Instruktioner för distanstenta

Anvisningar:

Alla lösningar bör skrivas efter varandra i ***en*** textfil (.txt).

Filen måste börja med studentens namn och en kort fritext där studenten intygar att alla lösningar i filen är skrivna på egenhand utan hjälp från andra och utan kopering från internet.

Varje svar måste börja med ***en tydlig rubrik*** i textfilen, t.ex.:

```
UPPGIFT 3
=========

Del a)

...

Del b)

...


UPPGIFT 4
=========

...
```

Man får ta kod från föreläsningspresentationerna, sina egna labblösningar och man får söka information om Javas API med Google.

Man får inte kommunicera med andra studenter eller på annat sätt be om lösningar. Man får inte söka lösningar på internet.

Lösningar bör skickas in som en textfil (dvs en fil som slutar med .txt) via Canvas Assignments innan tentan tar slut, dvs innan klockan 12:30.

Lycka till!

Kontakt:

Vid frågor går det att få tag på examinatorn Magnus Myréen med att ringa 0729744943 under tentans gång.

Data och Informationsteknik / Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
Ansvarig lärare: Magnus Myréen

DAT050 Objektorienterad programmering

Tentamen, måndag, 2021-01-04, 08:30-12:30

Vitsordsgränser: 3=24p, 4=36p, 5=48p, max 60p.

Kom ihåg att inte fastna på en uppgift. Bestäm i förväg din egen tidsgräns per uppgift. ***Lycka till!***

Uppgift 1: [15 poäng totalt] *att skriva klass, toString*

Denna uppgift handlar om att skriva en klass `SchoolClass` som representerar en skolklass. Instanser av `SchoolClass` ska innehålla följande information. [4 poäng]

- Klassens namn, t.ex. ”2D” eller ”5A”.
- Namnet på klassens lärare.
- Namnet på alla klassens elever.

`SchoolClass` ska ha följande konstruktor:

- En konstrukor som skapar en ny `SchoolClass` instans givet klassens namn (som `String`) och lärarens namn (som `String`). [3 poäng]

`SchoolClass` ska ha följande metoder:

- En instansmetod `addPupil` för att lägga till en ny elev. Metoden bör ta namnet (`String`) på eleven som input, den ska inte skriva ut något och den får inte returna något. Metoden bör alltså endast uppdatera instansen tillstånd. [3 poäng]

  Obs. Du får anta att det inte kommer att finnas elever med samma namn.

- En `toString` metod som *returnerar* en läsbar `String`-representation av klassens innehåll, som nedan. Denna `toString` metod får inte skriva ut något. [5 poäng]

Klassen ska vara skriven så att följande kod som använder `SchoolClass`:

```
/** Returnerar alltid en ny int array med slumpat innehåll.
SchoolClass c = new SchoolClass("2A", "Pelle Pedagog");
c.addPupil("Karl K");
c.addPupil("Anna D");
c.addPupil("Sofia S");
System.out.println(c.toString());
```

skriver ut:

```
[ SchoolClass 2A: Pelle Pedagog teaches Anna D, Karl K, Sofia S ]
```

För fulla poäng måste namnen på eleverna vara sorterad i alfabetisk ordning som ovan. Du får använda dig av vad som helst från Javas API i din lösning.

Din kod får inte vara specialiserad så att den endast fungera för exemplet ovan. Din kod får också inte ha en gräns på hur många studenter som kan finnas i en klass.

Tips. Undvik användning av arrays i din lösning.

Obs. Din kod får anta att användaren aldrig anropar konstruktorn eller metoderna med null.

Uppgift 2: [10 poäng totalt] *private/public, immutable*

Ge åtminstone en bra orsak varför intansvariabler bör vara deklarerade som `private` i Java. Ditt svar måste bestå av åtminstone ett kodexempel plus förklaring, dvs kod [5 poäng] och text [5 poäng].

Tips. Det är kanske enklast att svara på denna uppgift med att förklara hur en användare kan söndra klasser vars instansvariabler är deklarerade till något annat än `private`. Tänk till exempel på hur en användare kan söndra `RatNum` klassen från Lab 2 ifall instansvariablerna är deklarerade som `public`.

Obs. En välbeskriven giltig orsak är tillräckligt för fulla poäng. Man får beskriva flera orsaker, men notera att det blir poängavdrag ifall svaret inkluderar felaktiga orsaker.

Uppgift 3: [15 poäng totalt] *att skriva klass, interface, exceptions*

a) Definiera en klass `NoSuchCarException` [3 poäng]. Klassen behöver inte ha tillstånd, men den bör kunna användas som ett exception. Förklara kort varför klassen kan användas som ett exception [2 poäng].

b) Din uppgift är att implementera en metod med följande sigatur. [10 poäng]

```
public Car whichCarToBuy(List<Car> cars, int maxMoneyToSpend) { ... }
```

Metoden tar två parametrar `cars` och `maxMoneyToSpend`. Den första, dvs `cars`, är en lista av bilar av typen Car, se definitionen nedan. Den andra är hur mycket pengar som man har att spendera på ett bilköp.

```
public interface Car {
   public String getName();
   public int getPrice();
}
```

Metoden `whichCarToBuy` bör returnera den dyraste bilen från listan `cars` som kan köpas med pengarna `maxMoneyToSpend`. Om det inte finns någon sådan bil, då bör `whichCarToBuy` kasta ett `NoSuchCarException`.

Obs. Var försiktig med `null` värden i din kod. Dessa kan ju komma i input.

Uppgift 4: [10 poäng totalt] *testning, int typen, exceptions*

Denna uppgift handlar om att testa att funktionen `myFunction` håller sig till specifikationen som är skriven nedan i en Javadoc kommentar.

```
public interface MyInterface {

  /** Denna metod får endast kasta ett exception ifall n är 0 */
  public int myFunction(int n);

}
```

Skriv en metod `isOK` med signaturen som nedan. Metoden `isOK` bör returnera `true` ifall `myFunction` inuti `MyInterface` `mi` respekterar sin specifikation *för alla input*.

```
public boolean isOK(MyInterface mi) { ... }
```

Metoden `isOK` får aldrig släppa ut en exception, dvs den måste fånga och hantera alla exceptions som koden som testas producerar (oberoende om den testade koden har fel).

Obs. Specifikationen säger inte att `myFunction` kommer att kasta en exception då `n` är 0.

Uppgift 5: [10 poäng totalt] *arv, händelsehantering, Javas API*

Implementera en knapp-klass `RandomColorButton` som ärver `JButton` och som på alla sätt beter sig som en `JButton`, förutom vid knapptryck. Vid varje knapptryck bör en `RandomColorButton` byta färg till en slumpmässig färg.

Tips. Använd `JButton`, `ActionListener`, `Random` och `Color` från Javas API. Man kan byta färg på en knapp med att anropa `setBackground`-metoden i `JButton` klassen.