1, Domain description (Part A)

1 Domain description (part A)

Our domain is teaching at University. Our focus lies on students and teachers, and on the different tools and systems they use to support and organize their interaction. It follows a detailed description of this domain (inspired by the real-world, but simplified and cut off for the purpose of this lab).

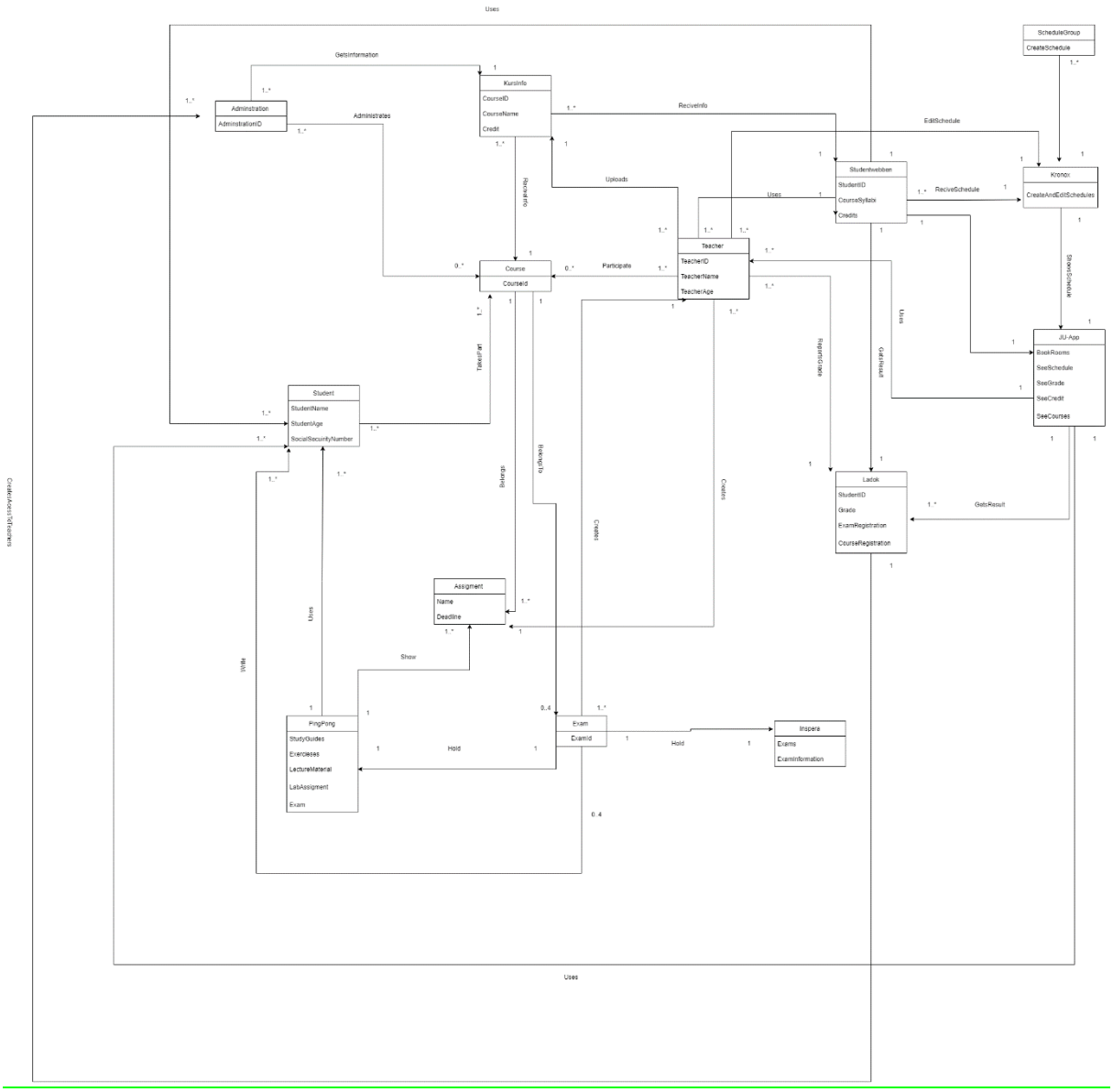
On the one hand we have users such as students, teachers, administrators, and on the other hand we have a number of systems that interact with one another and that the users interact with. PingPong is a system supporting the processes of teaching and learning, and communication between students and teachers. Teachers upload on PingPong material such as study guides, exercises, lab assignment instructions, lecture material. Students retrieve this material from PingPong. Teachers create on PingPong assignments for the students. Students submit their solutions for assignments on PingPong, where teachers retrieve submissions and give feedback via PingPong. There are two exam systems for digital exams: either PingPong as well, or Inspera. Teachers create digital exams, students can sit digital exams and submit their solutions via the exam system. Teachers can retrieve students' solutions for exams and give feedback via the exam system. Grades for exams and other examination elements are entered by the teachers into the system Ladok. This is only possible, however, if administrators have made a course and its examination

