NTNU
Institutt for datateknikk
og informasjonsvitenskap

Eksamensoppgave i

TDT4100 – Objektorientert programmering

Fredag 19. august 2011, kl. 09:00 - 13:00

Oppgaven er utarbeidet av faglærer Hallvard Trætteberg og kvalitetssikrer Trond Aalberg.
Kontaktperson under eksamen Hallvard Trætteberg (mobil 918 97263)

Språkform: Bokmål
Tillatte hjelpemidler: C
Én valgfri lærebok i Java er tillatt.
Bestemt, enkel kalkulator tillatt.

Sensurfrist: Fredag 9. september

Les oppgaveteksten nøye. Finn ut hva det spørres etter i hver oppgave.

Dersom du mener at opplysninger mangler i en oppgaveformulering gjør kort rede for de antagelser og forutsetninger som du finner det nødvendig å gjøre.

Del 1 – Metoder (30%)

”Tre små trøndere på Høili plass” er en barnesang som synges litt spesielt. Første vers synges med ’a’-lyd for alle vokalene, altså ”Tra sma trandara pa…”, andre vers synges med ’e’-lyd osv. for alle vokalene i alfabetet. I denne oppgaven skal du lage en metode m/hjelpemetoder for å skrive ut sanger på denne formen. Metodene og evt. felt du ønsker å innføre samles i en felles klasse kalt **Song**.

a) Skriv en metode **isVowel(char c)** som returnerer **true** dersom **c** er en vokal og **false** ellers. Både store og små bokstaver må håndteres.

b) Skriv en metode **String computeVerse(String originalVerse, char v)** som tar inn originalverset og en vokal **v** som parametre. Metoden skal returnere verset, med vokalene i originalverset erstattet med den angitte vokalen **v**. Dersom **computeVerse** kalles med **”Alfabet”** og **’i’** som argument, så skal **”Ilfibit”** returneres. Merk at også her må store og små bokstaver håndteres riktig. Dersom **v** ikke er en vokal så skal det kastes et passende unntak.

c) Skriv en metode **writeSong(String originalVerse)** som tar inn originalverset som parameter og skriver ut alle versene, hvor hvert vers har alle vokalene erstattet med en bestemt vokal. Det skal altså skrives ut ett vers for hver vokal i alfabetet. Dersom **writeSong** kalles med **”Alfabet”** som argument, så skal **”Alfabat\nElfebet\nIlfibit\n…\nYlfybyt”** skrives ut (**\n** er et linjeskift mellom versene).

d) Du skal gjøre det mulig å skrive ut en sang med vers for et bestemt/begrenset sett med vokaler. Dette skal gjøres ved at **Song**-klassen instansieres med originalverset og **writeSong**-metoden kalles med settet av vokaler. Dersom en utfører **new Song(”Alfabet”).writeSong(”aei”)**, så skal **”Alfabat\nElfebet\nIlfibit”** skrives ut (**\n** er fortsatt linjeskift). Forklar med tekst og kode hvilke endringer som må gjøres for at dette skal være mulig.

Del 2 – Klasser (35%)

Du skal realisere et rating/poengsystem for et nettsted hvor personer kan publisere artikler/innlegg. Både personer og innlegg har en rating. Ratingen er et tall mellom 1 og 5 og starter som 1. De som leser innleggene kan gi 1 til 5 poeng til innleggene og/eller forfatterne for å øke eller redusere deres rating. Ratingen til både personer og innlegg er gjennomsnittet av poengene de har fått. Dersom et innlegg har fått totalt 10 poeng på 5 poenggivinger, vil innlegget ha 10 / 5 = 2 som rating.

Personer med høy rating regnes som viktigere enn de med lav rating og poengene som gis vektes derfor på følgende måte: Dersom en person med rating R gir P poeng, så regnes det som R poenggivinger à P poeng. Dersom innlegget i eksemplet over, med 10 poeng på 5 poenggivinger, får 4 poeng (P) fra en person med 1 som rating (R), så vil den nye ratingen være (10 + P*R) / (5+R) = 14/6 = 2,333.... Ratingen øker altså fra 2 til 2,333... fordi samlet antall ratingpoeng øker fra 10 til 14 og antall poenggivinger fra 5 til 6. Dersom innlegget istedenfor får 4 poeng (P) fra en person med 3 som rating (R), så vil den nye ratingen være (10 + P*R) / (5+R) = 22/8 = 2,75. Ratingen øker altså fra 2 til 2,75 fordi samlet antall ratingpoeng øker fra 10 til 22 og antall poenggivinger fra 5 til 8.

