PROG2 Tenta 2014-05-02

Gäller SP:PROG2, DSK2:PROG2, FK:PROG2, FK:OOP, DSV1:P2 och ITK:P2

Tentan består av tre uppgifter. Max poäng är 38.
För betyget E (godkänd) krävs minst 23 poäng och **minst en poäng på varje uppgift.**
Betygen A-D ges enligt betygskriteria i Daisy och på nästa sida.

Hjälpmedel:

Tillåtna hjälpmedel är medhavda böcker om Java.

Anvisningar:

Skriv endast på ena sidan av bladen.
Påbörja varje ny uppgift på nytt blad.
Skriv tydligt - oläsbara svar beaktas inte.

Även icke fullständiga lösningar beaktas.
Kommentera gärna era lösningar.

Lycka till!

Tentan betygsätts i A/B/C/D/E/Fx/F-skalan enligt följande kriteria:

A - Samtliga lösningar är felfria (förutom uppenbara små misstag), fullständiga, genomförda med användning av de på kursen presenterade Java-teknikerna och lämpliga klasser/metoder ur Javas standardbibliotek och med kod som är klar, effektiv och inte onödigt omständlig. Lösningarna tar hänsyn till alla i uppgiftstexten angivna situationer och innehåller alla i uppgiftstexten begärda felkontroller. Däremot behöver de inte innehålla andra felkontroller eller ta hänsyn till andra situationer än de som angetts i uppgiftstexten.

B - Som för betyget A utom att någon eller ett par lösningar kan innehålla något grövre misstag eller underlåta att ta hänsyn till någon enstaka angiven situation eller någon enstaka begärd felkontroll. Lösningarna kan vara något mer omständliga än nödvändigt.

C - Lösningarna är i princip korrekta men kan vara behäftade med fel och/eller underlåta att ta hänsyn till någon enstaka angiven situation eller begärd felkontroll. Lösningarna kan vara mer omständliga än nödvändigt (t.ex. skrivna med egen kod där standardklasser/-metoder kunde ha använts).

D - Som för betyget C utom att någon eller ett par lösningar kan innehålla grövre fel av principiell karaktär. Lösningarna kan även innehålla grövre syntaktiska misstag (t.ex. sammanblandning med andra språk med liknande syntax).

E - Som för betyget D utom att de flesta eller alla lösningar är behäftade med grövre fel av principiell karaktär. Icke desto mindre måste lösningarna visa grundläggande förståelse för problemet och åtminstone en ansats till korrekt lösning.

Fx - En lösning är helt felaktig eller saknas medan de övriga lösningar är i princip korrekta, men kan vara behäftade med fel och/eller underlåta att ta hänsyn till någon enstaka angiven situation eller begärd felkontroll.

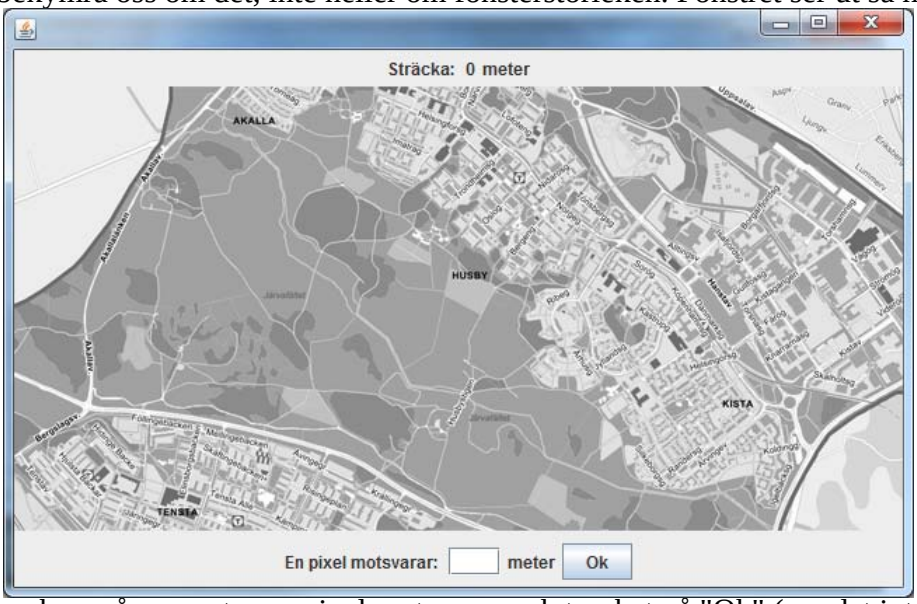
F - Flera lösningar är helt felaktiga eller saknas.

