1 Teori

Markera om följande påståenden är sanna eller falska:
(+1 för rätt svar, ingen förändring för fel svar)

Ett objekt är en instans av en viss klass.

- ○ Sant ✔
- ○ Falskt

Ett klassdiagram visar alla objekt som skapas av varje klass.

- ○ Falskt ✔
- ○ Sant

Ett use case diagram visar hur man använder en viss klass.

- ○ Sant
- ○ Falskt ✔

Ett klassdiagram visar alla attribut som varje klass har, men inte vilka värden varje attribut har.

- ○ Sant ✔
- ○ Falskt

Ett klassdiagram beskriver hur klasser och objekt samarbetar.

- ○ Falskt ✔
- ○ Sant

Interaktionsdiagram visar vilka metodanrop olika objekt gör på andra objekt.

- ○ Falskt
- ○ Sant ✔

Man gör ett interaktionsdiagram för varje systemhändelse.

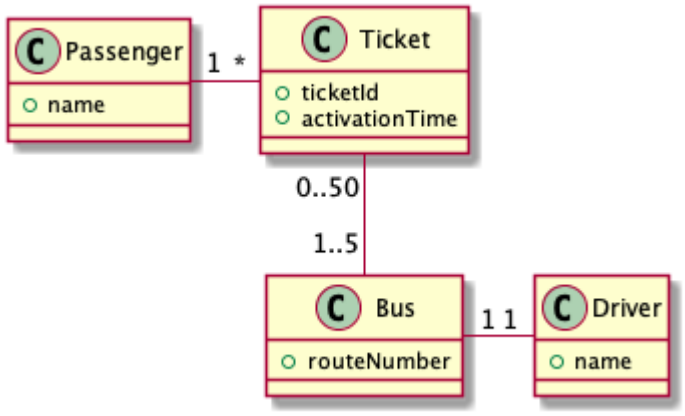
- ○ Sant ✔
- ○ Falskt

Design patterns beskriver hur man löser vanliga interaktioner med användarna av systemet i use cases.

- ○ Sant
- ○ Falskt ✔

Totalpoäng: 8

2 Klassdiagram



Klassdiagrammet beskriver en del av ett system för busstrafik.
För varje påstående nedan, markera om klassdiagrammet stödjer påståendet (sant) eller inte stödjer påståendet (falskt).

b1:Bus körs av happy:Driver

- Sant ✔
- Falskt

deadbeat:Driver åker med bussen b1:Bus, men kör inte.

- Sant
- Falskt ✔

per:Passenger har t1:Ticket.

- Falskt
- Sant ✔

per:Passenger tänker använda t1:Ticket till att först åka in till stan med b1:Bus, och sedan åka vidare till nästa stad med b2:Bus.

- Sant ✔
- Falskt

styrbjorn:Passenger har t2:Ticket till kustpilen:Train.

- ○ Falskt ✔
- ○ Sant

för att få high cohesion borde attributet "name" som finns både i Passenger och Driver brytas ut till en egen klass "Name".

- ○ Falskt ✔
- ○ Sant

happy:Driver och per:Passenger är bästa vänner så de passar på att prata lite när happy ser att per är med på bussen.

- ○ Sant
- ○ Falskt ✔

det finns inget sätt för per:Passenger att veta vilken :Bus som tar honom till stan.

