CHALMERS
EXAMINATION / TENTAMEN

Course code/kurskod	Course name/kursnamn			
DIT 185	Software Analysis and Design			
Anonymous code Anonym kod		Examination date Tentamensdatum	Number of pages Antal blad	Grade Betyg
DIT185-0036-REY		31/5 - 23	11	5

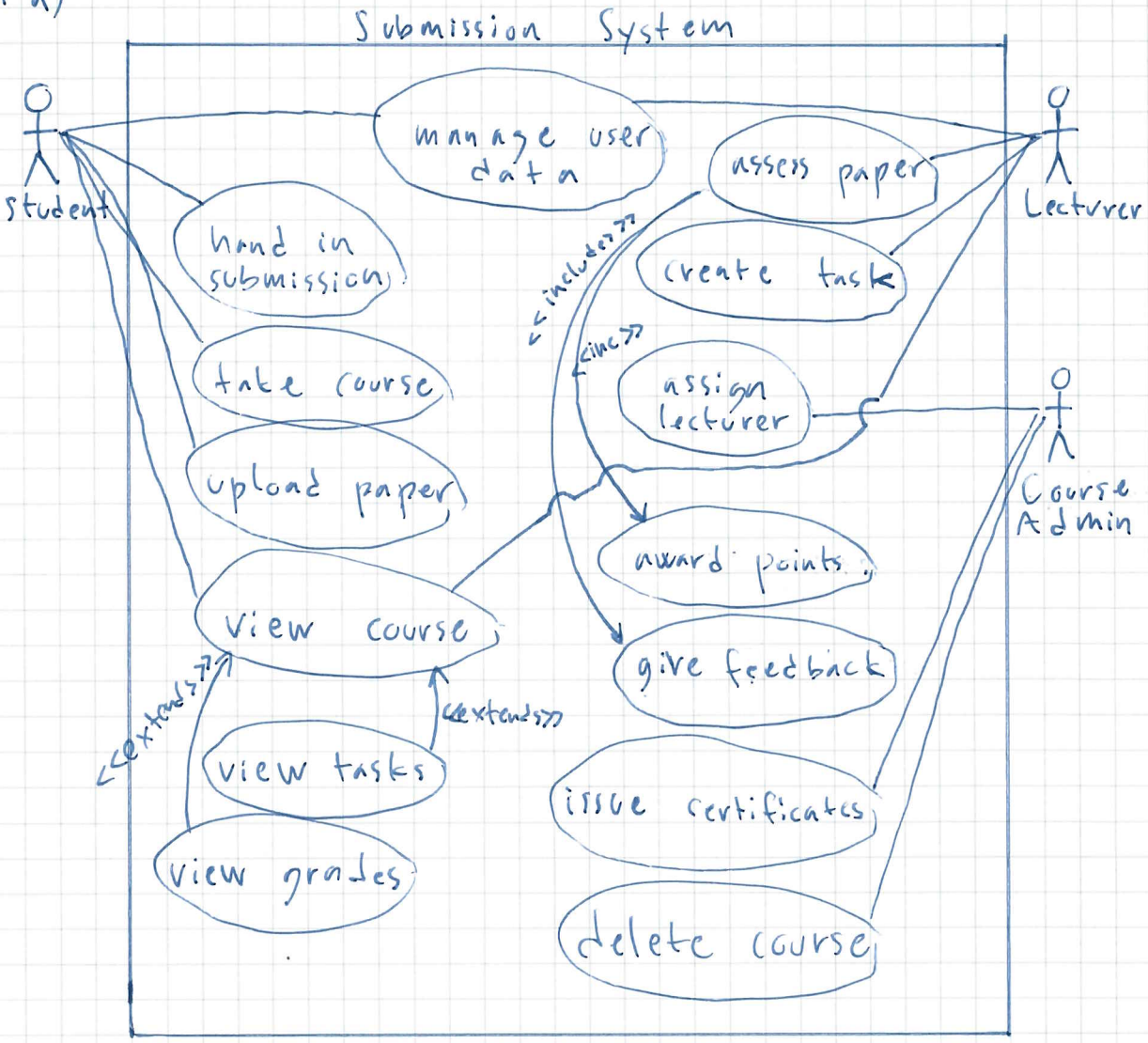
* I confirm that I've no mobile or other similar electronic equipment available during the examination.
Jag intygar att jag inte har mobiltelefon eller annan liknande elektronisk utrustning tillgänglig under eximinationen.

