2. semester efterår 2015: Opgaver i UML og JUnit

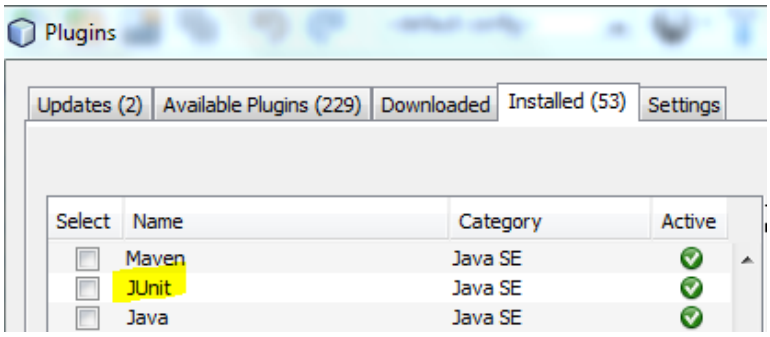
De følgende opgaver skal ikke afleveres, men er træningsopgaver, der skal forbedre jeres færdigheder i at udarbejde

- UML modeller over programdesign
- automatiseret JUnit test af klasser og deres metoder

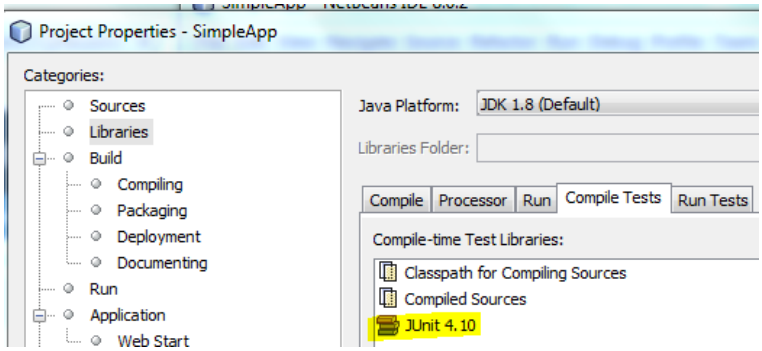
Vi gennemgår jeres løsningsforslag og evt. udfordringer med at løse opgaverne på fredag. Der er mulighed for sporadisk vejledning torsdag, men der er ingen undervisning denne dag og der føres derfor heller protokol.

I UML opgaverne, kan I vælge at håndtegne UML diagrammerne eller bruge et UML værktøj, for eksempelvis Software Ideas Modeler (til Windows), UMLet, UMLetino (online), easyUML (NetBeans plugin til klassediagrammer) – der findes mange flere derude ☺

Til JUnit opgaverne skal I sørge for at JUnit framework er installeret i NetBeans (det følger normalt med installation af NetBeans, hvilket kan tjekkes i hovedmenuen under Tools → Plugins →Installed :

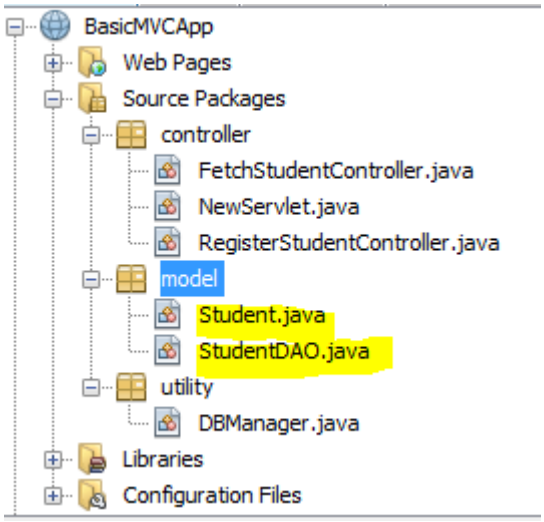


Når man opretter en JUnit test i et Java projekt, bliver man spurgt om hvilken version af JUnit, som man vil bruge. Vælg altid version 4. Herefter kommer JUnit biblioteket automatisk med i projektet:



Opgave 1 – UML klassediagram

Lav et UML klassediagram over klasserne i pakken `model` i projektet `BasicMVCApp` , som I finder på Fronter i DATSTUD rummet i mappen Architecture → Datasourcelayer:



Kort fortalt skal I modellere Java klasserne `Student.java` og `StudentDAO.java` i UML med:

- attributter (navn og datatype)
- metoder (navn, parametre og returtype)
- visibility (private hhv. public)
- associationer (husk pilretning, hvis det kun er den ene klasse, som har kendskab til relationen)
- multiplicitet

Når man anvender UML klassediagram til at modellere programdesign, kaldes det et klassedesigndiagram.