elements available for the teachers in the Ladok system. The administrators take the information about a course's examination elements form the system Kursinfo. It is the teachers who specify courses and their examination elements by creating course syllabi (kurs plan) and uploading this information to Kursinfo. A schedule group prepares the schedules for courses in the system Kronox. Teachers can also edit the schedule in Kronox. A mobile application JU-App is used by students and teachers. JU-App extracts schedule information from Kronox and information about grades and credits from Ladok in order to present it to the user in a readable manner. Students use the system Studentwebben (Current student) in order to view the schedule (extracted from Kronox), information about grades and credits (extracted from Ladok) and to get the course syllabi (extracted from Kursinfo).

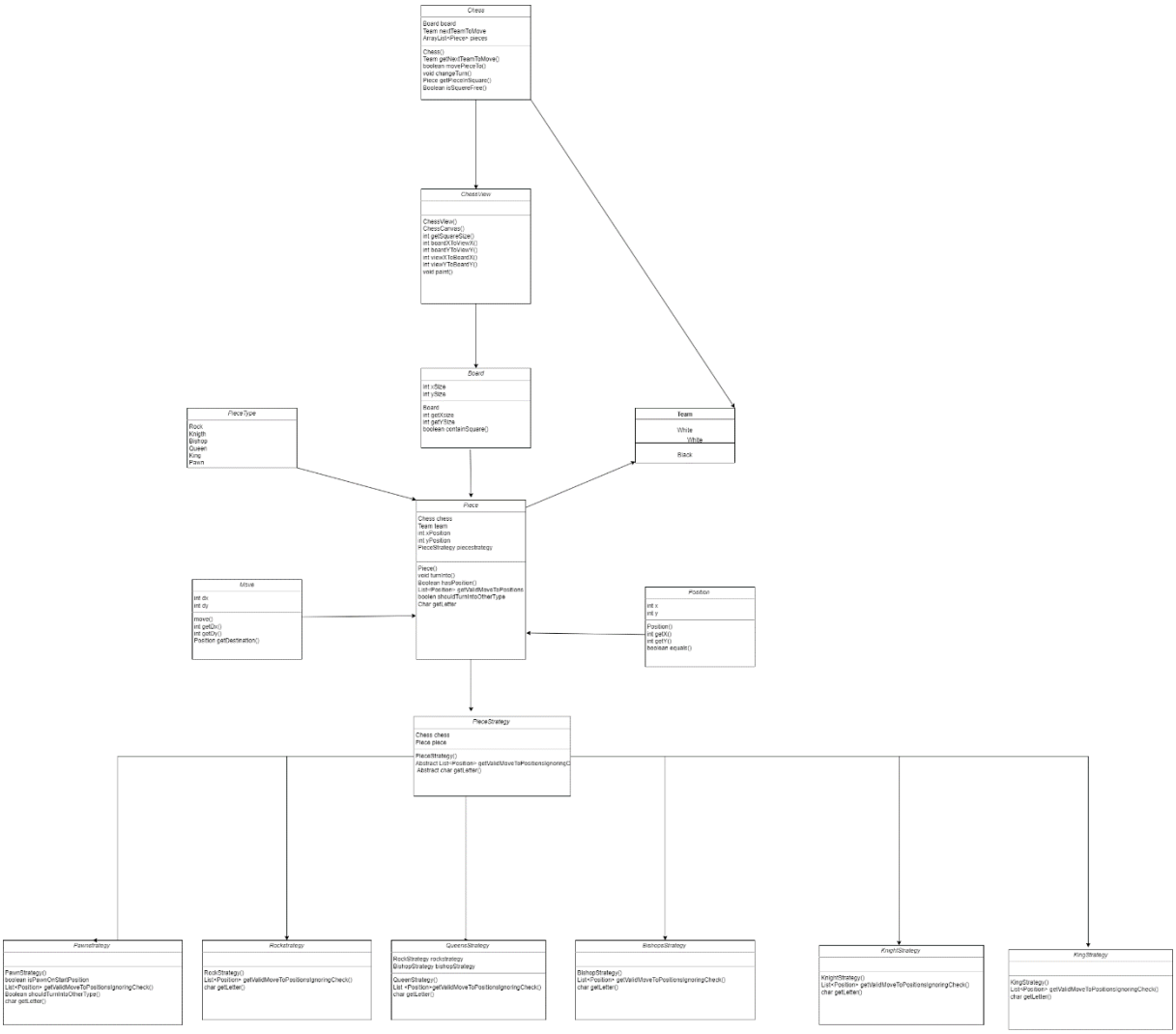
2, Your task (part A)

