Institutionen för TENTAMEN
Data- och informationsteknik 2016-02-13

PROGRAMMERINGSTEKNIK
för F1 och TM1 (TIN212)

OBS! Det kan finnas kurser med samma eller liknande namn på olika utbildningslinjer. Denna tentamen gäller *endast* för den eller de utbildningslinjer som anges ovan. Kontrollera därför noga att denna tentamen gäller för den utbildningslinje du själv går på.

TID 08:30 - 13:30

Ansvarig: Jan Skansholm

Förfrågningar: Jonathan Orrö, tel. 0708 867104

Betygsgränser: Sammanlagt maximalt 60 poäng.
På tentamen ges graderade betyg:
CTH: 3:a 24 poäng, 4:a 36 poäng, 5:a 48 poäng

Hjälpmedel: Bravaco, Simonson, *Java Programming From the Ground Up*, eller
Horstmann, *Java for everyone*, eller
Morelli, Walde, *Java, Java, Java: Object-oriented problem solving*, eller
Liang, *Introduction to Java programming Comprehensive version*, eller
Eckel, *Thinking in Java,* eller
Skansholm, *Java direkt med Swing*, Studentlitteratur
(Understrykningar och mindre anteckningar i boken är tillåtna.)

Inga kalkylatorer är tillåtna.

Tänk på:

- att skriva tydligt och disponera papperet på ett lämpligt sätt.
- att börja varje ny uppgift på nytt blad. Skriv endast på en sida av papperet
- Skriv den (anonyma) kod du fått av tentamensvakten på *alla* blad.

De råd och anvisningar som givits under kursen skall följas vid programkonstruktionerna. Det innebär bl.a. att onödigt komplicerade, långa och/eller ostrukturerade lösningar i värsta fall ej bedöms.

Uppgift 1) a) Följande fil innehåller fyra fel som upptäcks vid kompileringen. Vilka är dessa? Ange radnummer. För poäng krävs att felen förklaras.

```
 1 interface A {
 2   abstract int m1(int i);
 3 }
 4
 5 interface B implements A {
 6   public int k = 0;
 7   public int m2(int i);
 8 }
 9
10 interface C {
11   int i=0;
12   int m3(int i);
13 }
14
15 public abstract class D extends C implements B {
16   public D(int i) {k = i;}
17   public int m4(int i);
18 }
```

(2 p)

b) Antag att vi har följande klasser:

```
public interface Vehicle { ... }
public class Car implements Vehicle { ... }
public class Suv extends Car { ... }
```

Ange för var och en av följande rader om den är korrekt eller felaktig.

```
Vehicle v = new Car();
Vehicle v = new Suv();
Vehicle v = new Vehicle();
Car c = new Suv();
Car c = new Vehicle();
Suv s = new Suv();
Suv s = new Car();
```

(2 p)

c) Avsikten med följande programrader är att ta bort alla element som innehåller värdet noll ur en lista `li` av typen `List<Integer>`. Men det fungerar inte alltid som man tänkt. Ge ett exempel på när detta kan ske och föreslå hur man kan skriva om programraderna så att det fungerar.

```
if (li != null)
  for (int i=0; i < li.size(); i++)
    if (li.get(i) != null && li.get(i) == 0)
      li.remove(i);
```

(2 p)

Uppgift 2) Konstruera en klass `Tärning` som skall beskriva en vanlig tärning som man använder t.ex. när man spelar Fia. Följande metoder skall finnas: `kasta`, som ger tärningen ett slumpmässigt värde mellan 1 och 6, `avläsVärde` som avläser tärningens värde och ger det som resultat samt `sättVärde` som har en parameter vilken anger vilket sida på tärningen man vill skall vara upp. Vidare skall det finnas två konstruktorer, en som initierar tärningen till ett slumpmässigt värde och en som gör det möjligt att ge ett visst första värde till tärningen. Parametern till metoden `sättVärde` och till den andra konstruktorn måste ha ett värde mellan 1 och 6. Skulle så inte vara fallet skall klassen ge en exception av typen `IllegalArgumentException`.

(4 p)

Uppgift 3) Ett sätt att kryptera hemliga meddelanden är att använda Vigenèrekryptot. Meddelandet som skall krypteras skrivs då med stora bokstäver och man har tagit bort alla tecken som inte är bokstäver Man använder bara bokstäverna A-Z. Förutom meddelandet som skall krypteras har man också en hemlig nyckel. Denna är också en text som bara innehåller stora bokstäver. Bokstäverna A-Z numreras från 0 till 25. Krypteringen görs sedan genom att man skriver nyckeln under meddelandet och adderar bokstävernas nummer parvis. Additionen sker modulus 26. Man får då numret på den bokstav som skall ingå i det krypterade meddelandet. Om nyckeln är kortare än meddelandet upprepar man nyckeln. Det kan t.ex. se ut på följande sätt.

```
KOMIDAG   (meddelande)
CTHCTHC   (nyckel)
MHTKWHI   (krypterat meddelande)
```

För att dekryptera meddelandet måste man förstås känna till den hemliga nyckeln. Dekrypteringen går till på samma sätt som krypteringen, men istället för att addera bokstävernas nummer så subtraherar man numren för bokstäverna i nyckeln från numren för motsvarande bokstäver i det krypterade meddelandet.