Solved task Behandlade uppgifter No/nr		Points per task Poäng på uppgiften	Observe: Areas with bold contour are to completed by the teacher. Anmärkning: Rutor inom bred kontur ifylles av lärare.
1	✓	12	
2	✓	12	
3	✓	9	
4	✓	10	
5	✓	10	
6	✓	10	
7	✓	15	
8	✓	13	
9			
10			
11			
12			
13			
14			
15			
16			
17			
Bonus poäng			
Total examination points Summa poäng		91	

[redacted]

CHALMERS	Anonymous code	Points for question (to be filled in by teacher)	Consecutive page no. Löpande sid nr 1
	Anonym kod DIT185-GG36-RFY	Poäng på uppgiften (ifylles av lärare)	Question no. Uppgift nr 1

1 a)



8

3

1b)

Use case name: Manage user data.

Goal in context: The user wants to change their current user data.

Primary actor: SS User.

Precondition: The SS User is registered as a student or lecturer.

Success end condition: The SS User updates their user data.

Failed end condition: The SS User fails to update their user data.

Trigger: The SS User has new user information.

Main success scenario:
1. The SS User navigates to the user data page.
2. The SS User goes to the settings section.
3. The SS User enters the data they want to change.
4. The system prompts the user to verify themselves.
5. The system informs the SS User that their data has been updated.

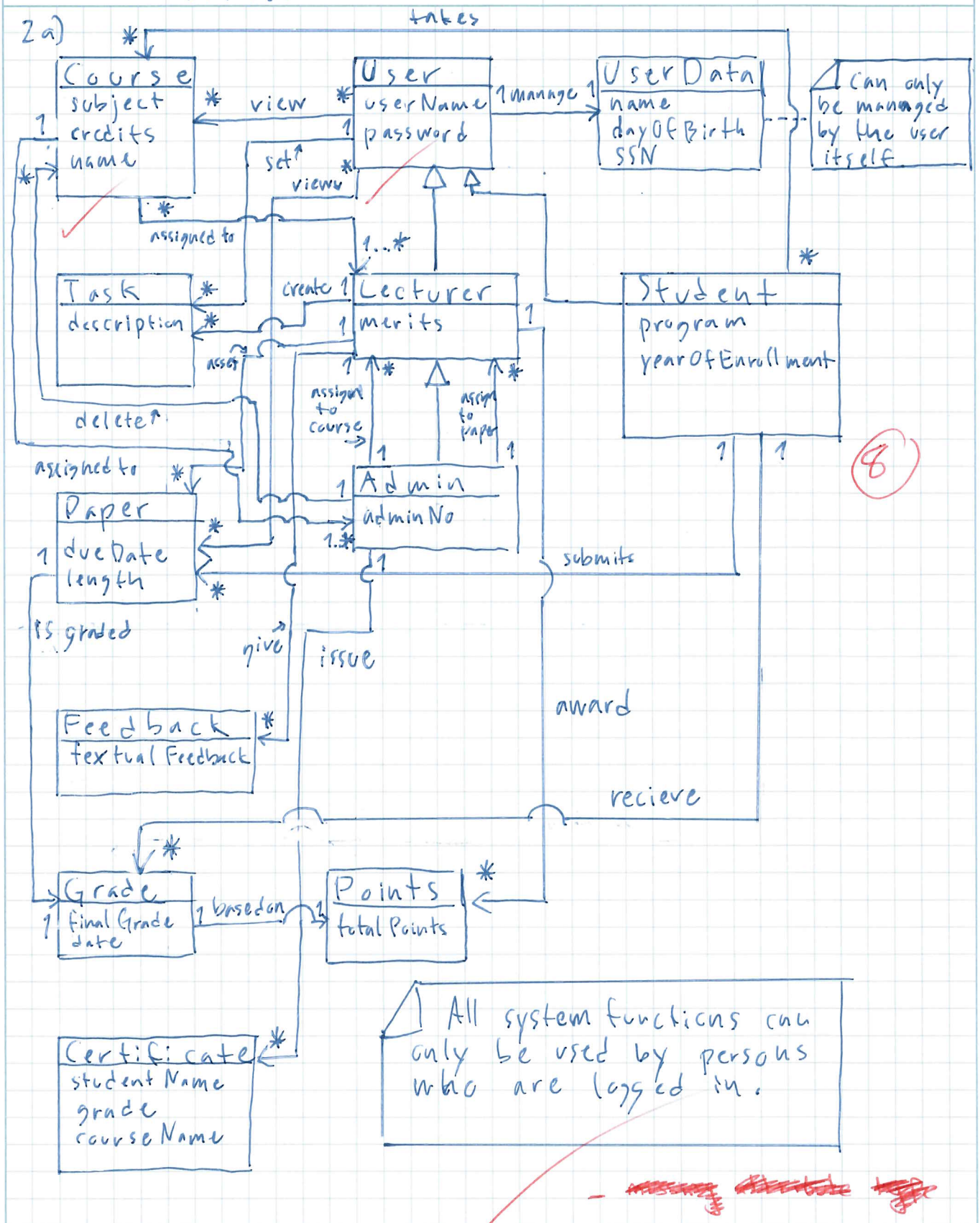
Extensions:
3a. The SS User fails to enter any information
3a1. The system prompts the SS User to enter valid information.