Do a noun analysis: identify potential conceptual classes. Decide which concepts should be present in your model.

Coneptual Class Category	Example
Systems used	Pingpong, Inspera, Ladok, Kronox, JU-App, studentwebben
Material	Study guides, exercises, lab assignment instructions, lecture material, schedule, course syllabi, information about credits and grades
Results	Grades, credits, exams, assignments
Role of people or organistation related to a the	Students, teachers and administration
Places	University


Uses
GetsInformation
Administration
AdministrationID
Administrates
KursInfo
CourseID
CourseName
Credit
ReciveInfo
EditSchedule
ScheduleGroup
CreateSchedule
Uploads
Recivelnfo
Studentwebben
StudentID
CourseSyllabx
Credits
ReciveSchedule
Kronox
CreateAndEditSchedules
Uses
Teacher
TeacherID
TeacherName
TeacherAge
Course
CourseId
Participate
ShowSchedule
JU-App
BookRooms
SeeSchedule
SeeGrade
SeeCredit
SeeCourses
TakesPart
RegistrGrade
GetsResult
Student
StudentName
StudentAge
SocialSecurityNumber
Belongs
BelongsTo
Creates
Ladok
StudentID
Grade
ExamRegistration
CourseRegistration
GetsResult
Assigment
Name
Deadline
Show
Write
PingPong
StudyGuides
Exercieses
LectureMaterial
LabAssigment
Exam
Hold
Exam
ExamId
Inspera
Exams
ExamInformation
CreatesAccessToTeachers
Uses

3,


Chess
Board board
Team nextTeamToMove
ArrayList<Piece> pieces
Chess()
Team getNextTeamToMove()
boolean movePiece()
void changeTurn()
Piece getPieceInSquare()
Boolean isSquareFree()
ChessView
ChessView()
ChessCanvas()
int getSquareSize()
int boardXToViewX()
int boardYToViewY()
int viewXToBoardX()
int viewYToBoardY()
void paint()
Board
int xSize
int ySize
Board
int getXSize
int getYSize
boolean containSquare()
PieceType
Rook
Knight
Bishop
Queen
King
Pawn
Team
White
Black
Piece
Chess chess
Team team
int xPosition
int yPosition
PieceStrategy pieceStrategy
Piece()
void turnInto()
Boolean hasPosition()
List<Position> getValidMoveToPositions
boolean shouldTurnIntoOtherType
Char getLetter
Move
int dx
int dy
move()
int getDx()
int getDy()
Position getDestination()
Position
int x
int y
Position()
int getX()
int getY()
boolean equals()
PieceStrategy
Chess chess
Piece piece
PieceStrategy()
Abstract List<Position> getValidMoveToPositionsIgnoring
Abstract char getLetter()
PawnsStrategy
PawnStrategy()
boolean isPawnOnStartPosition
List<Position> getValidMoveToPositionsIgnoringCheck()
Boolean shouldTurnIntoOtherType()
char getLetter()
RookStrategy
RookStrategy()
List<Position> getValidMoveToPositionsIgnoringCheck()
char getLetter()
QueenStrategy
RookStrategy rookstrategy
BishopStrategy bishopStrategy
QueenStrategy()
List <Position> getValidMoveToPositionsIgnoringCheck()
char getLetter()
BishopsStrategy
BishopStrategy()
List<Position> getValidMoveToPositionsIgnoringCheck()
char getLetter()
KnightStrategy
KnightStrategy()
List<Position> getValidMoveToPositionsIgnoringCheck()
char getLetter()
KingStrategy
KingStrategy()
List<Position> getValidMoveToPositionsIgnoringCheck()
char getLetter()