG to randomly choose one.

The main game loop in `Game` becomeds simpler. Now, it always called `currentPlayer.getNextMove(board)`, applies retunred move to game engine, check game-over conditon, invariant to human/auto mode.