Sub-variation: 4. The SS User may identify themselves via:

Email
Password for user
Text message
Bank-ID

4

2a)


takes
Course
subject
credits
name
view
User
userName
password
manage
UserData
name
dayOfBirth
SSN
Can only be managed by the user itself
set
views
assigned to
Task
description
create
Lecturer
merits
Student
program
yearOfEnrollment
assign
assigned to course
assign to paper
delete
assigned to
Admin
adminNo
Paper
dueDate
length
submits
is graded
give
issue
award
Feedback
textual Feedback
recieve
Grade
final Grade
date
based on
Points
total Points
Certificate
student Name
grade
courseName
All system functions can only be used by persons who are logged in.

8

3

2b) 1. I started with looking at the noun phrases in the textual description to find candidates for classes.

2. I modelled the real world entities into classes (4)

3. I modelled the conceptual entities into classes.

4. I explored the generalizations between the classes to find out if there were any inheritance.

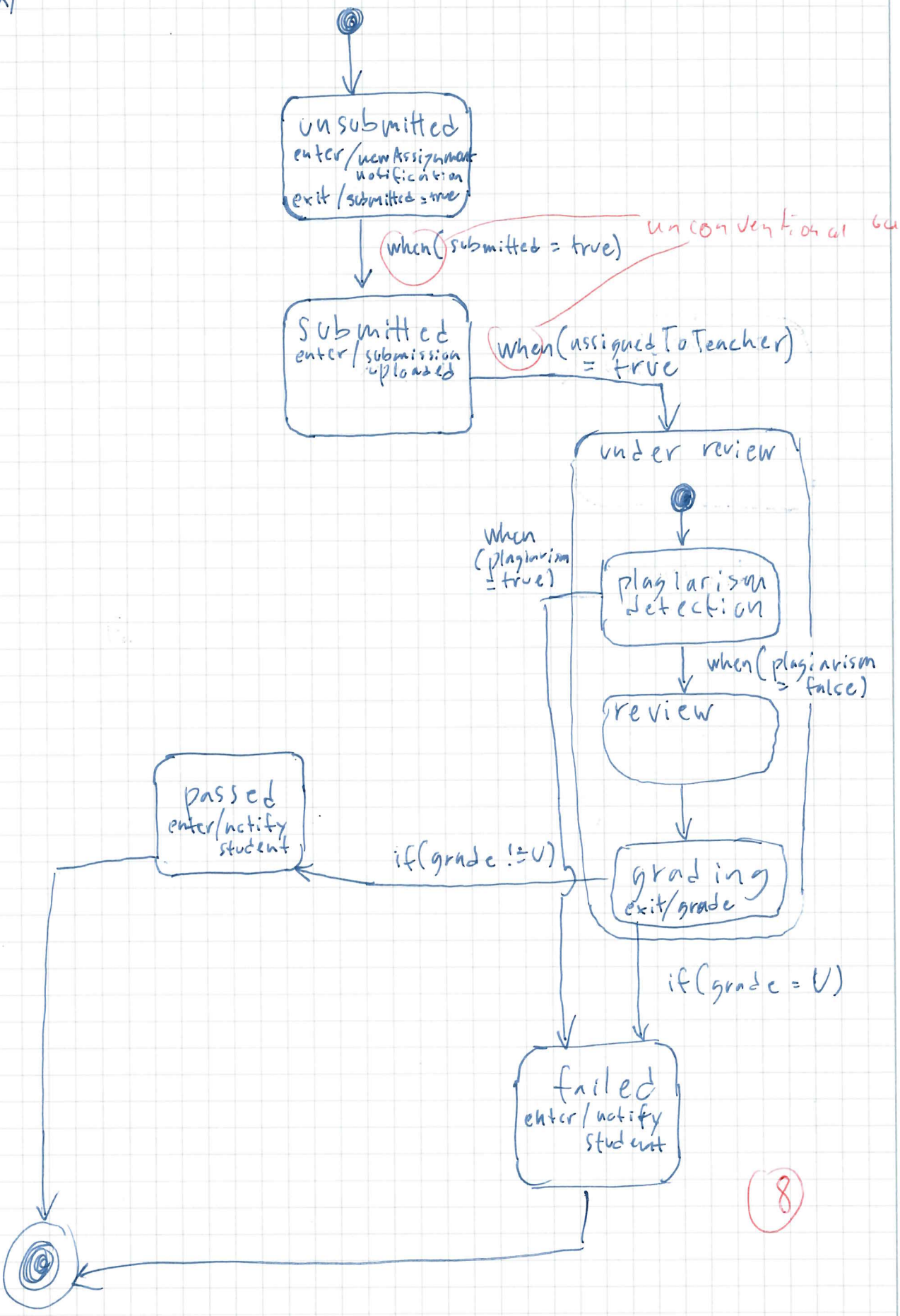
5. I explored the relations between the classes, gave them cardinalities and made sure that the relations were not classes of themselves.