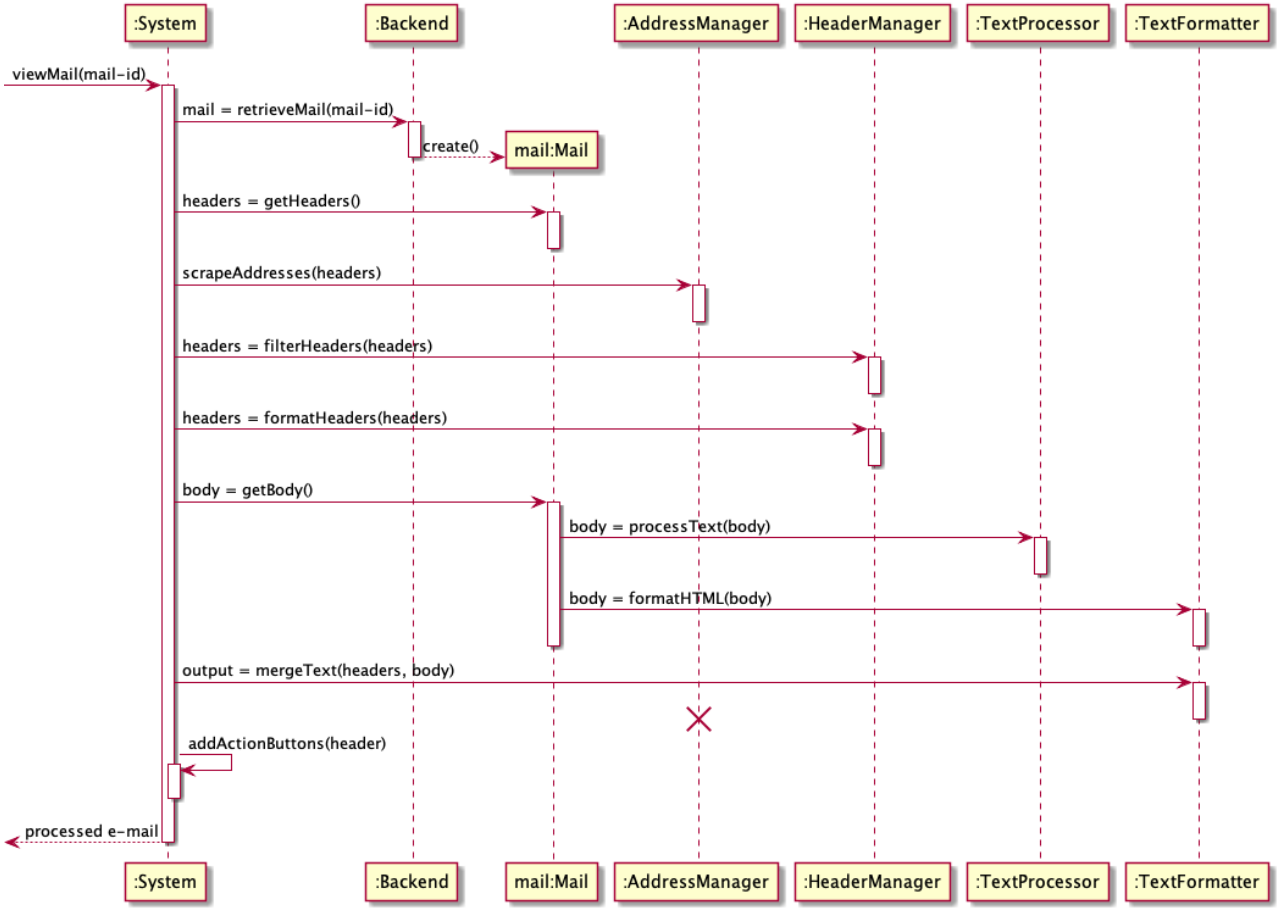
- ○ Sant ✔
- ○ Falskt

när en :Ticket har aktiverats på en :Bus så har man 24 timmar på sig att resa dit man skall.

- ○ Sant
- ○ Falskt ✔

Totalpoäng: 9

3 Interaktionsdiagram



Sekvensdiagrammet beskriver en del av ett mailprogram, specifikt vad som händer när man vill titta på ett visst mail.

Markera om följande påståenden är sanna eller falska:
(+1 för rätt svar, ingen förändring för fel svar)

:System har metoderna retreiveMail(), getHeaders(), scrapeAddresses(), filterHeaders(), formatHeaders(), getBody(), mergeText(), och addActionButtons().

- ○ Falskt ✔
- ○ Sant

anropet addActionButtons(header) måste gå till något annat objekt än :System

- ○ Sant
- ○ Falskt ✔

:Backend vet bara hur den skall hämta ett visst mail, men ingenting om vad mailet innehåller.

- Sant ✔
- Falskt

klassen System är en Controller för allt som skall göras med ett mail innan det kan visas.

- Falskt
- Sant ✔

enligt både High Cohesion och Low Coupling borde mail:Mail själv (och inte :System) se till att dess headers blir formatterade och filtrerade.

- Falskt
- Sant ✔

variabeln headers ligger sparad i :HeaderManager

- Falskt ✔
- Sant

:AddressManager har hängt sig och dör vid det stora X:et.

- Sant
- Falskt ✔

klassen Mail är Information Expert på allt som handlar om ett enskilt mail.