Betyget Fx innebär att studenten kan komplettera examinationen med en extra inlämningsuppgift för att få godkänt på aktuell tentamen (E men ej högre betyg). Kompletteringsuppgiften måste lämnas in enligt angiven deadline och kan endast användas för att få betyget E på den aktuella tentan.

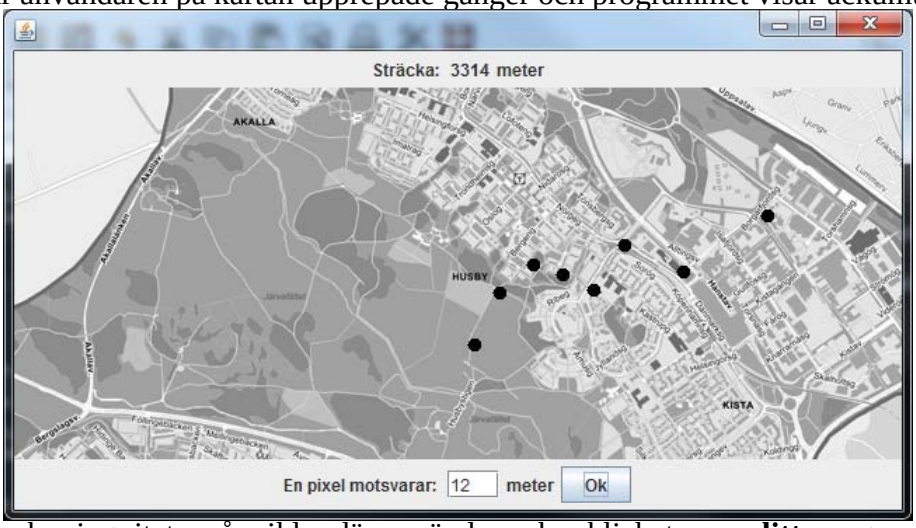
Uppgift 1 (13 poäng)

Den här uppgiften går ut på att komplettera ett program som visar en bild föreställande en karta och som kan användas för att på kartan mäta en sträcka som följer en (ev. krokig) väg. Detta görs genom att användaren klickar med musen på kartan och programmet räknar ut avståndet mellan den punkt där användaren klickat och föregående punkt, och lägger ihop dessa avstånd (se nedan om hur man räknar ut avståndet). På det sättet kan användaren följa en krokig stig med en serie raka sträckor.
För att kunna meddela avståndet i meter måste programmet veta hur många meter varje pixel motsvarar, detta ska användaren mata in.

När programmet startat visar det en karta (från en bildfil). Filens namn anges på kommandoraden, vi behöver inte bekymra oss om det, inte heller om fönsterstorleken. Fönstret ser ut så här:



Användaren anger hur många meter en pixel motsvarar och trycket på "Ok" (om det inte finns någon angivelse i textfältet eller om angivelsen är icke-numerisk bör ett felmeddelande visas i en dialogruta). Sedan klickar användaren på kartan upprepade gånger och programmet visar ackumulerade sträckan:



På bilden ovan har jag ritat små criklar där användaren har klickat, men **ditt program behöver inte göra det**, det ska bara räkna ut sträckan och visa den i övre panelen.

Om man har koordinater för två punkter, säg `x1`,`y1` och `x2`,`y2` så kan man räkna ut avståndet mellan dem genom Pythagoras sats: `avstånd` $= \sqrt{(x2-x1)^2 + (y2-y1)^2}$, översatt till Java:

```
double avstånd = Math.sqrt(Math.pow(x2-x1, 2)+Math.pow(y2-y1, 2));
```

Se nästa sida för kod som du ska komplettera.