Opgave 2 – UML sekvensdiagram

Lav et UML sekvensdiagram over metoden `saveStudentDetails` i klassen `StudentDAO.java` i projektet `BasicMVCApp` (samme projekt som opgave 1).

Opgave 3 – UML sekvensdiagram

Lav et UML sekvensdiagram over metoden `fetchAllStudent`i klassen `StudentDAO.java` i projektet `BasicMVCApp` (samme projekt som opgave 1).

Opgave4 – JUnit

Lav en utility klasse med en metode som kan transformere en talkode til ugedag.
1 = mandag; 2 = tirsdag; 3 = onsdag; 4 = torsdag; 5 = fredag; 6 = lørdag; 7 = søndag.

```
public String transformToWeekDay(int dayNumber)
```

Lav dernæst en JUnit test, som tester metoden.

Test også grænseværdierne, dvs. hvordan programmet reagerer på invalide input, som -2, 0 eller 8. Test cases på valide og invalide input skal skrives i hver sin test metode, men i samme JUnit testklasse.

Opgave 5 – Arkitektur

A.

Byg en lille web applikation med en simpel login funktion:

Applikationens logiske arkitektur skal være lagdelt, dvs. Java projektet skal have en logisk adskilt pakkestruktur med en:

a) brugergrænseflade (UI med jsp og servlet)
b) applikationslogik og domæneobjekter, dvs. softwareobjekter, der repræsenterer domænebegreber (fx `Konto` klasse i et banksystem, som kan beregne saldo ved banktransaktioner)
c) persistensservice (interface til database med jdbc)

Brug opgaven som udgangspunkt for diskussion af det overordnede design i jeres kommende projekt.

Hint: Man behøver ikke bygge funktionalitet i kronologisk rækkefølge. Det er muligt at udarbejde og teste en login funktion uden at have implementeret en opret bruger funktion først. I så fald kan man oprette brugerdata manuelt i databasen, som login funktionen kan afprøves på.

B.

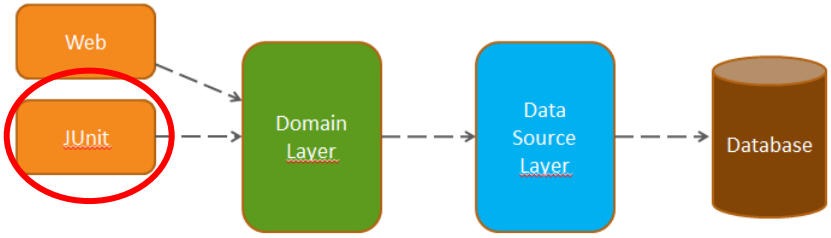
Udarbejd en arkitekturmodel over applikationen. Modellen behøver ikke være med UML syntaks, og udarbejdes ofte med farvelægning som eksemplet i bilag 1.

En anden mulighed er at anvende et UML pakkediagram (hvor hvert lag er angivet som en pakke med <<layer>> stereotype). Se eksempel i bilag 2.

Det er selvfølgelig også muligt at bruge et forsimplet UML klassediagram, hvor metoder og attributter udelades i klasserne og hvor der kun angives repræsentative klasser for hvert lag. Se eksempel i bilag 3.

