PROG2 Tenta 2014-01-17

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.

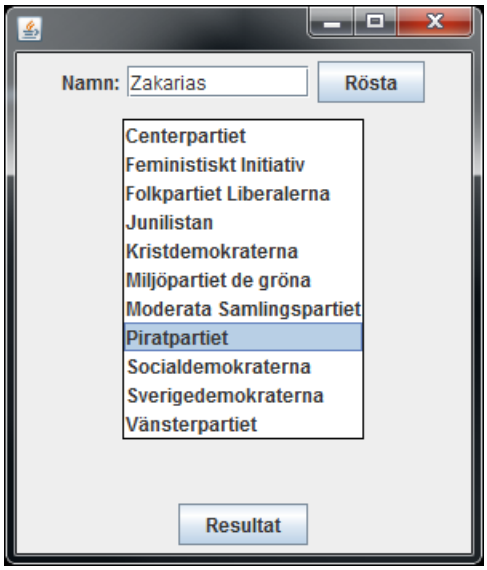
Fx - En lösning är helt felaktig eller saknas medan de övriga lösningar är i princip korrekta, men kan vara behäftade med fel och/eller underlåta att ta hänsyn till någon enstaka angiven situation eller begärd felkontroll.

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 (12 poäng)

Den 25 maj i år är det val till EU-parlamentet i Sverige. I en viss organisation har man bestämt sig för att göra ett provval som ett slags opinionsundersökning. Man vill göra detta med ett Java-program som skapar ett fönster med följande utseende (nedan till vänster):

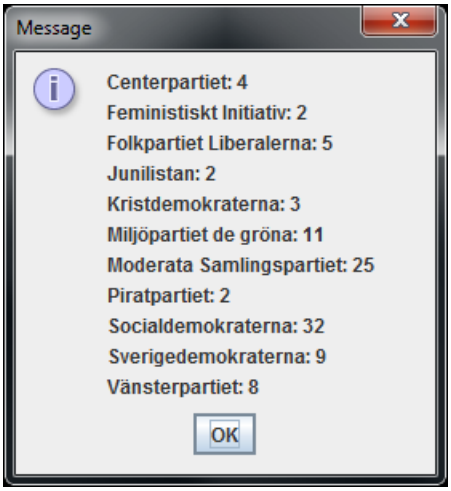


I mitten finns en lista med de partier man kan rösta på. När man ska rösta skriver man in sitt namn i textfältet upptill, väljer ett parti i listan och trycker på knappen "Rösta" (ovan till höger). Om man har glömt att skriva in ett namn eller välja ett parti så ska ett felmeddelande visas i en dialogruta.

Man får givetvis inte registrera vem som röstat på vilket parti, namnet är bara till för att kontrollera att personen med det namnet inte redan har röstat. Om personen med det inmatade namnet redan har röstat ska det komma ett felmeddelande i en dialogruta (vi antar att namnen är unika).

Efter att man har röstat ska namnfältet blankställas och markeringen från listan tas bort.

För att avsluta provvalet kan man trycka på knappen "Resultat". Då ska det visas en dialogruta med partierna och hur många röster de har fått (se nedan).



Samtidigt ska knappen "Rösta" göras otryckbar (man får inte rösta efter att ha sett resultatet).

Lite tips: man får fram det valda alternativet från en `JList` med metoden `getSelectedValue()`.
Finns det inget valt alternativ returnerar denna metod `null`.
Man tar bort markeringar från en `JList` med metoden `clearSelection()`.

Se nästa sida för kod som ska kompletteras.