Din uppgift är att skriva en klassmetod som utför kryptering eller dekryptering och som ger det krypterade resp. det dekrypterade meddelandet som resultat. Metoden ska ha tre parametrar: meddelandet som ska krypteras eller dekrypteras, nyckeln samt en **`boolean`** som anger om det gäller kryptering eller dekryptering. De två första parametrarna samt resultatet ska vara av typen `String`. Du får förutsätta att meddelandet och nyckeln bara innehåller stora bokstäver A-Z. *Tips*: Man kan tilldela ett värde av typen **`char`** direkt till en **`int`**, men för att tilldela åt andra hållet krävs en explicit typomvandling.

(12 p)

Uppgift 4) Man har i en textfil samlat uppgifter om ett antal personer. För varje person finns två rader i filen. På första raden står personens namn och adress och på andra raden finns personens ålder, längd och vikt. Längden anges i cm och vikten i kg. Man vill göra en medicinsk studie av överviktiga personer och söker därför personer vilkas s.k. *body mass index* (BMI) överstiger 30. BMI beräknas enligt formeln m/h^2 där m är vikten i kg och h längden i m. Skriv ett program som läser filen med personuppgifter. Programmet skall skapa en ny textfil som bara innehåller uppgifterna för dem vilkas BMI överstiger 30.

När man startar programmet skall det från kommandofönstret (via `System.in`) läsa in namnet på den befintliga filen och namnet på den nya filen. Om någon av filerna inte går att öppna skall programmet ge en lämplig felutskrift (med ev. filnamn) och avslutas. *Tips*. Om man misslyckas med att öppna en fil får man en `IOException`.

(12 p)

Uppgift 5) Standardklassen `String` har som du vet egenskapen att det inte går att ändra ett visst `String`-objekt. Varje gång man vill lägga en text i ett `String`-objekt skapas det ju et *nytt* objekt. Detta kan förstås vara lite ineffektivt om man arbetar med texter som skall ändras ofta. I Java finns emellertid även klasserna `StringBuilder` och `StringBuffer`. Objekt av dessa klasser innehåller också texter men texterna är ändringsbara. I denna uppgift skall du skriva en egen (kraftigt förkortad) version av klassen `StringBuilder`. Din klass skall heta `MyStringBuilder`.

I din klass *skall* texten lagras i en intern array med komponenter av typen **`char`**. (Du får inte lagra texten som en `String`.) Antalet komponenter i arrayen kallas *kapaciteten*. I arrayen kan man lagra texter vilkas längd inte överskrider kapaciteten. Om man försöker lagra en längre text skall kapaciteten automatiskt ökas (vi återkommer till detta).

I klassen `MyStringBuilder` skall det finnas två konstruktorer: Den första skall vara parameterlös och när den används skall den initiera det nya objektet så att kapaciteten blir lika med 16. Den andra konstruktorn skall ha en parameter av typen `String`. När denna konstruktor används skall kapaciteten bli lika med parameterns längd plus 16. Dessutom skall den text som finns i parametern placeras i början av den interna arrayen.

Förutom konstruktorerna skall klassen `MyStringBuilder` ha följande metoder:

- `length()`, ger längden av den text som finns i objektet,
- `capacity()`, ger objektets kapacitet,
- `ensureCapacity(k)`, utökar (om så behövs) objektets kapacitet så att den blir större än eller lika med `k`. Kapaciteten skall (ev. upprepade gånger) fördubblas tills villkoret är uppfyllt.
- `insert(p,s)`, skjuter in texten `s` (en `String`) med början i position `p` (en **`int`**), utökar kapaciteten om så behövs. Skall kontrollera att `p` är större än eller lika med noll och mindre än eller lika med `length()`. Om så inte är fallet skall en exception av standardklassen `IndexOutOfBounds` genereras. Om `p` är mindre än `length()` måste en del av den tidigare texten förskjutas för att bereda plats för `s`.
- `substring(p,m)`, ger som resultat en `String` som består av tecknen i positionerna `p` till `m-1`. Om `p` eller `m` är negativa eller större än `length()` eller om `p > m` så skall en exception av standardklassen `IndexOutOfBounds` genereras.
- `append(s)`, lägger till texten `s` (en `String`) sist, utökar kapaciteten om så behövs.
- `toString`, returnerar texten i objektet som en `String`.

