Laboration 2 – Containerklasser

Avsikten med laborationen är att du ska träna på att använda ett par containerklasser som finns i java. Skapa paketet ***laboration2*** innan du börjar lösa uppgifterna. txt-filerna som används i laborationen ska placeras i katalogen *files* som ska vara i projektet.

Uppgift 1 – ArrayList

Klasserna ***Person*** och ***Exercise1*** är givna. Studera *main*-metoden i klassen *Exercise1*.

- På första raden anropas metoden ***readPersons***. Metoden läser innehållet i filen *personer.txt* rad för rad. Av varje rad i textfilen skapas ett *Person*-objekt vilket lagras i en *ArrayList*. Referensen till ArrayList-objektet returneras och referensen lagras i variabeln *list*.
  personer.txt har följande innehåll:

```
531019-1234,Sven,Andersson
690320-2345,Elin,Gustavsson
600930-3456,Roland,Ek
840102-4567,Anna,Lundh
770123-5678,Martin,Andersson
800221-6789,Alma,Dahl
770610-7890,Steve,Nelson
660503-8901,Birgitta,Asp
290129-9012,Sten,Stensson
861213-0123,Eva,Flood
```

- På andra och tredje raden skapas Person-objekt vilka används senare i main-metoden.
- På fjärde och femte raden deklareras några variabler vilka används senare i main-metoden.

På resterande rader görs anrop till diverse metoder i klassen ArrayList. Din uppgift är att räkna ut funktionen i dessa anrop, dvs. vilken funktion metoden har, och skriva ditt svar. När du är färdig med samtliga lösningar kan du jämföra dina svar med svaren längre bak i laborationen.

Anrop

A. list.toString()

B. list.get(2)

C. list.add(pers1)

D. list.add(4, pers2)

E. list.size()

F. list.indexOf(pers2)

G. list.contains(pers1)

H. list.remove(1)

I. list.clear()

Uppgift 2 – ArrayList

Klassen ***Exercise2*** är given men ett antal metoder ska kompletteras med kod. När du är färdig med en metod så testar du den för att se att körresultatet är korrekt. När du är färdig med samtliga metoder så ska du jämföra dina lösningar med lösningarna som finns längre bak i laborationen.

A printPersons()
Metoden ska skriva ut Person-objekten som lagras i *list*. Anropet

```
ex2.printPersons();
```

bör ge följande utskriften:

```
Personnummer: 531019-1234, Namn: Sven Andersson
Personnummer: 690320-2345, Namn: Elin Gustavsson
Personnummer: 600930-3456, Namn: Roland Ek
Personnummer: 840102-4567, Namn: Anna Lundh
Personnummer: 770123-5678, Namn: Martin Andersson
Personnummer: 800221-6789, Namn: Alma Dahl
Personnummer: 770610-7890, Namn: Steve Nelson
Personnummer: 660503-8901, Namn: Birgitta Asp
Personnummer: 290129-9012, Namn: Sten Stensson
Personnummer: 861213-0123, Namn: Eva Flood
```

B position(String id)
Metoden ska returnera positionen på det Person-objekt som innehåller id. Om inget sådant finns ska -1 returneras.
I din lösning ska du loopa genom listan tills du finner ett objekt med korrekt id eller listan tar slut.
Exempel:
Anropet `ex2.position( "840102-4567" );` ska returnera värdet 3
Anropet `ex2.position( "111111-2222" );` ska returnera värdet -1

C printName(String id)
Metoden ska skriva namnet på den person vars id skickas med vid anropet. Om id ej hittas så ska ett meddelande liknande "111111-2222 okänd" skrivas ut.
Exempel:
Anropet `ex2.printName( "840102-4567" );` ska skriva: `Anna Lundh`

D existsFirstName(String firstName)
Metoden ska returnera *true* om det finns en person med bifogat förnamn. Finns det inte någon person med förnamnet ska *false* returneras.
Exempel
Anropet `ex2.existsFirstName( "Edit" );` ska returnera *false*
Anropet `ex2.existsFirstName( "Anna" );` ska returnera *true*

E changeLastName(String id, String lastName)
Metoden ska söka efter ett Person-objekt med angivet id. Om sådan finns ska efternamnet ändras till angivet efternamnet som ges som argument vid anropet.
Exempel
Anropet `ex2.changeLastname( "660503-8901", "Trädrot" );` ska ändra namnet på Birgitta Asp till Birgitta Trädrot.

Uppgift 3 – ArrayList

Klassen ***Exercise2*** innehåller metoden ***position2(String id)*** vilken söker efter ett Person-objekt i *list*. Metoden ska returnera det sökta personobjektets position om det finns och annars -1 (som metoden *position* i Uppgift 2). Följande sker i metoden:

1. Först skapas ett Person-objekt med angivet personnummer (*id*).
2. Sedan anropas metoden *indexOf* i klassen ArrayList och det skapade Person-objektet är argument vid anropet.

Metoden *indexOf* returnerar positionen för ett objekt om objektet finns i listan och -1 om objektet ej finns i listan. För att jämföra objekt i listan med argumentet *p* (se anrop i *position2*) används *equals*-metoden i klassen Person.