Nedan finns koden till klassen `EUVal` som du ska modifiera/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 EUVal extends JFrame{
6         final static String[] PARTIER =
7         {"Centerpartiet","Feministiskt Initiativ",
8          "Folkpartiet Liberalerna","Junilistan","Kristdemokraterna",
9          "Miljöpartiet de gröna","Moderata Samlingspartiet","Piratpartiet",
10         "Socialdemokraterna","Sverigedemokraterna","Vänsterpartiet"};
11
12        EUVal(){
13          JPanel uppe = new JPanel();
14          add(uppe, BorderLayout.NORTH);
15          uppe.add(new JLabel("Namn:"));
16          uppe.add(new JTextField(10));
17          uppe.add(new JButton("Rösta"));
18
19          JPanel center = new JPanel();
20          JList<String> partier=new JList<String>(PARTIER);
21          partier.setVisibleRowCount(PARTIER.length);
22          partier.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
23          center.add(partier);
24          add(center, BorderLayout.CENTER);
25
26          JPanel nere = new JPanel();
27          add(nere, BorderLayout.SOUTH);
28          nere.add(new JButton("Resultat"));
29
30          setDefaultCloseOperation(EXIT_ON_CLOSE);
31          setSize(300,350);
32          setVisible(true);
33        }
34
35        public static void main(String[] args){
36          new EUVal();
37        }
38    }
```

Uppgift 2 (13 poäng)

På ett visst företag ska man identifiera sina kunder med telefonnummer.
Man tänker sig använda följande två klasser:

```
public class TelNr{
    private int riktNr, abNr;

    public TelNr(int riktNr, int abNr){
      this.riktNr = riktNr;
      this.abNr = abNr;
    }
    public int getRiktNr() { return riktNr; }
    public int getAbonnentNr() { return abNr; }
    public String toString() {
      return "0" + riktNr + " - " + abNr;
    }
} // Telnr

class Kund{
    private String namn;
    private TelNr tel;
    // Andra attribut, oväsentliga här

    public Kund(String namn, TelNr tel){
      this.namn = namn;
      this.tel = tel;
    }
    public String getNamn() { return namn; }
    public TelNr getTel() { return tel; }
    public void setTel(TelNr nyttTel) { this.tel = nyttTel; }
    public String toString() {
      return namn + " " + tel;
    }
} // Kund
```

a) **förbered** klassen `TelNr` för att användas som nyckel i en `HashMap`

b) efter att en `HashMap<TelNr, Kund>` har använts en längre period inser man att många kunder har bytt telefonnummer med hjälp av metoden `setTel()` i klassen `Kund`. Man har alltså ändrat telefonnumret hos `Kund`-objektet utan att ändra nyckeln i Map:en, så dessa `Kund`-objekt finns under fel nyckel i Map:en. Man behöver nu därför en metod

```
static void repair(Map<TelNr, Kund> theMap);
```

som hittar alla `TelNr`/`Kund`-par där nyckeln i Map:en inte stämmer med `tel` i `Kund`-objektet, tar bort detta par ur Map:en och sätter in `Kund`-objektet i Map:en igen men med rätt TelNr som nyckel.

Dessutom vill man att denna metod skriver ut en lista på de `Kund`er som man uppdaterat på detta sätt, men i alfabetisk ordning efter `Kund`ernas namn.

Skriv denna metod.

Obs! Kom ihåg vid borttagandet av paren från Map:en att man inte får ta bort element direkt ur en datastruktur under iteration över denna datastruktur.

Uppgift 3 (13 poäng)

Den här uppgiften handlar om att skapa en återanvändbar datastruktur kallad `MinMaxQueue`. Detta är en datastruktur där man kan stoppa in objekt, man kan ta bort det största objektet som just nu finns i datastrukturen (och samtidigt få en referens till det borttagna objektet), och samma sak för det minsta objektet.

Sådana datastrukturer brukar implementeras på olika komplicerade sätt för att öka deras effektivitet, men om man struntar i effektiviteten så kan man implementera den med en vanlig `ArrayList`, på nästa sida finns koden för en sådan `ArrayMinMaxQueue` för heltal.

Metoden `add()` adderar ett nytt objekt till samlingen, `removeMax()` letar upp det största objektet, tar bort det ur samlingen och returnerer det och `removeMin()` gör motsvarande för det minsta objektet. Metoden `isEmpty()` returnerar `true` om samlingen är tom.

För att minska upprepning av kod använder remove-metoderna hjälpmetoden `findPos` som letar efter positionen för det största resp. minsta värdet. Denna metod är en intern arbetsmetod.

Din uppgift är att göra följande:

- man ska kunna hantera godtyckliga objekt i klassen, inte bara heltal. Gör om klassen till en generisk klass med elementtypen som generisk parameter.
  Obs att du måste ange att man ska kunna jämföra elementen, och att du även
  måste skriva om jämförelserna (som nu görs med operatorerna `<` resp. `>`, vilket bara kan göras med element av primitiva numeriska typer)

- `MinMaxQueue` kan implementeras på olika sätt men med samma funktionalitet. Man kan underlätta utbytbarheten mellan olika implementeringar genom att deklarera funktionaliteten i ett gränssnitt (*interface*), t.ex. med namnet just `MinMaxQueue`.
  Skriv detta gränssnitt och se till att din `ArrayList`-implementering deklarerar att den uppfyller gränssnittet

- lägg gränssnittet och klassen med `ArrayList`-implementeringen i ett paket `minmax`

- klassen och dess attribut och metoder är deklarerade utan några synlighetsmodifierare (`public` o.s.v.). Komplettera med lämpliga synlighetsmodifierare

- hjälpmetoden `findPos()` utgår från att det finns minst ett element i datastrukturen. Komplettera metoden så att undantaget `NoSuchElementException` genereras om det inte finns några element i datastrukturen

