Programmeringsteknik för I1
Övning 4

Övningsgrupp 2 (Sal E32/D32)

Johannes Hjorth
hjorth@nada.kth.se
Rum 4538 på plan 5 i D-huset
08 - 790 69 02

Kurshemsida:
http://www.nada.kth.se/kurser/kth/2D1310/

Övningsanteckningar och diagnostiska prov:
http://www.nada.kth.se/∼hjorth/teaching/

Administrativt

Kontrollera att ni har fått LAB4 inrapporterad i res-systemet genom att skriva

```
>res show prgi04
```

```
RAPPORTERADE RESULTAT FÖR: ÅÅMMDD-XXXX Efternamn, Förnamn
ANVNAMN: namn STUDIESTATUS: I-04 GRUPP: 2

Moment   Nr    Datum      Resultat   Rapp av

lab      2     041112     G          hjorth
lab      3     041118     G          hjorth
lab      4     041125     G          hjorth
```

Säg till om något är fel!

Kom ihåg de diagnostiska proven! Frågorna som kommer på tentan kommer vara snarlika, men inte exakt likadana.

Jag länkar till proven från min websida

`http://www.nada.kth.se/∼hjorth/teaching`

Dagens program

Repetera metoder, instanser och klasser

De hjälper dig strukturera upp ditt program.

Hur bygger man större program?

Vad är viktigt att tänka på?
Kompilera så tidigt som möjligt!

Vad är `public` och `private` bra för?

När ska vi använda dem, och varför?

Bli vän med Java API och dess dokumentation

Information om Javas inbyggda klasser och metoder

Vi äger en pankomat!

Vi vill skriva ett program för att styra vår pankomat! En pankomat är en apparat där blåögda personer kan sätta in sina pengar på ett privat konto, för att vid ett senare tillfälle förhoppningsvis kunna ta ut dem igen.

Allt ska skötas från ett tjusigt menysystem. Vi vill kunna läsa in kontona från fil när programmet startar och sedan skriva ner uppdateringarna till fil när vi avslutar programmet.

Första utkastet

Var börjar vi? Första steget är att fundera ut hur vi kan dela upp vår uppgift i mindre delar och vilka data som programmet behöver kunna hålla reda på för att lösa uppgiften.

Därefter skriver vi ett programskelett med tomma metoder. Tanken är att det ska gå att kompilera så snart som möjligt. Då kan vi nämligen dra nytta av kompilatorn för att hitta fel enklare.

Verifiera att en metod är korrekt innan ni går vidare med nästa.

Vi går igenom alla steg på övningen, nedan följer bara ett exempel på hur vi skulle kunna lösa uppgiften. Lösningen vi kommer fram till på tavlan kommer antagligen skilja sig från denna.

Konto.java

Detta är bara ett förslag på hur problemet kan lösas.

```
public class Konto {

    private static double ränta = 0.01;
    private double saldo;
    private String ägare;

    Konto(String ägare) { // Konstruktor A
        this.ägare = ägare;
        saldo = 0;
    }

    Konto(String ägare, double insättning) { // Konstruktor B
        this.ägare = ägare;
        saldo = insättning;
    }

    public static void sättRänta(double ränta) {
        Konto.ränta = ränta;
    }

    public static double ränta() {
        return ränta;
    }

    public void sättIn(double belopp) {
        saldo += belopp;
    }

    public boolean taUt(double belopp) {
        boolean giltigtUtag = false;

        if(belopp <= saldo) {
            saldo -= belopp;
            giltigtUtag = true;
        }

        return giltigtUtag;
    }

    public double kollaSaldo() {
        return saldo;
    }

    public String ägare() {
        return ägare;
    }

    public void läggTillRänta(int dagar) {
        saldo += ränta*saldo*(dagar/365.0); // obs, måste ha 365.0!!
    }

    public String toString() {
        return Ägare: " + ägare + " Saldo: " + saldo + " kronor.";
    }
}
```

Två nya saker är värt att notera i `Konto.java`.