Tips: Några metoder blir väldigt enkla om du är smart och utnyttjar annat du gjort.

(14 p)

Uppgift 6) Uppgiften är att skriva ett program som visar ett virtuellt tangentbord på skärmen, dvs ett program som visar en massa knappar vilka fungerar som ett vanligt tangentbord. När du kör ditt program skall det se ut som i följande figur (den nedersta långa knappen är mellanslag).

Ovanför knapparna skall den inmatade texten visas. När man trycker på en av knapparna läggs dess tecken till i slutet av texten. Det går inte att sudda det som redan står i texten och det skall inte gå att skriva eller förändra texten på något annat sätt än genom att trycka på knapparna. Det finns många knappar för att uppgiften inte skall gå att lösa med klipp-och-klistra-programmering. Försök istället hitta en mer strukturerad lösning, som inte är beroende av antalet knappar. Lösningar som innehåller “och så gör vi alla de andra knapparna på samma sätt”, “och så vidare”, “...” eller liknande kommer inte att godkännas, utan du måste skriva ut *hela* din lösning. (12 p)

Lösningar till tentamen 2016-02-13

```
// Uppgift 1 a
rad 5:  skall stå extends, inte implements.
        Ett gränssnitt kan utöka ett annat, dvs. lägga till fler krav,
        inte implementera det, dvs. innehålla hela metoder.
rad 15: Det får inte stå extends C eftersom C är ett gränssnitt, inte
  en klass
rad 16: Tilldelningen till k är felaktigeftersom k har egenskapen final.
rad 17: Metoden m4 måste antingen deklareras som abstrakt eller innehålla
  en metodkropp.

// Uppgift 1 b
    Vehicle v = new Car(); // ok
    Vehicle v = new Suv(); // ok
    Vehicle v = new Vehicle(); // Vehicle is abstract; cannot be
                               //instantiated
    Car c = new Suv(); // ok
    Car c = new Vehicle(); // Vehicle is abstract; cannot be
                           //instantiated
    Suv s = new Suv(); // ok
    Suv s = new Car(); // incompatible types

// Uppgift 1 c
Om två intilliggande element båda innehåller värdet 0 kommer det andra
av dessa element inte att tas bort. Ett sätt att lösa det är att gå
igenom listan baklänges.


// Uppgift 2

     public class Tarning {
       private int värde;

       public Tarning() {
         kasta();
       }

       public Tarning(int v) {
         sättVärde(v);
       }

       public int  avläsVärde() {
         return värde;
       }

       public void sättVärde(int v) {
         if (v >= 1 && v <= 6)
           värde = v;
         else
           throw new IllegalArgumentException();
       }
```

```
  public void kasta() {
    värde = (int) (Math.random() * 6) + 1;
  }
}
```

```
// Uppgift 3

 public static String crypt(String mes, String key, boolean decrypt) {
   int j = 0;
   String r = "";
   for (int i=0; i<mes.length(); i++) {
     int m = mes.charAt(i) - 'A';
     int k = key.charAt(j) - 'A';
     j = (j+1) % key.length();
     int n;
     if (decrypt)
       n = (26 + m - k) % 26;
     else
       n = (m + k) % 26;
     r += (char)(n + 'A');
   }
   return r;
 }

// Uppgift 4

import java.io.*;
import java.util.*;

public class BMI {
  public static void main(String[] arg) {
    System.out.println("Namnet på den befintliga filen? ");
    Scanner sc = new Scanner(System.in);
    String namn1 = sc.nextLine();
    System.out.println("Namnet på den nya filen? ");
    String namn2 = sc.nextLine();
    try {
      sc = new Scanner(new File(namn1));
    }
    catch (IOException e) {
      System.out.println("Kan inte öppna filen " + namn1);
      System.exit(1);
    }
    PrintWriter utström = null;
    try {
      utström = new PrintWriter(new BufferedWriter(new FileWriter(namn2)));
    }
    catch (IOException e) {
      System.out.println("Kan inte skapa filen " + namn2);
      System.exit(2);
```

```
        }
        while(sc.hasNext()) {
          String rad1 = sc.nextLine();
          int ålder = sc.nextInt();
          int längd = sc.nextInt();
          double vikt = sc.nextDouble();
          sc.nextLine();  // avsluta rad 2
          double bmi = vikt / (0.01*längd*0.01*längd);
          if (bmi > 30) {
            utström.println(rad1);
            utström.println(ålder + " " + längd + " " + vikt);
          }
        }
        utström.close();
      }
    }

// Uppgift 5
     public class MyStringBuilder {

     private char[] b;       // arrayen med texten
     private int n = 0;      // längden

     public MyStringBuilder() {
       b = new char[16];
     }

     public MyStringBuilder(String s) {
       n = s.length();
       b = new char[n + 16];
       for (int i=0; i<n; i++)
         b[i] = s.charAt(i);
     }

     public int length() {
       return n;
     }

     public int capacity() {
       return b.length;
     }

     public void ensureCapacity(int k) {
       int cap = b.length;
       // beräkna ny kapacitet
       while (cap < k)
         cap *= 2;
       // skapa nytt fält om kapaciteten behövde utökas
       if (b.length < k) {
         char[] b2 = new char[cap];       // skapa en ny array
         for (int i=0; i<b.length; i++)   // kopiera tidigare text
           b2[i] = b[i];
         b = b2;
```

```
    }
  }

  public void insert(int p, String s) {
    int d = s.length();
    if (p < 0 || p > n)
      throw new IndexOutOfBoundsException();
    ensureCapacity(n+d);
    for (int i=n-1; i>=p; i--)    // förskjut tidigare text d steg
                                  // åt höger
      b[i+d] = b[i];
    for (int i=0; i<d; i++)       // lägg in ny text
      b[p+i] = s.charAt(i);
    n += d;
  }

  public void append(String s) {
    insert(n, s);
  }

  public String substring(int p, int m) {
    if (p < 0 || p > n || m < 0 || m > n || p > m)
      throw new IndexOutOfBoundsException();
    String s = "";
    for (int i=p; i<m; i++)
      s += b[i];
    return s;
  }

  @Override
  public String toString() {
    return substring(0, n);
  }
}

// Alternativ version som använder fler standardmetoder
class MyStringBuilder2 {

  private char[] b;       // arrayen med texten
  private int n = 0;      // längden

  public MyStringBuilder2() {
    b = new char[16];
  }

  public MyStringBuilder2(String s) {
    n = s.length();
    b = new char[n + 16];
    s.getChars(0, n, b, 0);   // kopiera från s till b
  }

  public int length() {
    return n;
```

```
  }

  public int capacity() {
    return b.length;
  }

  public void ensureCapacity(int k) {
    int cap = b.length;
    // beräkna ny kapacitet
    while (cap < k)
      cap *= 2;
    // skapa nytt fält om kapaciteten behövde utökas
    if (b.length < k) {
      char[] b2 = new char[cap];       // skapa ny array
      System.arraycopy(b, 0, b2, 0, b.length);  // kopiera från
                                                // b till b2
      b = b2;
    }
  }

  public void insert(int p, String s) {
    int d = s.length();
    if (p < 0 || p > n)
      throw new IndexOutOfBoundsException();
    ensureCapacity(n+d);
    System.arraycopy(b, p, b, p+d, n-p); // förskjut tidigare text
                                         // d steg åt höger
    s.getChars(0, d, b, p);              // lägg in ny text
    n += d;
  }

  public void append(String s) {
    insert(n, s);
  }

  public String substring(int p, int m) {
    if (p < 0 || p > n || m < 0 || m > n || p > m)
      throw new IndexOutOfBoundsException();
    return new String(b, p, m-p);
  }

  @Override
  public String toString() {
    return new String(b, 0, n);
  }
}


// Uppgift 6

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