```
import java.util.*;

class ArrayMinMaxQueue{
    ArrayList<Integer> data = new ArrayList<Integer>();

    // Lägger till ett värde i datastrukturen
    void add(int value){
       data.add(value);
    }

    // Hjälpmetod som hittar positionen för största
    // eller minsta värdet beroende på argumentet forMax
    int findPos(boolean forMax){
       int pos = 0;
       if (forMax)
         for(int i = 1; i < data.size(); i++){
            if (data.get(pos) < data.get(i))
               pos = i;
         }
       else
         for(int i = 1; i < data.size(); i++){
            if (data.get(pos) > data.get(i))
               pos = i;
         }
       return pos;
    }

    // Tar bort och returnerar största värdet
    int removeMax(){
       int pos = findPos(true);
       int max = data.get(pos);
       data.remove(pos);
       return max;
    }

    // Tar bort och returnerar minsta värdet
    int removeMin(){
       int pos = findPos(false);
       int min = data.get(pos);
       data.remove(pos);
       return min;
    }

    // Returnerar true om datastrukturen är tom
    boolean isEmpty(){
     return data.isEmpty();
    }
}
```

Lösningsförslag PROG2 HT13 tenta 2014-01-17

Uppgift 1

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

class EUVal extends JFrame{
    final static String[] PARTIER =
    {"Centerpartiet","Feministiskt Initiativ",
     "Folkpartiet Liberalerna","Junilistan","Kristdemokraterna",
     "Miljöpartiet de gröna","Moderata Samlingspartiet","Piratpartiet",
     "Socialdemokraterna","Sverigedemokraterna","Vänsterpartiet"};

     Map<String, Integer> röster = new TreeMap<String,Integer>();
     JList<String> partier=new JList<String>(PARTIER);
     Set<String> röstat = new HashSet<String>();
     JTextField namnFält;
     JButton röstButton;
    EUVal(){
       JPanel uppe = new JPanel();
       add(uppe, BorderLayout.NORTH);
       uppe.add(new JLabel("Namn:"));
       namnFält = new JTextField(10);
       uppe.add(namnFält);

       röstButton = new JButton("Rösta");
       uppe.add(röstButton);
       röstButton.addActionListener(new RöstaLyss());
       JPanel center = new JPanel();
       partier.setVisibleRowCount(PARTIER.length);
       partier.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
       center.add(partier);
       add(center, BorderLayout.CENTER);

       JPanel nere = new JPanel();
       add(nere, BorderLayout.SOUTH);
       JButton resultatButton = new JButton("Resultat");
       nere.add(resultatButton);
       resultatButton.addActionListener(new ResultatLyss());

       setDefaultCloseOperation(EXIT_ON_CLOSE);
       setSize(300,350);
       setVisible(true);
    }

    class RöstaLyss implements ActionListener{
       public void actionPerformed(ActionEvent ave){
          String namn = namnFält.getText();
          String parti = partier.getSelectedValue();
          if (namn.equals(""))
             JOptionPane.showMessageDialog(null, "Namn ej ifyllt!");
          else if (parti == null)
             JOptionPane.showMessageDialog(null, "Parti ej valt!");
          else if (röstat.contains(namn))
             JOptionPane.showMessageDialog(null, "Redan röstat!");
          else {
             röstat.add(namn);
             Integer antalRöster = röster.get(parti);
             if (antalRöster == null)
                antalRöster = 0;
             röster.put(parti, antalRöster+1);
          }
          namnFält.setText("");
          partier.clearSelection();
          }
    }
```

```
   class ResultatLyss implements ActionListener{
      public void actionPerformed(ActionEvent ave){
         namnFält.setEnabled(false);
         röstButton.setEnabled(false);
         String str = "";
         for(Map.Entry<String,Integer> me : röster.entrySet())
            str += me.getKey()+": " + me.getValue() + "\n";
         JOptionPane.showMessageDialog(null, str);
      }
   }

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

Uppgift 2

