Lab 2:

Part A:

Noun Analysis:

Places**:** University

Systems: Inspera, JU-App, Pingpong, Ladok, Kronox, studentwebben

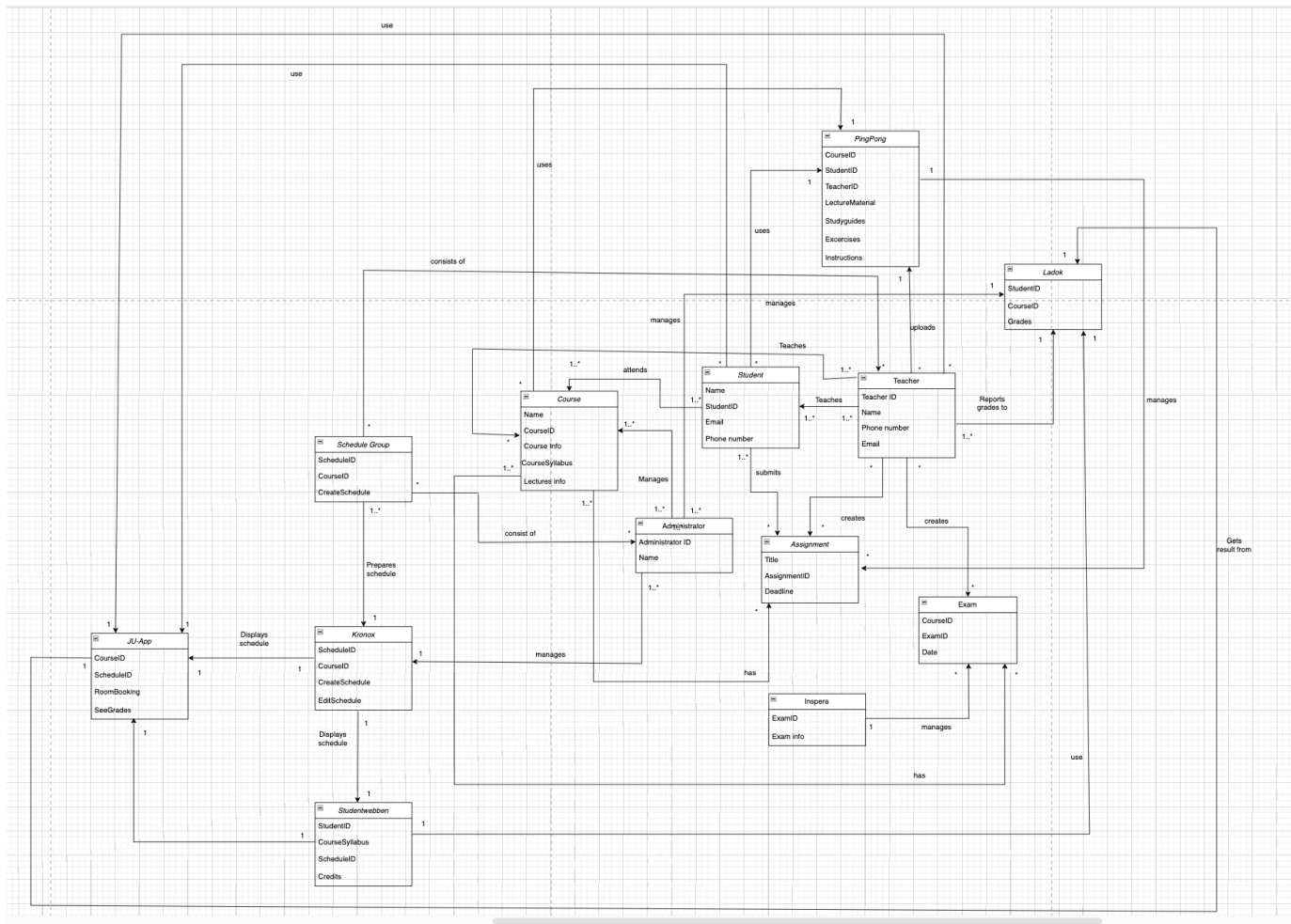Users: Student, Teacher, Administrator, Schedule group

Material (Attributes): Exercises, instructions, schedule, study guides, lecture material, course syllabus, info about grades and credits,

More attributes: see diagram

Results/tests: Grades, credits, assignments and exams

Core (most essential part): Course

Explanation: Starting with the "Course" makes total sense. It's like the backbone of the system. Courses define what's being taught, who's teaching, and how students engage. Everything else, like materials, assignments, exams, and even schedules, revolves around courses. It's where the magic of teaching and learning begins, making it the natural starting point for designing the system.

