Distanstentamen TDDE10/725G90, 2020-06-10

Regler - generellt

- 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. Observera att frågor kan ställas i samband med legitimering, och legitimering kommer att ha företräde under tentans tidiga skede. **Jourtelefon:** 013-285886 (Magnus Nielsen)
- 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 ska sitta i en ostörd miljö utan andra personer i samma rum. Du ska hela tiden vara uppkopplad och synlig i Zoom videosamtal. Under eventuell rast sätter du bara upp en skylt som det står ”RAST” på framför kameran, och skickar in en rastinlämning via sendlab (oavsett uppgift) enligt länken nedan om du behöver sträcka på benen.
- Alla former av regelbrott medför att din tentamen underkänns direkt.

Regler - hjälpmedel

- Du får lov att använda dig av valfri Java-bok samt ett en egen-gjord ”fusklapp” (dubbelsidig A4).
- Du får lov att använda dig av Oracle’s api-dokumentation: https://docs.oracle.com/javase/8/docs/api/ samt TDDE10/725G90-kurshemsida: https://www.ida.liu.se/~TDDE10. Alla andra internetresurser är strikt förbjudna.
- All kommunikation är förbjuden, undantaget frågor till kurspersonal.
- All form av kopiering eller avskrift är förbjuden.
- Dina lösningar skickas in enligt instruktion i varje uppgift.
- Var beredd på att i efterhand kunna redogöra för dina svar.

Regler - betyg

Helhetsbetyget baseras på tentans båda delar, och du bör i båda uppgiftsdelarna uppfylla kraven för den betygsnivå du siktar på.

Tidsgränser

Ordinare tid: 08:00-12:00. Del 1: 08:00-10:00, Del 2: 10:00-12:00
Förlängd tid: 08:00-13:30. Del 1: 08:00-10:45, Del 2: 10:45-13:30

Del 1

Regler - inlämningar del 1

Inlämning sker via sendlab:
https://www.ida.liu.se/sendlab/student/send?code=tdde10_725g90_20200610_exam

- Slutinlämning senast kl 10:00 (10:45 förlängd tid)
- Inlämningen ska företrädesvis bestå av en zip- eller tar-fil med samtliga .java filer du har skrivit.
  Zip: Hur du gör en zip-fil beror på ditt operativsystem. Vanligen kan du markera samtliga filer och högerklicka (eller motsvarande) och ”lägg till i arkiv”.
  Tar: I terminal, kör exempelvis: tar cvf uppg1.tar *.java för att skapa en fil ”uppg1.tar” som innehåller samtliga java-filer i den mapp du för närvarande står.
  Oavsett vilket du väljer är det bra om du öppnar filen innan du skickar in för att bekräfta att det fungerade och att alla filer (alla .java-filer, skicka inte .class) finns med.
  Individuella filer: Det går även att skicka .java-filerna individuellt genom att lägga till dem en och en, och trycka på ”More files” i sendlabfönstret.
- Unvik väldigt gärna åääö helt i dina källkodsfiler, även i kommentarer. Det orsakar en hel del extra arbete att fixa till teckenkodning, undviker man åääö uppstår inte dessa problem. Ett tips är att skriva såväl källkod som kommentarer på engelska om du känner dig bekväm med det.

Uppgift Del 1 kl 08:00 - 10:00 (08:00-10:45)

Du ska skapa ett program uppbyggt av *minst* två klasser. Båda klasserna ska vara starkt förknippade med temat **Högtider**. Ditt program ska demonstrera dina kunskaper om *konstruktorer*, *datamedlemmar/instansvariabler*, *instansmetoder* och *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. Känner du att du inte har några kunskaper inom området går det utmärkt att hitta på.

- Du skall ha minst en klass som ärver från en annan (en *superklass* och en *subklass*).
- De (minst) två klasserna ska tillsammans visa på *minst* två exempel inom respektive kunskapsområde: *konstruktorer*, *datamedlemmar (instansvariabler)*, *instansmetoder (metoder tillhörande en instans av klassen)* och *inkapsling*. *Minst* en av instansmetoderna ska ta emot parametrar och utföra en för klassen relevant operation. Att endast returnera eller modifiera en instansvariabel är för trivialt (getters och setters räknas alltså inte).

Koncept

Under kursens gång har vi behandlat och arbetat med många olika objektorienterade koncept. Några av dessa är:

- Polymorfi
- Statiska variabler (och användning därav) **och** exceptions. Både skapande av *minst* ett eget Exception samt uppvisad användning därav. Dessa två koncept räknas som ett.
- Multipla konstruktorer (flera konstruktorer i en och samma klass)
- Abstrakta metoder (minst en, samt överskuggning av densamma)

För de olika betygen följer vissa krav på koncept som ska finnas med i din kod.

Regler - bedömning del 1

Obs! Betygsgränserna *kan* komma att justeras i efterhand (nedåt). För de olika betygen krävs:

- att du noga följt alla instruktioner och krav ställda i uppgiften
- att din kod följer god programmeringsstil (korrekt indentering, deklarerad synlighet, etc. Tänk på vad du lärt dig i labbserien.)
- att klasser har ett tydligt ansvar och metoder har en väl definierad uppgift
- att din kod har bra inkapsling och lämpliga avvägningar mellan lokala variabler och instansvariabler

TDDE10:

- **Betyg 3:** Exempel på *minst* ett valfritt koncept ovan ska finnas med i din inlämning.
- **Betyg 4:** Exempel på *minst* tre av ovanstående koncept ovan ska finnas med i din inlämning.
- **Betyg 5:** Exempel på samtliga av de fyra koncepten ovan ska finnas med, samt minst en egen generisk klass. Det kan vara enklare att göra en separat generisk klass.

725G90:

- **Betyg G:** Exempel på *minst* ett valfritt koncept ovan ska finnas med i din inlämning.
- **Betyg VG alternativ 1:** *Minst* två exempel vardera på *minst* två av ovanstående koncept ska finnas med i din inlämning, samt minst en egen generisk klass. Det kan vara enklare att göra en separat generisk klass.
- **Betyg VG alternativ 2:** Exempel på alla fyra koncepten ovan.

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 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` och ska implementera division men inte kommer på vad division innebär 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 tydligt som i divisionsexemplet.

Kom ihåg att hela tiden hålla dig till ditt tema **Högtider**.

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 de områden vi angivit.