Nedan finns koden till klassen `Kartan` som du ska komplettera (radnumreringen ingår inte i koden, den är till för att du enkelt skall kunna förklara var du gör tillägg/ändringar):

```
1     import javax.swing.*;
2     import java.awt.*;
3     import java.awt.event.*;
4
5     class KartPanel extends JPanel{
6         private ImageIcon bilden;
7
8         public KartPanel(String filnamn){
9           setLayout(null);
10          bilden = new ImageIcon(filnamn);
11        }
12
13        protected void paintComponent(Graphics g){
14          super.paintComponent(g);
15          g.drawImage(bilden.getImage(),0,0,this);
16        }
17    }
18
19    class Kartan extends JFrame{
20        Kartan(String filnamn){
21
22          JPanel norra = new JPanel();
23          add(norra, BorderLayout.NORTH);
24          norra.add(new JLabel("Sträcka: "));
25          norra.add(new JLabel("0"));
26          norra.add(new JLabel("meter"));
27
28          add(new KartPanel(filnamn));
29
30          JPanel södra = new JPanel();
31          add(södra, BorderLayout.SOUTH);
32          södra.add(new JLabel("En pixel motsvarar: "));
33          södra.add(new JTextField(3));
34          södra.add(new JLabel("meter"));
35          södra.add(new JButton("Ok"));
36
37          setDefaultCloseOperation(EXIT_ON_CLOSE);
38          setSize(650,350);
39          setVisible(true);
40        }
41        public static void main(String[] args){
42          new Kartan(args[0]);
43        }
44    }
```

Uppgift 2 (12 poäng)

I båtklubbar har man ett vaktschema - varje dag på året måste någon av medlemmarna gå nattvakt på klubbens brygga. Man vill ha ett Java-program där man kan ange för varje dag vilken medlem som ska ha vakten.
Dagar representeras med ett veckonummer och veckodagen inom veckan, fast veckodagen representeras med ett nummer 0 - 6, där 0 är måndag, 1 är tisdag o.s.v.:

```
class Dag{
    private int veckoNr, veckoDag;

    private static final String[] VDAGAR =
      {"Måndag","Tisdag","Onsdag","Torsdag","Fredag","Lördag","Söndag"};

    public Dag(int veckoNr, int veckoDag){
      if (veckoNr < 1 || veckoNr > 53 || veckoDag < 0 || veckoDag > 6)
          throw new IllegalArgumentException();
      this.veckoNr = veckoNr;
      this.veckoDag = veckoDag;
    }

    public String toString() {
      return "vecka " + veckoNr + " " + VDAGAR[veckoDag];
    }

} // Dag
```

Klassen `Dag` behöver kompletteras i denna uppgift.

Klubbmedlemmar representeras helt enkelt med deras namn (`String`).

a) Man vill kunna använda objekt av klassen `Dag` som nycklar i en `Map`. Man har inte bestämt sig om man ska använda en `HashMap` eller en `TreeMap`, så klassen `Dag` måste förberedas för att fungera som nycklar i båda. Komplettera klassen `Dag` så att den kan användas som nyckel både i en `HashMap` och i en `TreeMap`

b) Programmet som kommer att använda klassen `Dag` kommer att ha en
`Map<Dag, String> vaktSchema;`
med objekt av klassen `Dag` som nycklar och medlemmarnas namn som värden.

Men man vill även kunna få fram för varje medlem, vilka dagar som denna har bokat (obs att en medlem kan ha bokat flera dagar). Man vill därför ha en metod som tar ett sådant vaktSchema som argument och som returnerar en annan Map, där medlemmarnas namn är nyckel, och vilka dagar de har bokat är värden.