```
class TelNr{
    private int riktNr, abNr;

    public TelNr(int riktNr, int abNr){
          this.riktNr = riktNr;
          this.abNr = abNr;
    }

    public int getRiktNr() { return riktNr; }
    public int getAbonnentNr() { return abNr; }
    public String toString() {
          return "0"+riktNr+" - "+abNr;
    }
   public boolean equals(Object other){
      if (other instanceof TelNr){
         TelNr t = (TelNr)other;
         return riktNr == t.riktNr && abNr == t.abNr;
      }
      else
         return false;
   }

   public int hashCode(){
      return riktNr + abNr * 1000;
   }
} // Telnr
```

```
class NamnCmp implements Comparator<Kund>{
   public int compare(Kund k1, Kund k2){
      return k1.getNamn().compareTo(k2.getNamn());
   }
}
static void repair(Map<TelNr, Kund> kunder){
   Set<TelNr> tels = new HashSet<TelNr>();
   for(Map.Entry<TelNr,Kund> me : kunder.entrySet() ){
      TelNr key = me.getKey();
      Kund kund = me.getValue();
      if (!key.equals(kund.getTel()))
         tels.add(key);
  }
  List<Kund> forUtskrift = new ArrayList<Kund>();
  for(TelNr tnr : tels) {
     Kund k = kunder.remove(tnr);
     kunder.put(k.getTel(), k);
     forUtskrift.add(k);
  }
  Collections.sort(forUtskrift, new NamnCmp());
  for(Kund k : forUtskrift)
     System.out.println(k);
}
```

Uppgift 3

```
package minmax;

public interface MinMaxQueue<E extends Comparable<E>>{
   void add(E value);
   E removeMax();
   E removeMin();
   boolean isEmpty();
}
```

```
package minmax;
import java.util.*;
public class ArrayMinMaxQueue<E extends Comparable<E>>
                                implements MinMaxQueue<E>{
    private ArrayList<E> data = new ArrayList<E>();

    public void add(E value){
       data.add(value);
    }

    private int findPos(boolean forMax){
       if (data.isEmpty())
          throw new NoSuchElementException("Tom kö!");

       int pos = 0;
       if (forMax)
          for(int i = 1; i < data.size(); i++){
             if (data.get(pos).compareTo(data.get(i)) < 0)
                pos = i;
          }
       else
           for(int i = 1; i < data.size(); i++){
             if (data.get(pos).compareTo(data.get(i)) > 0)
                pos = i;
           }
       return pos;
    }

    public E removeMax(){
       int pos = findPos(true);
       E max = data.get(pos);
       data.remove(pos);
       return max;
    }

    public E removeMin(){
       int pos = findPos(false);
       E min = data.get(pos);
       data.remove(pos);
       return min;
    }

    public boolean isEmpty(){
       return data.isEmpty();
    }
}
```