Linköpings Tekniska Högskola
Institutionen för Datavetenskap
Magnus Nielsen och Emma Enocksson Svensson

27 augusti 2021

TDDE10 & 725G90
Objektorienterad programmering i Java
Datortentamen (DAT1)
2021-08-27, 08–12

Examinator:	Torbjörn Jonsson
Jour:	Magnus Nielsen (telefon: 013-28 58 86)
Antal uppgifter:	1
Max poäng:	20 poäng
TDDE10:	Betyg 3 = 10p, 4 = 14p, 5 = 18p.
725G90:	Betyg G = 10p, VG = 16p.
Hjälpmedel:	Eclipse, Kurswebbsidan samt Oracle's api-dokumentation.
Tidsgränser:	Ordinarie: 08:00-12:00, förlängd tid: 08:00-13:30.

REGLER

- Tentan genomförs individuellt.
- Du ska sitta i en ostörd miljö utan andra personer i samma rum. Du ska hela tiden vara uppkopplad och synlig i Zoom videosamtal. Ombersörj att ditt ansikte alltid är synligt. Under eventuella toalettbesök sätter du bara upp en skylt eller ett papper som det står "RAST" på framför kameran, och skickar in en extrainlämning via sendlab enligt länken nedan.
- Du får lov att använda dig av Oracle's api-dokumentation samt TDDE10/725G90-kurshemsida (klickbara länkar). Alla andra internetresurser är strikt förbjudna.
- All kommunikation är förbjuden, undantaget frågor till kurspersonal.
- Var beredd på att i efterhand kunna redogöra för dina svar. Vi kan komma att höra av oss till ett urval för att låta dem redogöra för sina lösningar / svar.
- Alla former av regelbrott medför att din tentamen underkänns direkt.
- Undvik väldigt gärna åäö helt i dina källkodsfiler, även i kommentarer. Det orsakar en hel del extra arbete att fixa till teckenkodning, undviker man åäö uppstår inte dessa problem. Ett tips är att skriva såväl källkod som kommentarer på engelska om du känner dig bekväm med det.

FRÅGOR

Frågor ställs genom funktionen “Ask for help” i Zoom. Tryck på knappen så kommer du att flyttas till ett privat rum tillsammans med en tekniskt kunnig jourperson. Jourtelefonen kan användas om du tappar internet eller om du har problem med Zoom.

BEDÖMNING

Bedömningen görs baserat på allt vi har gått igenom i kursen. Sådant som gav kompletteringar på laboration kommer att kunna ge poängavdrag på tentan. Ex: Följer ej kodstandard, abstraktionsfel, eller felaktig användning av de principer / koncept du tar med i koden. Avdragen är aldrig större än antalet poäng en princip / ett koncept ger, och du kan således inte förlora på att försöka.

INLÄMNING

Slutinlämningen av din kod ska vara skriven enligt god programmeringssed för Java. Det betyder att tekniker som finns för att underlätta användning, utveckling, felsökning och underhåll ska användas. Sendlabsidan för tentan hittar du på:
https://www.ida.liu.se/sendlab/student/send?code=tdde10_725g90_20210827_exam

- Slutinlämning senast kl 12:00 (13:30 förlängd tid)
- Inlämningen ska bestå av alla dina .java-filer, eller en zip-/tar-fil med samtliga .java filer du har skrivit.

Tänk på att det är upp till dig att visa så mycket kunskap du kan och att du på denna tenta har stor frihet att faktiskt visa allt du kan inom parametrarna för kursen. Håll i åtanke all feedback du har fått under kursens gång.

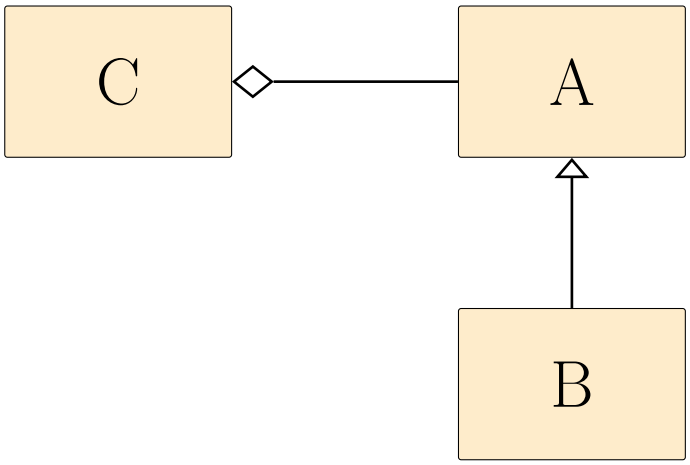
Lycka till!