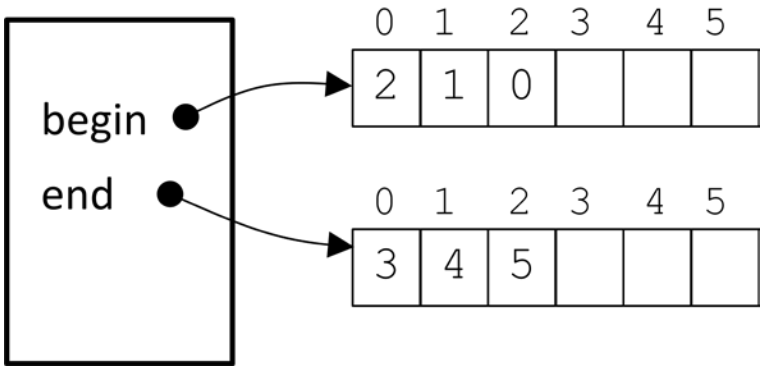
Skriv denna metod. Den ska vara en statisk metod som alltså tar en Map<Dag, String> som argument och den ska skapa och returnera den inverterade Map:en. I den nya Map:en ska värdena ligga i sorteringsordning efter medlemmarnas namn (så att eventuell utskrift skulle bli i den ordningen, fast du ska inte göra någon utskrift).

Uppgift 3 (13 poäng)

Denna uppgift går ut på att skriva ett generiskt gränssnitt (*interface*) `Deque<E>` och en generisk klass `ArrayDeque<E>` som implementerar detta gränssnitt.

Deque står för *Double-ended queue* och är en datastruktur där man kan stoppa in resp. ta bort element både i början och i slutet av datastrukturen med samma effektivitet. Dessutom kan man hämta ett godtyckligt element genom att indexera.

Deque skule kunna implementeras med en länkad lista, men då blir indexeringsoperationen mycket ineffektiv. Om den implementeras med hjälp av en `ArrayList` så blir insättningar och borttag i början ineffektiva.
Istället brukar Deque implementeras genom att använda två `ArrayList`:or: en för början av datastrukturen och en för slutet av den. `ArrayList`:an för början av datastrukturen är lagrad baklänges, så istället för att stoppa in eller ta bort element i början på den (ineffektivt) stoppar man in eller tar bort dem i slutet på den (effektivt).



Tillämpningar som använder `ArrayDeque` vet givetvis inget om de två `ArrayList`:or, för dem ska `ArrayDeque` bete sig som om den var en sammanhängande sekvens av element.

`Deque<E>` och `ArrayDeque<E>` ska ha följande metoder:

- `addFirst` – tar som argument ett objekt av rätt typ och adderar det i början av datastrukturen (dvs i slutet av den `ArrayList`:an som tjänstgör som början)
- `addLast` - tar som argument ett objekt av rätt typ och adderar det i slutet av datastrukturen (dvs i slutet av den `ArrayList`:an som tjänstgör som slutet)
- `get` – tar ett heltalsindex och returnerar referensen till elementet på denna position. Ger `IndexOutOfBoundsException` om indexet är för stort eller mindre än 0. Obs att indexet måste översättas till rätt position i någon av de två `ArrayList`:or
- `size` – returnerar hur många element som finns i datastrukturen

Det borde givetvis finnas metoder för att ta bort första resp. sista elementet, men du kan bortse här från dessa metoder.

Skriv gränssnittet `Deque<E>` och klassen `ArrayDeque<E>` som implementerar detta gränssnitt.

Lösningsförslag PROG2 VT14 tenta 2014-05-02

Uppgift 1

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

class KartPanel extends JPanel{
    private ImageIcon bilden;

    public KartPanel(String filnamn){
          setLayout(null);
          bilden = new ImageIcon(filnamn);
    }

    protected void paintComponent(Graphics g){
          super.paintComponent(g);
          g.drawImage(bilden.getImage(),0,0,this);
    }
}

class Kartan extends JFrame{
     int meterPerPixel;
     int previousX = 0, previousY = 0;
     double sträcka = 0.0;

     JTextField meterPerPixelFält;
     JLabel sträckaLabel;
     KartPanel kp;

    Kartan(String filnamn){
          JPanel norra = new JPanel();
          add(norra, BorderLayout.NORTH);
          norra.add(new JLabel("Sträcka: "));
          sträckaLabel = new JLabel("0");
          norra.add(sträckaLabel);
          norra.add(new JLabel("meter"));

          kp = new KartPanel(filnamn);
          add(kp);