Rating av personer håndteres på samme måte som for innlegg.

a) Implementer klassen **Person** med *felt* og *innkapslingsmetoder* for navn og nødvendige *felt* for å håndtere ratingen. For å endre ratingen skal **Person**-klassen kun ha én public-metode, **receivePoints(int points, Person giver)**, som kan brukes når personen **giver** gir **points** poeng til

Person-objektet som metoden kalles på. Dersom **randi** og **jon** refererer til hvert sitt **Person**-objekt, så skal altså kallet **randi.receivePoints(5, jon)** oppdatere **randi** sin rating iht. de 5 poengene og **jon** sin rating, som beskrevet over. Husk å validere argumentene!

b) Du skal implementere en **Submission**-klasse (innlegg) og tenker at du kan spare kode ved å flytte kode for å håndtere rating til en egen klasse og (gjen)bruke denne vha. arv. Implementer en slik løsning.

c) Delegering er en fleksibel teknikk for å (gjen)bruke nyttig funksjonalitet i andre klasser. Vis hvordan kode for å håndtere rating kan samles i en egen klasse, som klassene **Person** og **Submission** kan delegere til, istedenfor å bruke arv.

Del 3 – Testing (15%)

a) Forklar med tekst og kode hvordan du vil teste hver av metodene i klassen **Song** fra oppgave 1 a-c.

b) Forklar med tekst og kode hvordan du vil teste **Song**-klassen fra oppgave 1 d).

Del 4 – Java-forståelse (20%)

a) I Java må alle deklarasjoner av felt, parametre og variabler inneholde en type, f.eks. **int a**. Hva er hensikten med deklarasjon av typer i et programmeringsspråk?

b) Hva skrives ut av følgende kode? Forklar hvorfor!
System.out.println(”Hallvard sin alder er ” + 4 * 10 + 4);

c) I tabellen vises innmaten til tre varianter av klassen **A**. Dersom en utfører **new A()** for hver variant, får en utskriften som vist under, hhv. ”1”, ”2” og ”1”. Forklar hvorfor!

private int i = 0; **private int foo() {** ** i = i + 1;** ** return i;** **}** **public A() {** ** if (foo() != 0 || foo() == 0) {** ** System.out.println(i);** ** }** **}**	**private int i = 0;** **private int foo() {** ** i = i + 1;** ** return i;** **}** **public A() {** ** if (foo() == 0 || foo() != 0) {** ** System.out.println(i);** ** }** **}**	**private int i = 0;** **private int foo() {** ** return i++;** **}** **public A() {** ** if (foo() == 0 || foo() != 0) {** ** System.out.println(i);** ** }** **}**
1	2	1

d) Anta følgende metode-deklarasjon:

```
int foo(int i) {
  if (i > 0) {
    System.out.println(”i: ” + foo(i - 1));
  }
  return i;
}
```

Hva blir skrevet ut dersom en utfører **foo(2)**? Forklar hvorfor!

NTNU
Institutt for datateknikk
og informasjonsvitenskap

Exam for

TDT4100 – Object-oriented programming

Friday 19. August 2011, 09:00 - 13:00

The exam is made by responsible teacher Hallvard Trætteberg and quality assurer Trond Aalberg.
Contact person during the exam is Hallvard Trætteberg (mobile 918 97263)

Language: English
Supporting material: C
One printed Java textbook is allowed. Other printed books are not allowed.

Deadline for results: Friday 9. September.

Read the text carefully. Make sure you understand what you are supposed to do.

If information is missing you must clarify what assumptions you find necessary.

Note the percentages for each part, so you use your time wisely.