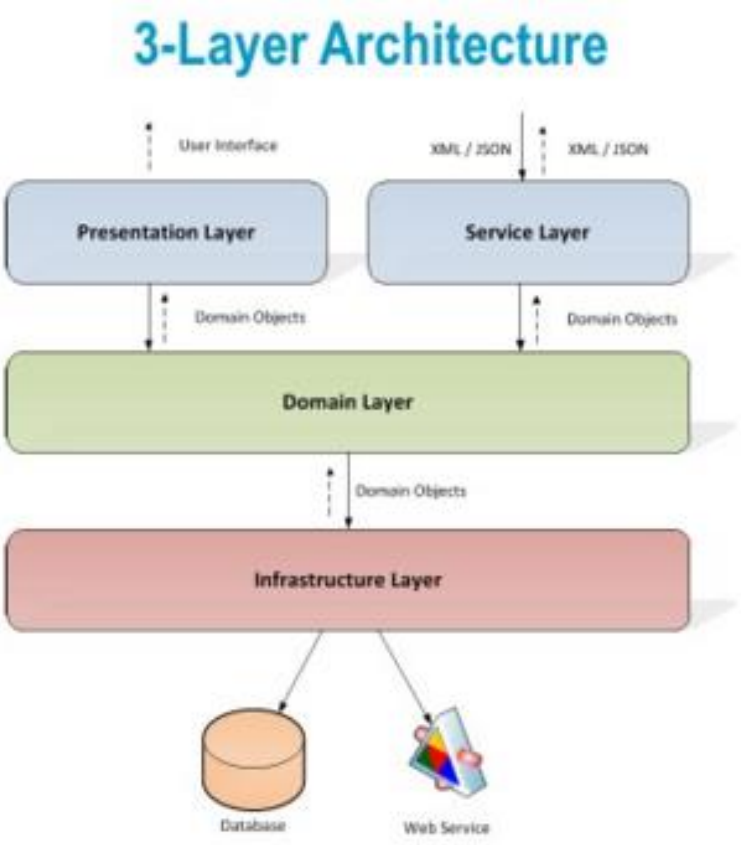
C.

Udskift præsentationslaget (UI) med en testklient (JUnit test), som automatisk tester login funktionen.

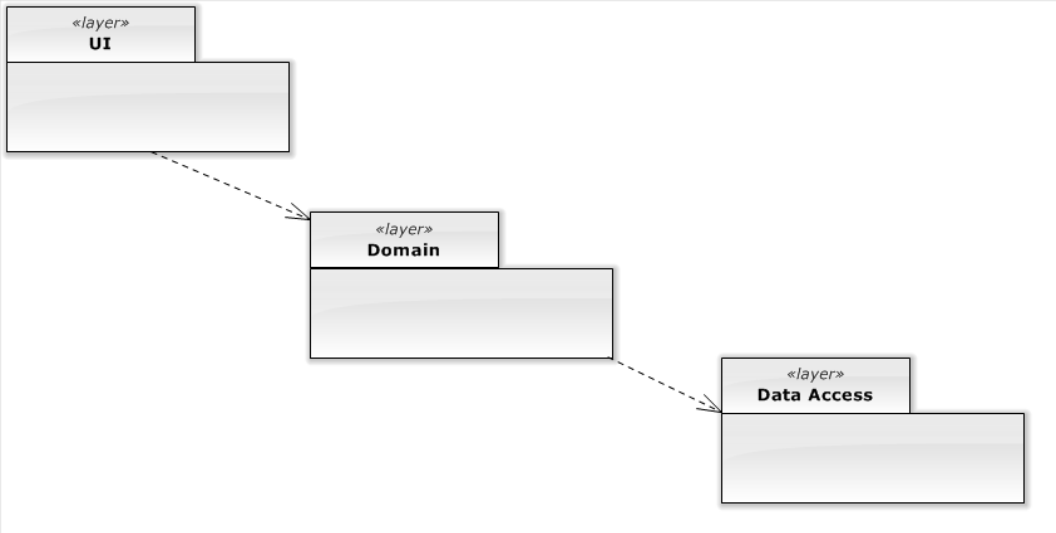


Husk at teste fejlscenarier, som fx indtastning af forkert brugernavn eller password.

Bilag 1 Arkitekturmodel eksempel uden UML



Bilag 2 Arkitekturmodel eksempel med UML Package diagram



Bilag 3 Arkitekturmodel eksempel med (forsimplet) UML klassediagram