- En klass ha fler än en konstruktor.
- Om två `int` delas med varandra får vi heltalsdivision, vilket inte alltid är önskvärt. För att inte detta ska ske skriver vi antalet dagar som decimaltalet `365.0`.

Pankomat.java

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

public class Pankomat {

    private ArrayList konton;
    private String dataFil;
    private BufferedReader inmatning;

    public static void main(String args[]) throws IOException {
        Pankomat bank = new Pankomat("konton.txt");

        bank.skrivUtMeny();
        while(bank.hanteraMeny()) {
            bank.skrivUtMeny();
        }
    }

    Pankomat(String filnamn) { // Konstruktor

        dataFil = filnamn;
        konton = new ArrayList();
        inmatning = new BufferedReader(new InputStreamReader(System.in));

        läsInFil(filnamn); // läs in kontoinformationen

    }

    public void skrivUtMeny() {
        System.out.println("1. Skapa nytt konto");
        System.out.println("2. Sätt in pengar");
        System.out.println("3. Ta ut pengar");
        System.out.println("4. Lista konton");
        System.out.println("5. Avsluta programmet");
        System.out.print("Ditt val: ");
    }
```

```
public boolean hanteraMeny() throws IOException {

    boolean ejKlar = true; // dags att avsluta programmet?

    int val = Integer.parseInt(inmatning.readLine());

    switch(val) {
    case 1:
        skapaKonto();
        break;
    case 2:
        System.out.println("Ännu inte implementerat!");
        break;
    case 3:
        if(taUtPengar()) {
            System.out.println("Pengarna uttagna, ha en bra dag!");
        } else {
            System.out.println("Så mycket pengar finns inte på kontot!");
        }
        break;
    case 4:
        System.out.println(toString());
        break;
    case 5:
        System.out.println("Ha en bra dag!");
        skrivTillFil(dataFil);
        ejKlar = false;
        break;
    default:
        System.out.println("Fel inmatning!");
    }

    return ejKlar;
}
```

```
public void skapaKonto() throws IOException {
    String namn;
    double belopp;

    System.out.print("Vad heter du? Namn: ");
    namn = inmatning.readLine();

    System.out.print("Hur mycket vill du ha på kontot? Belopp: ");
    belopp = Double.parseDouble(inmatning.readLine());

    konton.add(new Konto(namn, belopp));
}

public boolean taUtPengar() throws IOException {
    Konto k = null;
    String namn;
    double belopp;
    boolean uttagLyckades;

    while(k == null) { // null om ej giltigt konto

        System.out.print("Vems konto vill du ta ut pengar från? Namn: ");
        namn = inmatning.readLine();

        k = hittaKonto(namn);
    }

    System.out.print("Hur mycket pengar vill du ta ut? Belopp: ");
    belopp = Double.parseDouble(inmatning.readLine());

    uttagLyckades = k.taUt(belopp);

    return uttagLyckades;
}
```

```
private Konto hittaKonto(String ägare) { // retunerar null om ej finns
    int i = 0;
    Konto k = null, tmp;

    while(k == null && i < konton.size()) {
        tmp = (Konto) konton.get(i);

        if(ägare.compareTo(tmp.ägare()) == 0)
            k = tmp;

        i++; // obs glöm ej!!
    }

    return k;
}

public String toString() {
    StringBuffer s = new StringBuffer();
    Konto tmp;

    s.append("Räntan är " + Konto.ränta() + "\n");

    for(int i = 0; i < konton.size(); i++) {
        tmp = (Konto) konton.get(i);
        s.append(tmp.toString() + "\n");
    }

    return s.toString();
}

private void läsInFil(String filnamn) {
    SimpleTextFileReader inFil = new SimpleTextFileReader(filnamn);
    StringTokenizer ord;
    String rad = inFil.readLine();
    String ägare;
    double saldo;

    // specialbehandla första raden (räntan)
    ord = new StringTokenizer(rad);
    ord.nextToken(); // kasta bort texten "Ränta:"
    Konto.sättRänta(Double.parseDouble(ord.nextToken()));
```