          JPanel södra = new JPanel();
          add(södra, BorderLayout.SOUTH);
          södra.add(new JLabel("En pixel motsvarar: "));
          meterPerPixelFält = new JTextField(3);
          södra.add(meterPerPixelFält);
          södra.add(new JLabel("meter"));
          JButton okButton = new JButton("Ok");
          södra.add(okButton);
          okButton.addActionListener(new MeterPerPixelLyss());

          setDefaultCloseOperation(EXIT_ON_CLOSE);
          setSize(650,350);
          setVisible(true);
    }
```

```
    class MeterPerPixelLyss implements ActionListener{
       public void actionPerformed(ActionEvent ave){
          try{
             meterPerPixel = Integer.parseInt(meterPerPixelFält.getText());
             kp.addMouseListener(new KlickLyss());
          }catch(NumberFormatException e){
             JOptionPane.showMessageDialog(Kartan.this, "Fel!");
          }
       }
    }

    class KlickLyss extends MouseAdapter{
       public void mouseClicked(MouseEvent mev){
          int x = mev.getX();
          int y = mev.getY();
          if (previousX != 0){
             double avstånd = Math.sqrt(Math.pow(x - previousX, 2) +
                                        Math.pow(y - previousY, 2));
             sträcka += avstånd * meterPerPixel;
             sträckaLabel.setText("" + (int)sträcka);
          }
          previousX = x;
          previousY = y;
       }
    }

    public static void main(String[] args){
          new Kartan(args[0]);
    }
}
```

Uppgift 2

```
class Dag implements Comparable<Dag>{
    private int veckoNr, veckoDag;
    private static final String[] VDAGAR =
    {"Måndag","Tisdag","Onsdag","Torsdag","Fredag","Lördag","Söndag"};

    public Dag(int veckoNr, int veckoDag){
       if (veckoNr < 1 || veckoNr > 53 || veckoDag < 0 || veckoDag > 6)
          throw new IllegalArgumentException();
       this.veckoNr = veckoNr;
       this.veckoDag = veckoDag;
    }
    public String toString() {
       return "vecka " + veckoNr + " " + VDAGAR[veckoDag];
    }
    public boolean equals(Object other){
        if (other instanceof Dag){
           Dag d = (Dag)other;
              return veckoNr == d.veckoNr && veckoDag == d.veckoDag;
        }
        else
           return false;
    }

    public int hashCode(){
         return veckoNr * 7 + veckoDag;
    }

    public int compareTo(Dag other){
       int cmp = veckoNr - other.veckoNr;
       if (cmp != 0)
          return cmp;
       return veckoDag - other.veckoDag;
    }

} // Dag
```

```
static Map<String, List<Dag>> perMedlem(Map<Dag,String> vaktSchema){
   Map<String, List<Dag>> nyMap = new TreeMap<>();
   for(Map.Entry<Dag,String> me : vaktSchema.entrySet()){
      Dag dag = me.getKey();
      String namn = me.getValue();

      List<Dag> dagLista = nyMap.get(namn);
      if (dagLista == null){
         dagLista = new ArrayList<Dag>();
         nyMap.put(namn, dagLista);
      }
      dagLista.add(dag);
   } // for
   return nyMap;
}
```

Uppgift 3

```
public interface Deque<E>{
   void addLast(E val);
   void addFirst(E val);
   E get(int index);
   int size();
}
```

```
import java.util.*;

public class ArrayDeque<E> implements Deque<E>{
   private ArrayList<E> begin = new ArrayList<E>();
   private ArrayList<E> end = new ArrayList<E>();

   public void addFirst(E val){
      begin.add(val);
   }

   public void addLast(E val){
      end.add(val);
   }

   public E get(int index){
      if (index < 0 || index > size())
         throw new IndexOutOfBoundsException("Fel i ArrayDeque.get");

      if (index < begin.size()){
         int beginIndex = begin.size()-1-index;
         return begin.get(beginIndex);
      }
      else{
         int endIndex = index - begin.size();
         return end.get(endIndex);
      }
    }

    public int size(){
         return begin.size() + end.size();
    }
}
```