Magnus Nielsen och Emma Enocksson Svensson

Uppgift

Läs hela uppgiftslydelsen (båda sidor) innan du börjar för att undvika onödiga poängtapp och stress. Du ska skapa ett program uppbyggt av (minst) tre klasser, du får gärna lägga till klasser om du känner behov därav. Klasserna ska vara starkt förknippade med temat **Lampor**. Ditt program ska demonstrera dina kunskaper om *konstruktorer*, *instansvariabler*, *instansmetoder* och *god inkapsling*. Ditt huvudprogram ska visa hur dina klasser används genom att testa varje del av respektive klass. Observera att du inte behöver vara expert inom ditt givna tema, men det är **obligatoriskt** att förhålla dig till det. Känner du att du inte har några kunskaper inom området går det utmärkt att hitta på, låt fantasin sätta rimliga gränser.

Obligatoriskt: Utöver detta ska klassernas struktur följa det enkla klassdiagrammet nedan:



Grundläggande struktur (6p)

Klasserna ska tillsammans visa prov på respektive kunskapsområde:

- *konstruktorer* minst två lämpliga icke-tomma (1p).
- *datamedlemmar (instansvariabler)*, minst en, som fyller ett syfte, per klass. (1p).
- *instansmetoder (metoder tillhörande en instans av klassen)*, minst en per klass (1p).
- *god inkapsling* (1p).
- *minst* en av instansmetoderna ska ta emot parametrar och utföra en för klassen relevant operation (1p). Att endast returnera eller modifiera en instansvariabel är för trivialt (getters och setters räknas alltså inte).
- Klasserna ska därtill uppfylla *klassdiagrammet* ovan. (1p).

Koncept / principer (12p)

Under kursens gång har vi behandlat och arbetat med många olika objektorienterade koncept. För att tjäna in ytterligare poäng på tentan måste du uppvisa förståelse för nedanstående koncept. Om full poäng ska ges för konceptet ska du använda det på ett *rimligt* sätt. Alltså: Det räcker inte med att du kan skriva XYZ (vilket koncept du nu har valt), utan det ska tjäna ett syfte / användas på ett rimligt sätt för full poäng. Du får fritt välja vilket eller vilka koncept/principer du väljer att ta med i din lösning.

- Polymorfi, både en klasstruktur som tillåter det samt *nyttjande därav*. (2p för uppfyllande av samtliga krav, 0p om du inte nyttjar det).
- Statiska variabler och användning därav, minst en rimlig metod som behandlar den statiska variabeln krävs och uppvisar förståelse för statiska variabler. (2p).
- Exception: Både skapande av *minst* ett eget Exception samt uppvisad användning därav. RuntimeException: 1p, Exception: 1p. För full poäng måste du skapa ett av varje sort, samt använda dem rimligt och visa att du förstår skillnaden mellan dem i din källkod. (2p).
- Multipla konstruktorer (flera *rimliga* konstruktorer i en och samma klass). (2p).
- Abstrakta metoder (minst en, samt överskuggning av densamma). (2p).
- Generiska klasser, skapande av egen generisk klass (1p) samt användning av densamma (1p). Totalt: (2p).

I vissa fall kan det vara rimligt att visa konceptet i huvudprogrammet (main), i andra fall kan det vara rimligare att visa det i klasserna, beroende på vilket (eller vilka) koncept just du har valt. Det ska tjäna ett syfte för klassen (eller klasserna), men det behöver inte nödvändigtvis vara vettigt utifrån temat. Om du t.ex. har skivit klassen `Date (datum)` och ska implementera division och inte kommer på vad division betyder för datum så är det okej att din division räknar ut något annat, t.ex. hur många år det är mellan datumen. Kommentera dock gärna om det inte är självklart / tydligt som i divisionsexemplet.

Teorifråga (2p)

Denna del är obligatorisk. Det krävs att du gör en ansats att besvara frågan. Du måste inte ha svarat rätt, men för godkänt krävs det att du har försökt. Teorifrågan kan du besvara antingen i en egen textfil, eller som kommentar i din main fil (eller varför inte i en print i main-filen).

Fråga: Redogör kortfattat för överskuggning (override): Vad innebär det och hur fungerar det?