```
    // Läs in alla konton
    rad = inFil.readLine();

    while(rad  != null) {
        // System.out.println("Inläst rad: " + rad);
        ord = new StringTokenizer(rad);

        ord.nextToken(); // kasta bort texten "Ägare:"
        ägare = ord.nextToken();

        ord.nextToken(); // kasta bort texten "Saldo:"
        saldo = Double.parseDouble(ord.nextToken());

        // spara undan kontot i arrayen
        konton.add(new Konto(ägare, saldo));

        // läs in nästa rad
        rad = inFil.readLine();
    }

    inFil.close(); // stäng filen
}

private void skrivTillFil(String filnamn) {

    SimpleTextFileWriter utFil = new SimpleTextFileWriter(filnamn);

    utFil.println("Ränta: " + Konto.ränta());

    for(int i = 0; i < konton.size(); i++) {
        utFil.println((Konto) konton.get(i));
    }

  utFil.close(); // stäng filen
}
}
```

I Pankomat.java möter vi ett flertal nya klasser och metoder. Slå gärna upp StringBuffer och StringTokenizer i JAVA API för att se vilka metoder dessa klasser tillhandahåller.

Anledningen till att vi använder StringBuffer är att den tillåter att man ändra på strängen den innehåller, till skillnad från String vilkars sträng är en konstant efter det att den väl skapats.

För att kunna dela upp de från fil inlästa raderna i "ord" använder vi oss av StringTokenizer.

Indatafilen konton.txt ser ut som följer:

```
Ränta: 0.01
Ägare: Johannes Saldo: 100.0 kronor.
Ägare: Annika Saldo: 150000.0 kronor.
Ägare: Sten Saldo: 1000.0 kronor.
Ägare: Lisa Saldo: 14000.0 kronor.
```

Vi kör koden

```
>javac Pankomat.java
>java Pankomat
1. Skapa nytt konto
2. Sätt in pengar
3. Ta ut pengar
4. Lista konton
5. Avsluta programmet
Ditt val: 4
Räntan är 0.01
Ägare: Johannes Saldo: 100.0 kronor.
Ägare: Annika Saldo: 150000.0 kronor.
Ägare: Sten Saldo: 1000.0 kronor.
Ägare: Lisa Saldo: 14000.0 kronor.

1. Skapa nytt konto
2. Sätt in pengar
3. Ta ut pengar
4. Lista konton
5. Avsluta programmet
Ditt val: 3
Vems konto vill du ta ut pengar från? Namn: Johannes
Hur mycket pengar vill du ta ut? Belopp: 26
Pengarna uttagna, ha en bra dag!
```

SimpleTextFileReader.java

Det här är inte en inbyggd klass, utan den har skrivits av några lärare på Nada. Ni hittar filen i kurskatalogen tillsammans med SimpleTextFileWriter.java. De gör en del av felhanteringen vid inläsning från fil och utskrift till fil.

```
import java.io.*;

class SimpleTextFileReader {
   private BufferedReader in;

   // Skapa en förbindelse för att läsa från stdin.
   public SimpleTextFileReader() {
      in = new BufferedReader(new InputStreamReader(System.in));
   }

   // Skapa en förbindelse för att läsa från filen filename.
   public SimpleTextFileReader(String filename) {
      try {
         in = new BufferedReader(new FileReader(filename));
      } catch(FileNotFoundException e) {
         e.printStackTrace();
         System.exit(0);
      }
   }

   // Läser en textrad. Returnerar null vid filslut.
   public String readLine() {
      String str = null;
      try {
         str = in.readLine();
      } catch(IOException e) {
         e.printStackTrace();
         System.exit(0);
      }
      return str;
   }

   // Stänger förbindelsen till filen.
   public void close() {
      try {
         in.close();
      } catch(IOException e) {
         e.printStackTrace();
         System.exit(0);
      }
   }

   // Test
   public static void main(String[] args) {
      // Öppna en förbindelse till stdin.
      SimpleTextFileReader stdin = new SimpleTextFileReader();

      // Läs en textrad från stdin
      System.out.println("Vad heter du?");
      String str = stdin.readLine();
      System.out.println("Hej " + str + "!");
   }
}
```