Part 1 – Methods (30%)

”Three little Trondheim’ers at Høili place” is a children song with a special singing ”logic”. First verse is sung with an’a’-sound for all the vowels, i.e.” Thraa lattla Trandhaam’ars at Haala placa …”, the second verse is sung with an ’e’-sound etc. for all the vowels in the alphabet. In this part you must write methods w/helper methods for printing songs with this logic. The methods and fields you need are all put in the **Song** class.

a) Write the method **isVowel(char c)** which returns **true** if **c** is a vowel and **false** otherwise. Both upper and lower case letters must be supported.

b) Write a method **String computeVerse(String originalVerse, char v)** which receives the original verse and a vowel **v** as parameters. The method must return the verse, with the vowels in the original replaced by the provided vowel **v**. If **computeVerse** is called with **”Alfabet”** and **’i’** as arguments, then **”Ilfibit”** must be returned. Note that upper and lower case letters must be handled correctly. If **v** isn’t a vowel, a suitable exception must be thrown.

c) Write a method **writeSong(String originalVerse)** which receives the original verse as parameter and prints all verses, where each verse has all the vowels replaces by a specific vowel. I.e. one verse pr. vowel in the alphabet must be printed. If **writeSong** is called with**”Alfabet”** as argument, then **”Alfabat\nElfebet\nIlfibit\n…\nYlfybyt”** must be printed (**\n** is newline between the verses).

d) You must make it possible to print a song for a specific/limited set of vowels. It should be possible to do this by instantiating the **Song** class with the original verse and calling the **writeSong**-method with the desired set of vowels. If **new Song(”Alfabet”).writeSong(”aei”)** is executed, then **”Alfabat\nElfebet\nIlfibit”** must be printed (**\n** is still newline). Explain with text and code what changes must be made to enable this.

Part 2 – Classes (35%)

You must implemet a rating system for a web site where people can post submissions. Both people and submissions have a rating. The rating is a number between 1 and 5 that starts out as 1. The readers can give from 1 to 5 points to the submissions and/or authors to increase or decrease their rating. The rating of a person or submission is the average of all the points they have received. If a submission has received a total of 10 points from 5 ratings, the submission will have 10/5=2 as rating.

People with high rating are considered more important than those with lower rating and the points they give are weighted as follows: If a person with rating R gives P poeng, it is counted as if P points are given R times. If the submission in the example above, with 10 poeng after 5 ratings, receives 4 points (P) from a person with 1 as rating (R), then the new rating will be (10 + P*R) / (5+R) = 14/6 = 2,333.... I.e. the rating will increase from 2 to 2,333... because the total rating points increases from 10 to 14 and the rating count increases from 5 to 6. If the person instead has 3 as rating (R), then the new rating will be (10 + P*R) / (5+R) = 22/8 = 2,75. I.e. the rating will increase from 2 to 2,75 because the total rating points increases from 10 to 22 and the rating count increases from 5 to 8.

Rating of people is handled the same way as for submissions.

a) Implement the class **Person** with *fields* and *encapsulation methods* for name and necessary *fields* for managing the rating. There will only be one method, **receivePoints(int points, Person giver)**, for

changing the rating. This method is used when the person **giver** gives **points** points to the **Person** object that the method is invoked on. If **randi** and **jon** refer to **Person** objekts, then the call **randi.receivePoints(5, jon)** must **randi**'s rating according to the 5 points and **jon**'s rating, as explained above. Remember to validate the arguments!

b) You must implement a **Submission** class and hope to save code by moving the code for managing a rating to a separate class and (re)using this class using inheritance. Implement such a solution.

c) Delegation is a flexible technique for (re)using functionality from other classes. Show how code for managing a rating can be moved to a separate class, which the classes **Person** and **Submission** can delegate to, istead of using inheritance.

Part 3 – Testing (15%)

a) Explain with text and code how you would test each methods in the **Song** class from part 1 a-c).

b) Explain with text and code how you would test the **Song** class from part 1 d).