- Sant ✔
- Falskt

klasserna AddressManager och HeaderManager heter så eftersom de båda två ärver från den gemensamma basklassen Manager

- Falskt ✔
- Sant

klassen Mail är en Controller för vad som behöver göras med mail-innehållet (body).

- Falskt
- Sant ✔

Totalpoäng: 10

4 GRASP-mönster

Markera om följande påståenden är sanna eller falska:
(+1 för rätt svar, ingen förändring för fel svar)

En klass kan vara både en Creator och en Controller.

- Falskt
- Sant ✔

Det får bara finnas en instans av en Information Expert.

- Sant
- Falskt ✔

High Cohesion går ut på att varje klass skall ha så få och så välavgränsade ansvarsområden som möjligt.

- Sant ✔
- Falskt

Low Coupling går ut på att man skall sträva efter att ha så få och så "lösa" associationer som möjligt mellan klasser i ett system.

- Falskt
- Sant ✔

En Creator är en klass för att skapa slumptal.

- Falskt ✔
- Sant

Controllers kan anropa andra Controllers.

- Sant ✔
- Falskt

Enligt Pure Fabrication får inte klasser som ingår i Abstract Factory göra någonting annat.

- Sant
- Falskt ✔

Objektet main:GUIController, som är en Controller, ansvarar för att kontrollera att användaren använder gränssnittet rätt.

- Sant
- Falskt ✔

Objektet main:GUIController, som är en Controller, ansvarar för att skicka vidare händelser som användaren genererar mot gränssnittet till andra delar av applikationen som utför själva jobbet.

- Sant ✔
- Falskt

Totalpoäng: 9

5 Design Patterns

Markera om följande påståenden är sanna eller falska:
(+1 för rätt svar, ingen förändring för fel svar)

Ett Strategy pattern består av minst tre klasser med rollerna Context, AbstractStrategy, och ConcreteStrategy

- Falskt
- Sant ✔

Designmönstret Strategy Pattern handlar om att man vill kunna lösa en viss uppgift på olika sätt, så man behöver ha olika implementationer som kompilatorn kan hjälpa till att välja mellan.

- Sant ✔
- Falskt

Singleton använder sig av Pure Fabrication

- Sant
- Falskt ✔

Abstract Factory används för att skapa rätt typ av objekt givet ett visst kontext, där resten av systemet inte behöver veta exakt vilken typ objektet är.

- Sant ✔
- Falskt

Designmönstret Abstract Factory är bara ett specialfall av Strategy Pattern

- Sant ✔
- Falskt

Designmönstret Factory handlar om att all data (Facts) skall samlas i så få klasser som möjligt.

- ○ Falskt ✔
- ○ Sant

Singleton betyder att man bara får anropa klassen en gång

- ○ Sant
- ○ Falskt ✔

Totalpoäng: 7

6 Design Patterns II

a) I bokföringsprogrammet MyBucks kan man göra olika typer av analyser på sin ekonomi.

Systemet använder sig av designmönstret [Välj alternativ] (Strategy, Observer, State, Factory, Singleton) för att hantera dessa olika analyser.

b) En viss post (s.k. verifikation) i bokföringsprogrammet MyBucks kan vara öppen för redigering, sparad, repektive låst. Vissa funktioner är bara tillgängliga när verifikationen är öppen, en del är bara tillgängliga på sparade verifikationer. Låsta verifikationer kan man bara titta på, generera rapporter för, eller köra olika analyser på. För att hantera att olika funktioner är tillgängiga använder

sig MyBucks av designmönstret [Välj alternativ] (Singleton, State, Strategy, Factory, Observer).

c) Man kan konfigurera MyBucks så att olika tester körs när man sparar en verifikation (t.ex. kolla så att slutsumman blir rätt, att det finns tillräckligt med pengar kvar i kontantkassan, att rätt skatt

är dragen, osv.). MyBucks använder sig av designmönstret [Välj alternativ] (Factory, Observer, State, Strategy, Singleton) för att berätta att det finns en ny verifikation att köra tester mot.

Totalpoäng: 3

Betygsgränser

Betygsgränserna för den här tentan är:

Betyg	Procent	Poäng
MAX	100%	46
A	90%	41
B	80%	37
C	70%	32
D	65%	30
E	60%	27

Lycka till!