Note: Since the domain model became very large, I put some small arrows for the name of the relationships to make clear to what relation they belong.

I am aware that this is not part of the syntax of the UML.

3

3 a)


unsubmitted
enter / newAssignment notification
exit / submitted = true
when(submitted = true)
unconventional but ok
submitted
enter / submission uploaded
when(assignedToTeacher) = true
under review
when (plagiarism = true)
plagiarism detection
when(plagiarism = false)
review
passed
enter/notify student
if(grade != U)
grading
exit/grade
if(grade = U)
failed
enter/notify student

8

3

3b)

Before:

SciencePaper : Paper
submittedBy = Joel submitted = false plagiarism = false assignedToTeacher = false grade = null.

Submission →

After:

Science Paper : Paper
submittedBy = Joel submitted = true plagiarism = false assignedToTeacher = false grade = null

should illustrate transitions, not states -3

Before:

MathPaper : Paper
SubmittedBy : Joel Submitted = true plagiarism = false assignedToTeacher = true grade = null

passed →

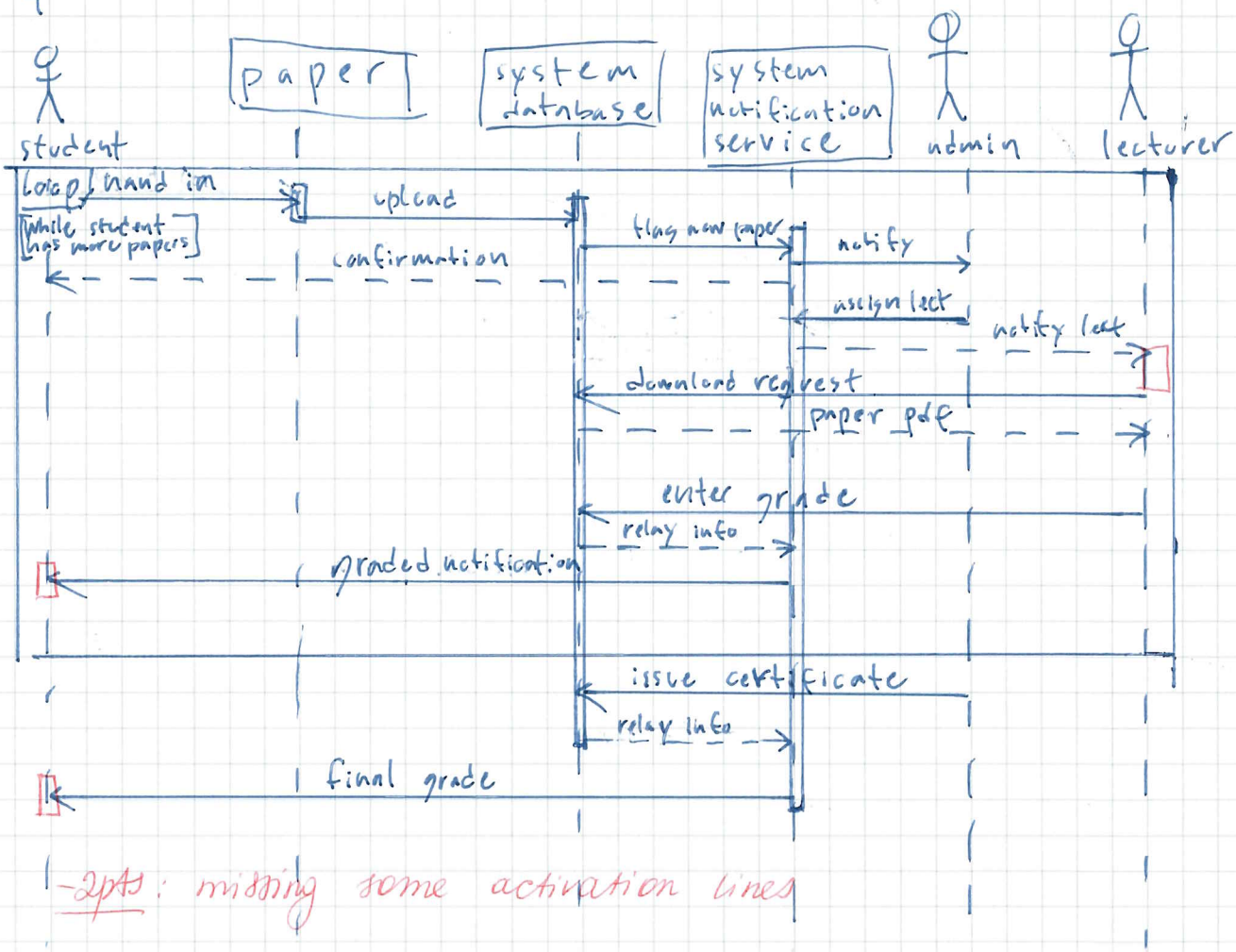
After:

MathPaper : Paper
SubmittedBy = Joel Submitted = true plagiarism = false assignedToTeacher = true grade = 5

1

3

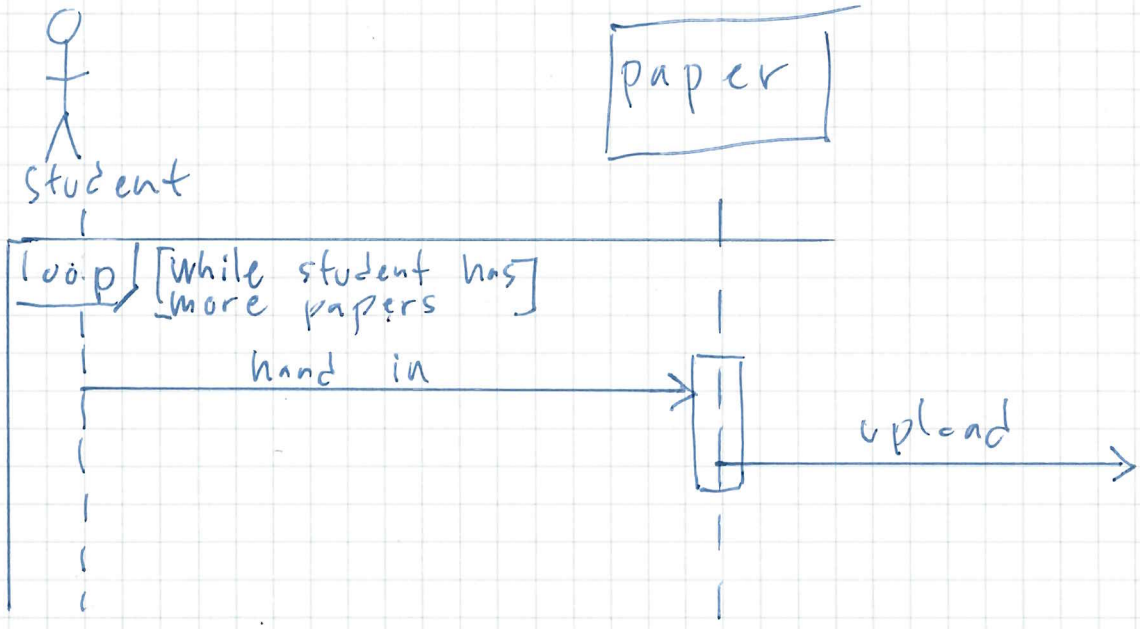
4



-2pts: missing some activation lines

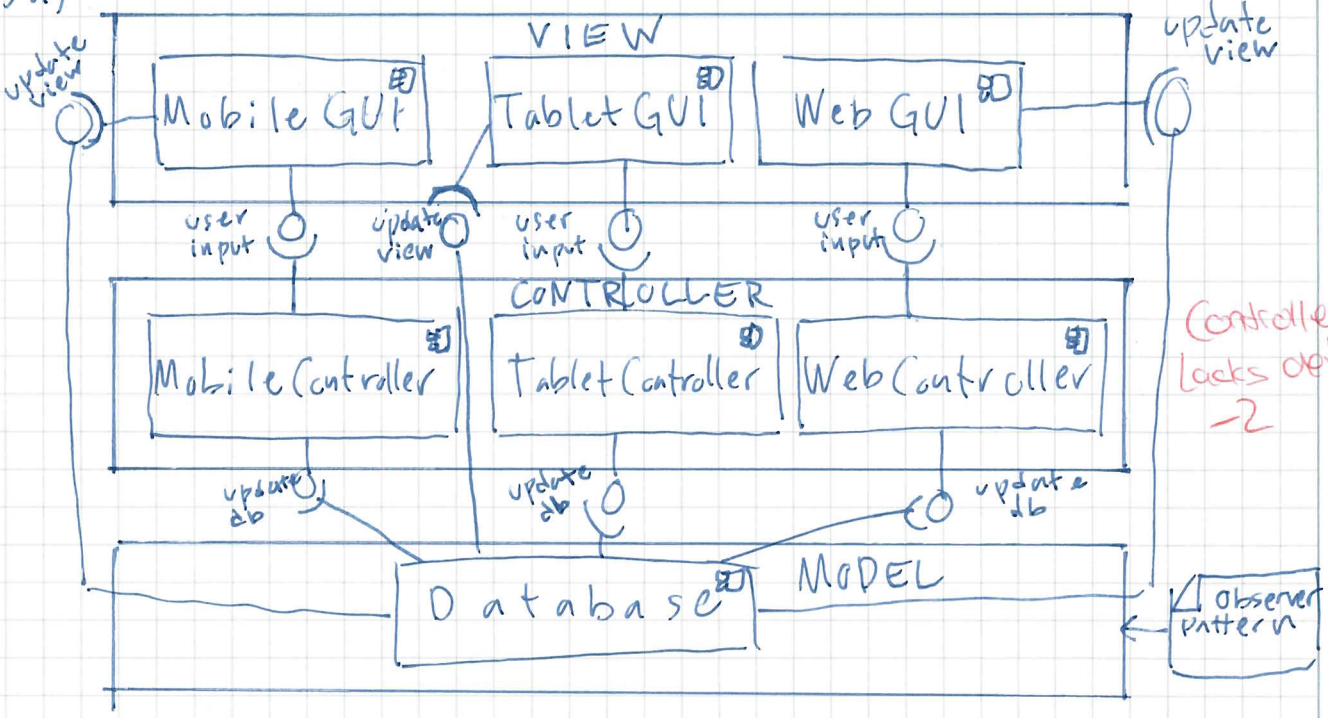
Clarification:

Due to lack of space the loop and the first action "hand in" are a bit smashed together. This is what it should look like:



3

5a)



Controller locks obtain -2

5b) The submission system is implemented with Model View Controller architectural style.