Del 4 – Java-forståelse (20%)

a) In Java all declarations of fields parameters and variables must include a type, e.g. **int a**. What is the purpose of including types in declarations in a programming language?

b) What is printed by the following code? Explain why!
System.out.println(”Hallvard's age is ” + 4 * 10 + 4);

c) The table shows the content of three variants of the class **A**. If **new A()** is executed for each variant, the content below is printed,”1”, ”2” og ”1”, respectively. Explain why!

private int i = 0; **private int foo() {** ** i = i + 1;** ** return i;** **}** **public A() {** ** if (foo() != 0 || foo() == 0) {** ** System.out.println(i);** ** }** **}**	**private int i = 0;** **private int foo() {** ** i = i + 1;** ** return i;** **}** **public A() {** ** if (foo() == 0 || foo() != 0) {** ** System.out.println(i);** ** }** **}**	**private int i = 0;** **private int foo() {** ** return i++;** **}** **public A() {** ** if (foo() == 0 || foo() != 0) {** ** System.out.println(i);** ** }** **}**
1	2	1

d) Give the following method declaration:

```
int foo(int i) {
  if (i > 0) {
    System.out.println(”i: ” + foo(i - 1));
  }
  return i;
}
```

What is printed if **foo(2)** is executed? Explain why!

NTNU
Institutt for datateknikk
og informasjonsvitenskap

Eksamensoppgåve i

TDT4100 – Objektorientert programmering

Fredag 19. august 2011, kl. 09:00 - 13:00

Oppgåva er utarbeidd av faglærar Hallvard Trætteberg og kvalitetssikra av Trond Aalberg.
Kontaktperson under eksamen Hallvard Trætteberg (mobil 918 97263)

Språkform: Nynorsk
Tillatne hjelpemiddel: C
Éi valfri lærebok i Java er tillaten.
Bestemt, enkel kalkulator tillaten.

Sensurfrist: Fredag 9. september.

Les oppgåveteksten nøye. Finn ut kva det er spurt etter i kvar oppgåve.

Dersom du meiner at opplysningar manglar i ei oppgåveformulering, gjer kort greie for dei føresetnadene som du finn naudsynte.

Del 1 – Metodar (30%)

”Tre små trøndarar på Høili plass” er ein barnesong som blir sunge på ein spesiell måte. Første vers har ’a’-lyd for alle vokalane, altså ”Tra sma trandarar pa…”, andre vers har ’e’-lyd, osb. for alle vokalane i alfabetet. I denne oppgåva skal du lage ein metode m/hjelpemetodar for å skrive ut songar på denne forma. Metodane og evt. felt du ønskjer å innføre skal samlast i ein sams klasse kalla **Song**.

a) Skriv ein metode **isVowel(char c)** som returnerer **true** dersom **c** er ein vokal og **false** elles. Både store og små bokstavar må handterast.

b) Skriv ein metode **String computeVerse(String originalVerse, char v)** som tar inn originalverset og ein vokal **v** som parametarar. Metoden skal returnere verset, med vokalane i originalverset erstatta med den gitte vokalen **v**. Dersom **computeVerse** blir kalla med **”Alfabet”** og **’i’** som argument, så skal **”Ilfibit”** returnerast. Merk at også her må store og små bokstavar handterast riktig. Dersom **v** ikkje er ein vokal, så skal det kastast eit passande unntak.

c) Skriv ein metode **writeSong(String originalVerse)** som tar inn originalverset som parameter og skriv ut alle versa, der kvart vers har alle vokalane erstatta med ein bestemt vokal. Det skal altså skrivast ut eitt vers for kvar vokal i alfabetet. Dersom **writeSong** kallast med **”Alfabet”** som argument, så skal **”Alfabat\nElfebet\nIlfibit\n…\nYlfybyt”** skrivast ut (**\n** er eit linjeskift mellom versa).

d) Du skal gjøre det mogeleg å skrive ut ein song med vers for eit bestemt/avgrensa sett med vokalar. Dette skal gjerast ved at **Song**-klassen blir instansiert med originalverset og **writeSong**-metoden blir kalla med settet av vokalar. Dersom ein utfører **new Song(”Alfabet”).writeSong(”aei”)**, så skal **”Alfabat\nElfebet\nIlfibit”** skrivast ut (**\n** er framleis linjeskift). Forklar med tekst og kode kva endringar som må gjerast for at dette skal være mogeleg.

Del 2 – Klassar (35%)

Du skal realisere eit rating/poengsystem for ein nettstad der personer kan publisere artiklar/innlegg. Både personar og innlegg har ein rating. Ratingen er eit tal mellom 1 og 5 og startar som 1. Dei som les innlegga kan gi 1 til 5 poeng til innlegga og/eller forfattarane for å auke eller redusere ratingen deira. Ratingen til både personar og innlegg er gjennomsnittet av poenga dei har fått. Dersom eit innlegg har fått totalt 10 poeng på 5 poenggjevingar, vil innlegget ha 10 / 5 = 2 som rating.

