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

22 mars 2023

TDDE10 & 725G90
Objektorienterad programmering i Java
Datortentamen (DAT1)
2023-03-23, 14–18

Examinator:	Emma Enocksson Svensson (telefon: 013-28 18 08)
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 samt Oracle's api-dokumentation.
Tidsgränser:	Ordinarie: 14:00-18:00, förlängd tid: 14:00-19:30.

REGLER

- Tentan genomförs individuellt.
- Du får lov att använda dig av Oracle's api-dokumentation (klickbar länk). Alla andra internetresurser är strikt förbjudna.
- All kommunikation är förbjuden, undantaget frågor till kurspersonal.
- Alla former av regelbrott medför att din tentamen underkänns direkt.
- Unvik 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 i första hand genom "send question" i tentaklienten. Vid tekniska problem räcker du upp handen så kommer teknisk jour och hjälper till. Frågor relaterade till tentan, eller delar av densamma, kommer ej att besvaras i sal, utan sköts via tentaklienten.

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. Du skickar in din lösning via tentaklienten (send assignment). Den praktiska uppgiften skickar du in som *Assignment1*, och den teoretiska som *Assignment2*.

- Slutinlämning senast kl 18:00 (19:30 förlängd tid)
- Den praktiska inlämningen ska bestå av alla dina .java-filer (om du använder Eclipse ligger de i src-mappen), eller en zip-/tar-fil med samtliga .java filer du har skrivit.
- Den teoretiska inlämningen ska bestå av en textfil, du väljer själv rimligt format (.odt, .txt, .org, ...)

Viktigt! Om du får responsen "Assignment# Request received" vet du med säkerhet att din inlämning har kommit in. Får du inte denna respons bör du starta om din tentaklient och skicka in på nytt. Du kan kryssa ner den och starta den igen med "fisk-länken" på desktop.

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.

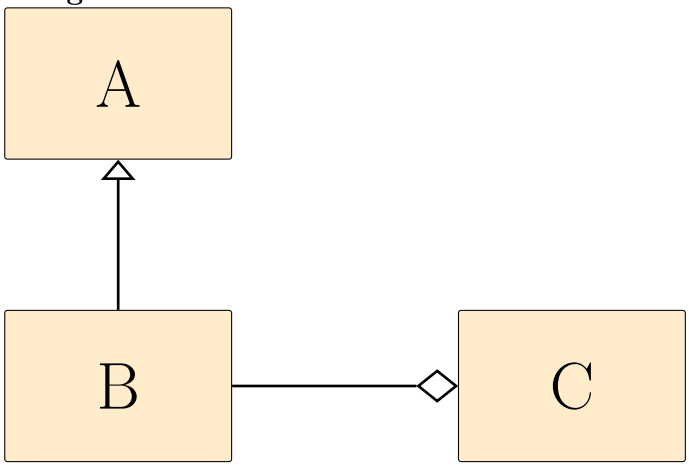
Lycka till!

Emma Enocksson Svensson och Magnus Nielsen

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 **maträtter**. 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 (7p)

Klasserna ska tillsammans visa prov på respektive kunskapsområde:

- *konstruktorer*, minst två (totalt) icke-tomma rimliga konstruktorer (1p).
- *datamedlemmar (instansvariabler)*, minst en 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 och hålla sig inom det givna *temat* (**maträtter**) (2p).

Koncept / principer (11p)

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. Utöka dina lösningar från den grundläggande delen för att visa upp följande koncept. Vid behov får du lägga till ytterligare klasser.

- Polymorfi (2p).
- Statiska variabler och användning därav, minst en metod som behandlar den statiska variabeln krävs. (2p).
- Exception: Både skapande av *minst* ett eget Exception samt uppvisad användning därav. (1p).
- Multipla konstruktorer (flera *rimliga* konstruktorer i en och samma klass) (2p).
- Abstrakta metoder (minst en, samt överskuggning av densamma) (2p).
- Generiska klasser (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 (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 ska besvaras på egen textfil och skickas in som “Assignment2” i tentaklienten.

Fråga: Redogör kortfattat för vilka klasser som representerar A, B och C i klassdiagrammet, samt varför du valde att representera dem på det viset. Exempelvis: varför passade det bra att använda arv just här? Du behöver inte gå in på detaljer.