The user inputs their file in whatever platform they use.

The controller takes the input and updates the database.

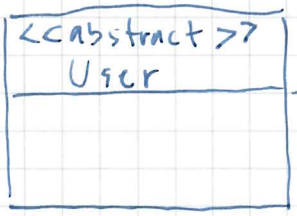
The database is implementing the Observer pattern, so it notifies the views of the change and then View displays it. 2

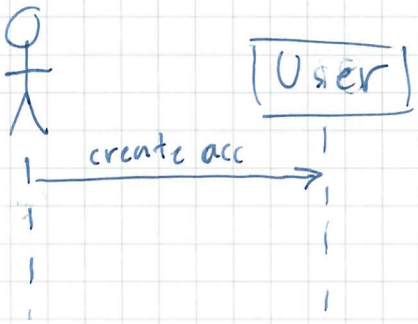
This architectural style is good for separating the user from the inner workings of the system. It is also modular since you can add new components at any layer of the system.

3

6 a)

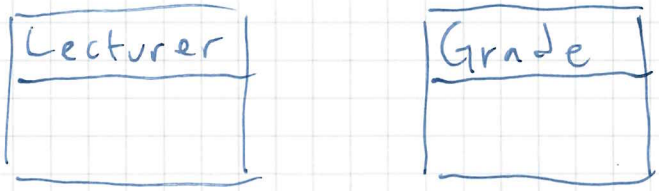
1.

