Skriftelig prøve den 12. januar 2001 Kursusnummer 49104

Kursusnavn: Programmering.

Tilladte hjælpemidler: Alle *skriftlige* hjælpemidler

Opgavesættet består af fire opgaver, der har følgende vægtning:

Opgave 1	20%
Opgave 2	20%
Opgave 3	30%
Opgave 4	30%

Sættet bedømmes efter 13-skalaen og tæller $1/3$ af den samlede karaktér for kursus 49104 og 49105.

Opgave 1

Spørgsmål 1.1

Vis, hvad følgende Java-program skriver på skærmen når det køres.

```
public class FunnyStrings
{
   private static
   String funny (String str, int n)
      {
         String res = "";
         int len = str.length();

         for (int i=0; i<n; i=i+1)
         {
            res = res + str.charAt(i%len);
         }
         return res;
      }

  public static
   void main (String[] args)
      {
         System.out.println(funny("abc",3));
         System.out.println(funny("abc",5));
         System.out.println(funny("0123456789",6));
      } // main
} // FunnyStrings
```

Spørgsmål 1.2

Når spørgsmålene 1.2.A, 1.2.B og 1.2.C er besvaret skal følgende programstump

```
public class NordicFun
{
   private static boolean isSenName (String name)     {  ...  }
   private static String  change (String name)        {  ...  }
   private static String  toSwedish (String lastname) {  ...  }

   public static
   void main (String[] args)
      {
         String[] name = {"Petersen", "Jensen", "Dahl",
                          "Olsen", "Berggreen"};
         for(int i=0; i<name.length; i=i+1)
            System.out.println(toSwedish(name[i]));
      }
} // NordicFun
```

udskrive

```
Peterson
Jenson
Dahl
Olson
Berggreen
```

Alle efternavne der ender på "sen" udskrives med "son" som endelse, resten udskrives uden ændringer.

Vink: metoder fra `String` klassen kan med fordel anvendes i denne opgave.

Spørgsmål 1.2.A

Færdiggør metoden **`boolean`** `isSenName(String name)` så den returnérer **`true`**, hvis `name` ender på "sen" og **`false`** ellers.

Spørgsmål 1.2.B

Færdiggør metoden `String change(String name)` så den returner en streng, hvor det næstsidste bogstav i strengen `name` er blevet ændret til et `'o'`, idet man kan antage at længden af `name` er større end 2.

Spørgsmål 1.2.C

Lav `String toSwedish(String lastname)` metoden færdig ved hjælp af de to hjælpemetoder `isSenName` og `change`. `toSwedish` skal returnére en "svensk" udgave af argumentet `lastname`, det vil sige at hvis `lastname` ender på "sen" skal der returnéres en streng, hvor endelsen er ændret til "son" og ellers skal der bare returnéres det oprindelige navn.

Opgave 2

Følgende programstump er givet:

```
public class Factorise
{
   private static
   int minDivsor (int n) {  ...  }

   private static
   String factorise (int n) {  ...  }

   public static
   void main (String[] args)
      {
         System.out.println("42 = " + factorise(42));
         System.out.println("100 = " + factorise(100));
         System.out.println("385 = " + factorise(385));
         System.out.println("1386 = " + factorise(1386));
         System.out.println("1729 = " + factorise(1729));
      }
}
```

Når programmet er gjort færdigt skal det udskrive følgende:

```
42 = 2*3*7
100 = 2*2*5*5
385 = 5*7*11
1386 = 2*3*3*7*11
1729 = 7*13*19
```

Spørgsmål 2.1

Lav `minDivisor` så den returnérer det midste tal større end 1 som går op i argumentet `n` ($n \geq 2$). For eksempel har vi at `minDivisor(4)` giver 2 og `minDivisor(5)` giver 5 (fordi 5 er det største tal større end 1 som går op i 5).

Spørgsmål 2.2

Brug `minDivisor` metoden til at færdiggøre `factorise` metoden, så den returnérer en streng med faktoriseringen af argumentet `n` ($n > 0$). Som set ovenfor giver `factorise(42)` strengen `"2*3*7"`.