Part B:

KingStrategy
- + KingStrategy(chess: Chess, piece: Piece)
- + getValidMoveToPositionsIgnoringCheck(): List<Position>
- + getletter(): char

RockStrategy
- + RockStrategy(chess: Chess, piece: Piece)
- + getValidMoveToPositionsIgnoringCheck(): List<Position>
- + getletter(): char

QueenStrategy
- + QueenStrategy(chess: Chess, piece: Piece)
- + getValidMoveToPositionsIgnoringCheck(): List<Position>
- + getletter(): char

KnightStrategy
- + KnightStrategy(chess: Chess, piece: Piece)
- + getValidMoveToPositionsIgnoringCheck(): List<Position>
- + getletter(): char

PawnStrategy
- + PawnStrategy(chess: Chess, piece: Piece)
- + isPawnOnStartPosition(): boolean
- + getValidMoveToPositionsIgnoringCheck(): List<Position>
- + shouldTurnIntoOtherType(): boolean
- + getletter(): char

BishopStrategy
- + BishopStrategy(chess: Chess, piece: Piece)
- + getValidMoveToPositionsIgnoringCheck(): List<Position>
- + getletter(): char

PieceStrategy
- - chess: Chess
- - piece: Piece
- + PieceStrategy(chess: Chess, piece: Piece)
- + getValidMoveToPositionsIgnoringCheck(): List<Position>
- + getLetter(): char
- + shouldTurnIntoOtherType(): boolean

Extends (KingStrategy, RockStrategy, QueenStrategy, KnightStrategy, PawnStrategy, BishopStrategy → PieceStrategy)

Position
- - x: int
- - y: int
- + Position(x: int, y: int)
- + getX(): int
- + getY(): int
- + equals(object: Object): boolean

Piece
- - chess: Chess
- - team: Team
- - xPosition: int
- - yPosition: int
- - pieceStrategy: PieceStrategy
- + Piece(chess: Chess, team: Team, xPosition: int, yPosition: int, type: PieceType)
- + turnInto(type: PieceType): void
- + hasPosition(x: int, y: int): boolean
- + getValidMoveToPositions(): List<Position>
- + shouldTurnIntoOtherType(): boolean
- + getLetter(): char

Chess
- - board: Board
- - nextTeamToMove: Team
- - pieces: ArrayList<Piece>
- + Chess()
- + getNextTeamToMove(): Team
- + movePieceTo(piece: Piece, moveToPosition: Position): boolean
- + changeTurn
- + getPieceInSquare(x: int, y: int): Piece
- + isSquareFree(x: int, y: int): boolean

<<Enumeration>> Team
- White
- Black

Move
- - dx: int
- - dy: int
- + Move(dx: int, dy: int)
- + getDx(): int
- + getDy(): int
- + getDestination(movingFrom: Position): Position

<<Enumeration>> PieceType
- Rock
- Knight
- Bishop
- King
- Queen

Board
- - xSize: int
- - ySize: int
- + Board(xSize: int, ySize: int)
- + getXSize(): int
- + getYSize(): int
- + containsSquare(x: int, y: int): boolean

Main
- + main(args: String[]): void

ChessView
- - chess: Chess
- - selectedPiece: Piece
- + ChessView(chess: Chess)
- + paint(graphics: Graphics)
- + mousePressed(event: MouseEvent)

Use relationships: PieceStrategy → Chess; Piece → Position; Piece → Team; Piece → Move; Piece → PieceType; Chess → Team; Chess → Board; Main → Chess; Main → ChessView; ChessView → Move