```
public class Tangentbord extends JFrame implements ActionListener {
  private final String[] b = {"1", "2", "3", "4", "5", "6", "7", "8", "9",
                              "0", "+", "Q", "W", "E", "R", "T", "Y", "U",
                              "I", "O", "P", "Å", "A", "S", "D", "F", "G",
                              "H", "J", "K", "L", "Ö", "Ä", "Z", "X", "C",
                              "V", "B", "N", "M", ",", ".", "-"};
  private JLabel output = new JLabel(" ");

  public Tangentbord() {
    setLayout(new BorderLayout());
    add(output, BorderLayout.NORTH);
    JPanel buttons = new JPanel(new GridLayout(4, 11));
    for (int i = 0; i < b.length; i++) {
      JButton tmp = new JButton(b[i]);
      tmp.addActionListener(this);
      buttons.add(tmp);
    }
    add(buttons, BorderLayout.CENTER);
    JButton tmp = new JButton(" ");
    tmp.addActionListener(this);
    add(tmp, BorderLayout.SOUTH);
    pack();
    setDefaultCloseOperation(EXIT_ON_CLOSE);
    setVisible(true);
  }

  public void actionPerformed(ActionEvent e) {
    JButton jb = (JButton) e.getSource();
    output.setText(output.getText() + jb.getText());
  }

  public static void main(String[] arg) {
    new Tangentbord();
  }
}
```