Opgave 3

I denne opgave skal der laves et program, `GUI_Exchange`, med en grafisk brugergrænseflade. Et screenshot fra programmet kunne se således ud:

Programmet skal virke på den måde, at brugeren kan taste tal ind i de to tekstfelter på øverste linje og få lavet en omregning fra den ene valuta til den anden ved tryk på en af omregningsknapperne. For eksempel vil et tryk på `"Euro -> $"` knappen omregne tallet i venstre tekstfelt til et antal dollars i det højre. Trykker man på `"Clear"`-knappen skal begge tekstfelter gøres tomme.
NB: vi arbejder kun med *hele* tal i denne opgave.
Som hjælp til omregningen er der givet klassen `Exchange`, der har metoder til at foretage omregningen mellem de to valutaer:

```
public class Exchange
{
   public static
   int euro2dollars (int euro)
      {
         return (int) (euro * 0.9);
      }

   public static
   int dollars2euro (int dollars)
      {
         return (int) (dollars / 0.9);
      }
}
```

På næste side er der et programskelet for `GUI_Exchange` programmet, som skal gøres færdigt.

! Det er ikke nødvendigt at skrive hele programmet — nøjes med at skrive de metoder og klasser I laver.

Spørgsmål 3.1

Gør `GUI_Exchange` konstruktoren færdig — det vil sige: initialisering af instansdata, tilføjelse af de grafiske komponenter til klassens *content pane* (husk at der er tale om en `JFrame`, så man kan ikke tilføje komponenter direkte).

! **NB:** i dette spørgsmål kan man tage for givet at `MyListener` klassen er implementeret således, at den kan lytte til alle knapperne og sørger for passende opdatering(er).

Spørgsmål 3.2

Lav `MyListener` klassen færdig således, at den lytter til alle knapperne og sørger for passende opdatering(er) som følge af modtagne `ActionEvent`s fra knapperne.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import Exchange;

public class GUI_Exchange extends JFrame
{
   private JButton euro2dollars,dollars2euro,clear;
   private JTextField euro,dollars;

   public
   GUI_Exchange ()
      {
          ...
      }

   private class MyListener implements ActionListener
   {
      public
      void actionPerformed (ActionEvent event)
         {
             ...
         } // actionPerformed
   } // MyListener


   public static
   void main (String[] args)
      {
         GUI_Exchange f = new GUI_Exchange();
         f.pack();
         f.show();
      }
} // GUI_Exchange
```

Opgave 4

I denne opgave ser vi på en simpel model af kurser på DTU.

Et kursus beskrives med følgende interface

```
import java.util.Vector;

public interface Course
{
   String name();
   boolean isBlockedBy(Course course);
   Vector requirements();
}
```

hvor opførslen af metoderne er givet nedenfor.

- `String name()`
  Returnérer navnet på kurset.
- **`boolean`** `isBlockedBy(Course course)`
  `c.isBlockedBy(c2)` giver **`true`**, hvis `c2` er blandt de kurser som spærrer for kursus `c`. For eksempel spærrer 49104 for informatikkernes Java kursus (49137).
- `Vector requirements()`
  Returnérer en vektor indeholdende alle de kurser som dette kursus har som forudsætninger. For eksempel er 49104 forudsætningskursus 49161 Funktionsprogrammering. For kurser uden forudsætninger returnéres en tom vektor.

Spørgsmål 4.1

Lav en klasse `Basic_Course` som implementérer `Course` interfacet med følgende ekstra betingelser:

- konstruktoren skal have formen
  `Basic_Course(String name, Course[] blocking)` hvor `name` er navnet på kurset og arrayet `blocking` indeholder alle de kurser, som spærrer for dette kursus.
- instanser af `Basic_Course` har *ingen* forudsætninger.

Spørgsmål 4.2

Lav en metode

`boolean` `requirementsOK(Vector passed, Course c)`

som afgør om alle `c`'s forudsætningskurser er indeholdt i vektoren `passed`. Metoden skal returnére **`true`**, hvis dette er tilfældet og **`false`** ellers.