Personar med høg rating blir rekna som viktigare enn dei med låg rating, og poenga som blir gitt blir derfor vekta på følgjande måte: Dersom ein person med rating R gir P poeng, så blir det rekna som R poenggjevingar à P poeng. Dersom innlegget i dømet over, med 10 poeng på 5 poenggjevingar, får 4 poeng (P) fra en person med 1 som rating (R), så vil den nye ratingen vere (10 + P*R) / (5+R) = 14/6 = 2,333.... Ratingen aukar altså frå 2 til 2,333... fordi summen av ratingpoeng aukar frå 10 til 14 og talet på poenggjevingar frå 5 til 6. Dersom innlegget i staden får 4 poeng (P) frå ein person med 3 som rating (R), så vil den nye ratingen vere (10 + P*R) / (5+R) = 22/8 = 2,75. Ratingen aukar altså frå 2 til 2,75 fordi summen av ratingpoeng aukar frå 10 til 22 og talet på poenggjevingar frå 5 til 8.

Rating av personer blir handtert på same måte som for innlegg.

a) Implementer klassen **Person** med *felt* og *innkapslingsmetodar* for namn og naudsynte *felt* for å handtere ratingen. For å endre ratingen skal **Person**-klassen berre ha éin public-metode,

receivePoints(int points, Person giver), som kan brukast når personen **giver** gir **points** poeng til **Person**-objektet som metoden blir kalla på. Dersom **randi** og **jon** refererer til kvart sitt **Person**-objekt, så skal altså kallet **randi.receivePoints(5, jon)** oppdatere **randi** sin rating iht. de 5 poenga og **jon** sin rating, som skildra over. Hugs å validere argumenta!

b) Du skal implementere ein **Submission**-klasse (innlegg) og tenkjer at du kan spare kode ved å flytte kode for å handtere rating til ein eigen klasse og (gjen)bruke denne vha. arv. Implementer ei slik løysing.

c) Delegering er ein fleksibel teknikk for å (gjen)bruke nyttig funksjonalitet i andre klassar. Vis korleis kode for å handtere rating kan samlast i en eigen klasse, som klassane **Person** og **Submission** kan delegere til, i staden for å bruke arv.

Del 3 – Testing (15%)

a) Forklar med tekst og kode korleis du vil teste kvar av metodane i klassen **Song** frå oppgåve 1 a-c.

b) Forklar med tekst og kode korleis du vil teste **Song**-klassen frå oppgåve 1 d).

Del 4 – Java-forståing (20%)

a) I Java må alle deklarasjonar av felt, parameterar og variable innehalde en type, t.d. **int a**. Kva er føremålet med deklarasjon av typar i eit programmeringsspråk?

b) Kva blir skrive ut av følgjande kode? Forklar kvifor!
System.out.println(”Hallvard sin alder er ” + 4 * 10 + 4);

c) I tabellen har vi vist innmaten til tre variantar av klassen **A**. Dersom ein utfører **new A()** for kvar variant, får ein utskrifta som vist under, hhv. ”1”, ”2” og ”1”. Forklar kvifor!

private int i = 0; **private int foo() {** ** i = i + 1;** ** return i;** **}** **public A() {** ** if (foo() != 0 || foo() == 0) {** ** System.out.println(i);** ** }** **}**	**private int i = 0;** **private int foo() {** ** i = i + 1;** ** return i;** **}** **public A() {** ** if (foo() == 0 || foo() != 0) {** ** System.out.println(i);** ** }** **}**	**private int i = 0;** **private int foo() {** ** return i++;** **}** **public A() {** ** if (foo() == 0 || foo() != 0) {** ** System.out.println(i);** ** }** **}**
1	2	1

d) Anta følgjande metode-deklarasjon:

```
int foo(int i) {
  if (i > 0) {
    System.out.println(”i: ” + foo(i - 1));
  }
  return i;
}
```

Kva blir skrive ut dersom ein utfører foo(2)? Forklar kvifor!