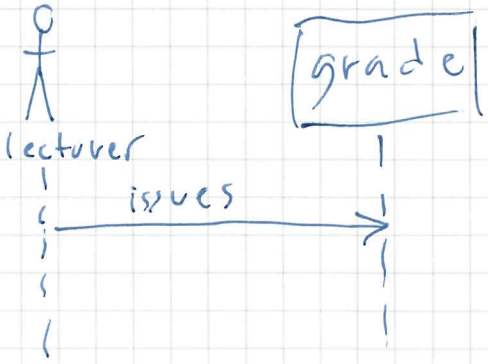




Violation of rule 1. User cannot be abstract in sequence diagram.

2.





Violation of rule 5. Lecturer has no association with grade in class diagram, but clearly needs to have one.

6 b) The steps needed to make the domain model into a class diagram and a design model is:

1. Give attributes specified types.
2. List methods for each class.
3. Remove textual explainations like "has-a" and replace them with symbols for aggregations, composition etc.
4. Give methods and attributes encapsulation indicators like "-", "+" and "#".
5. To ensure that the class diagram is consistent with the sequence diagram:

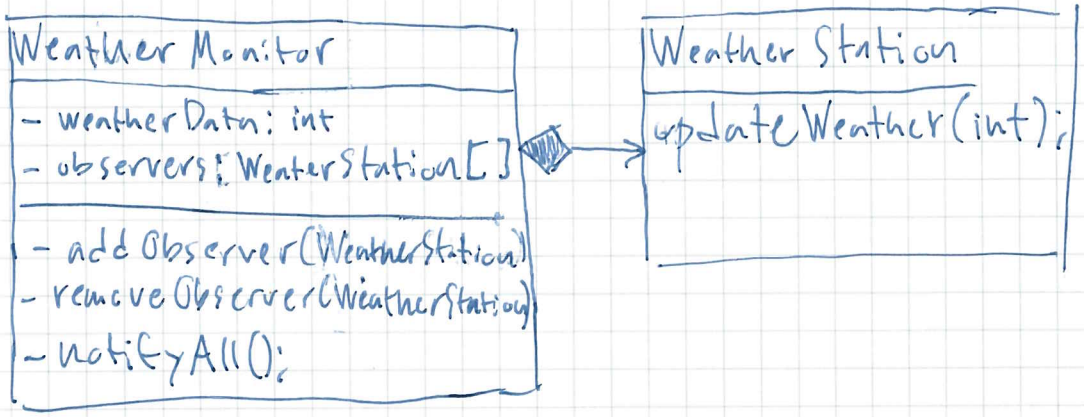
Semantic consistency: Every entity should have same name.
Syntactic: Follow UML notation
Structural: The structure of classes remain the same.
Behavioural: Classes and entities behave in the same way.

Great!

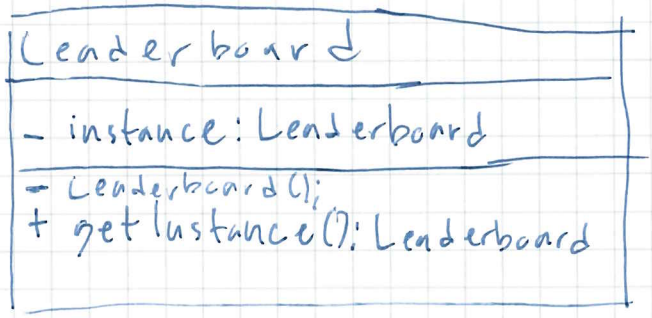
10

3

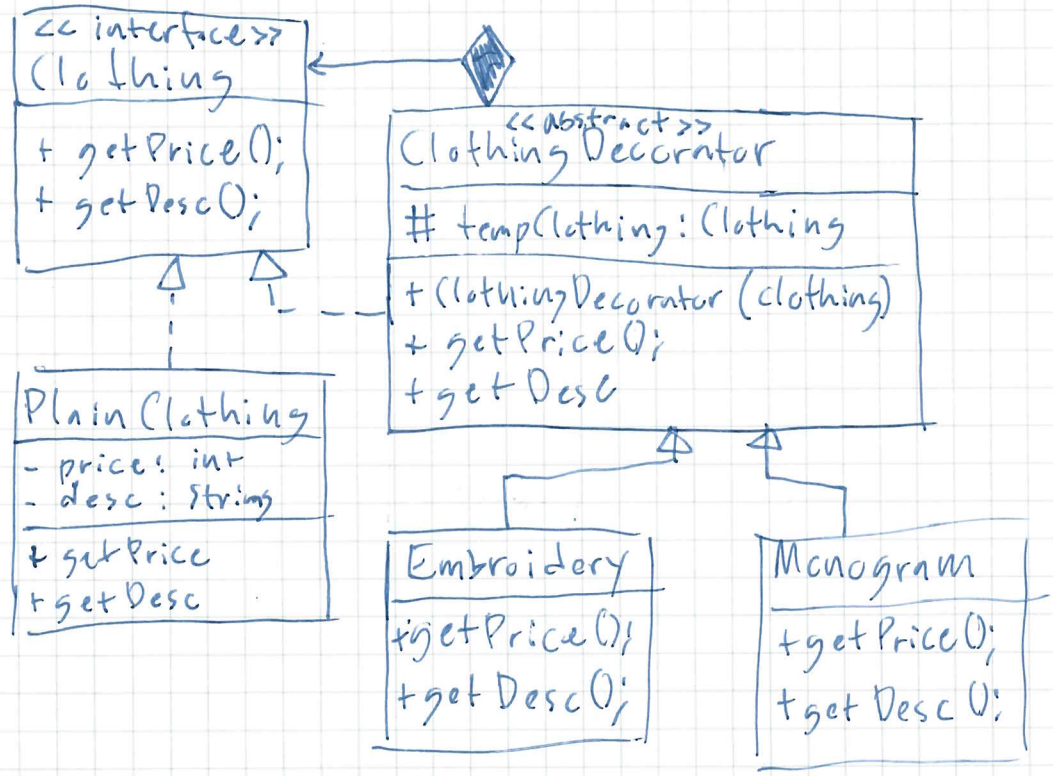
7 Case 1: Observer Pattern



Case 2: Singleton Pattern



Case 3: Decorator Pattern



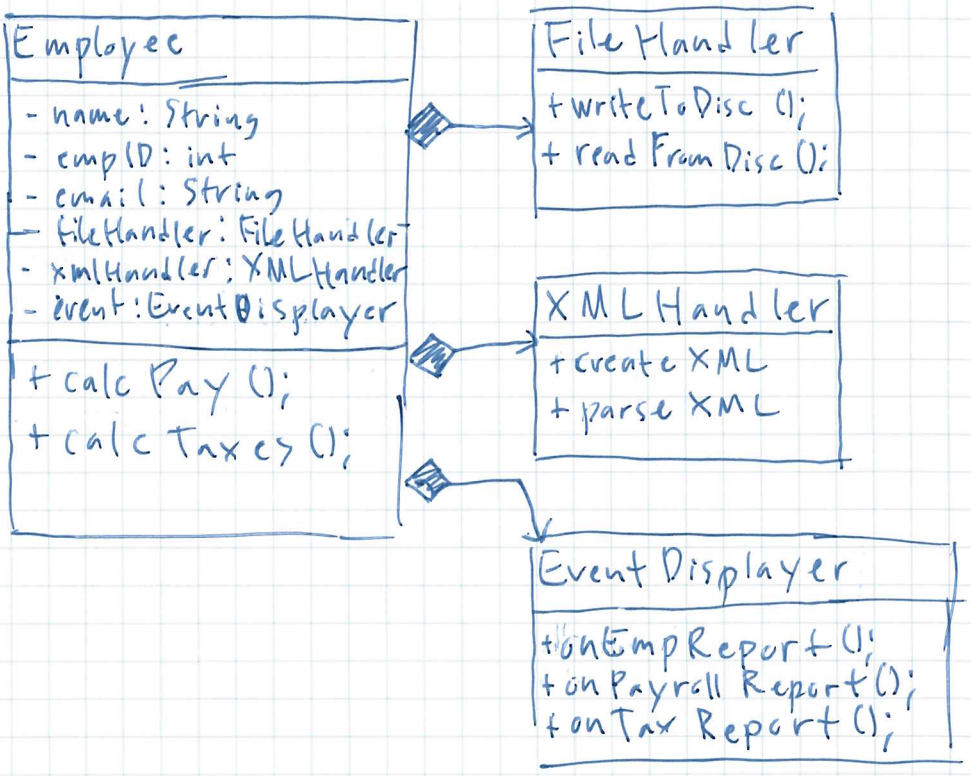
3

8 a) 1. The class violates the single responsibility principle. The employee should not conceptually be responsible for reading/writing or parsing XML files.

2. The class is not cohesive. The methods in the class does not conceptually belong to an employee.

3. Not modular. The class should be split up to allow the system to be more modular and allow for the classes to be used in other parts of the system. +

b)



-2 no multip. relations

c) My improved class dingram separates the classes so that they only preform actions that they should be responsible for.

This makes the system more modular and the client can call methods from classes at any time they are needed.

This design opens up the possibility of inheritance, The employee could have subclasses like manager, and the FileHandler could be made to have subclasses for different input-types.

13