Men klassen Person har ej överskuggat den ärvda versionen av *equals* (från klassen Object). Det är din uppgift att skriva metoden equals i klassen Person,

public boolean equals(Object obj)

så att metoden returnerar *true* om

- *obj* inte är *null* och av typen Person
- båda objekten har samma *id*

Annars ska metoden returnera *false*.

Uppgift 4 – ArrayList

Klassen ***Exercise2*** innehåller metoderna ***sort*** och ***position3*** vilka är färdigskrivna men avmarkerade (ger rödmarkering vid aktivering). Orsaken är att klassen Person måste implementera interfacet ***Comparable***

```
public class Person implements Comparable<Person> {
    :
    public int compareTo( Person p ) {
        // Lägg till kod
    }
}
```

innan *Collections.sort()* respektive *Collections.binarySearch()* anropas. *Comparable* ska implementeras på så sätt att *compareTo* ska returnera

- -1 om aktuellt objekt har lägre id än argumentet
- 0 om objekten har samma id
- 1 om aktuellt objekt har högre id än argumentet

I din lösning kan (ska) du dra nytta av *Comparable*-implementeringen i klassen String.

Om du löst uppgiften korrekt ska du få en utskrift av en ordnad lista och därefter sökresultatet från anropet av *Collections.binareySearch()*.

Uppgift 5 – HashMap

Klasserna ***Population*** och ***Exercise5*** är givna. Studera *main*-metoden i klassen *Exercise5*.

- På första raden anropas metoden ***readPopulation***. Metoden läser innehållet i filen *befolkning.txt* rad för rad. Av varje rad i textfilen skapas ett *Population*-objekt vilket lagras som ***value*** i en *HashMap*. ***key*** till Population-objektet är landets namn. Referensen till HashMap-objektet returneras och referensen lagras i variabeln *population*.
  Öppna filen *befolkning.txt* i Anteckningar så du ser vad den innehåller.
- På andra raden skapas ett *Population*-objekt vilka används senare i main-metoden.
- På tredje raden anropas klassmetoden *info*. Metoden info skriver ut innehållet i den HashMap som bifogas vid anropet. Om du kör programmet så ser du att länderna skrivs ut blandade, inte i den ordning som de lagrades.

På resterande rader görs anrop till diverse metoder i klassen *HashMap*. Din uppgift är att räkna ut funktionen i dessa anrop och skriva upp ditt svar.

Anrop

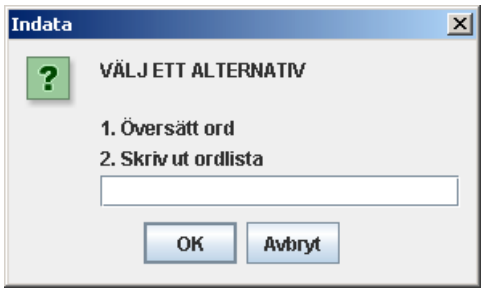
A. population.toString()
B. population.size()
C. population.get("Liberia")
D. population.containsKey("Sweden")
E. population.put("Ladonia", pop1)
F. population.remove("Sverige")
G. population.clear()
H. Studera metoden info(population) och förklara hur den fungerar.

Uppgift 6 – HashMap

I ***Exercise6*** läses par <engelskt ord,svenskt ord> in i HashMapen *dictionary* (lexikon). Det engelska ordet är alltså nyckeln och det svenska ordet värdet.

Om du kör main-metoden i *Exercise6* så visar sig en dialog med två val, 1. Översätt ord och 2. Skriv ut ordlista.

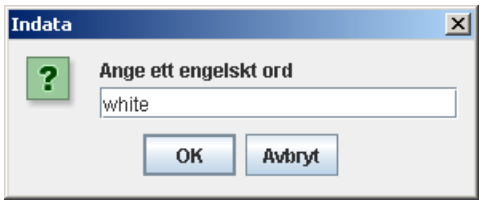
- Om du väljer det första alternativet så händer ingenting.
- Om du väljer det andra alternativet så skrivs alla par, vilka lagras i ordlistan, ut i output-fönstret.
- Om du väljer "Avbryt" så avslutas programmet.
- Om du skriver in ett alternativ som inte finns eller matar in skräp så får du göra en inmatning på nytt.

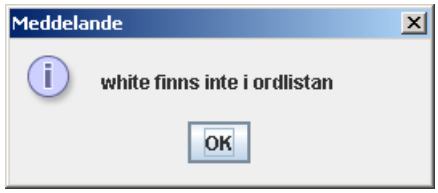


Nu ska du komplettera metodern ***translate*** så att följande händer när användaren väljer det första alternativet:

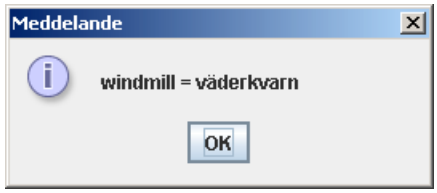
En inmatningsdialog ska visas. Dialogen frågar efter ett engelskt ord.

Om ordet inte finns i ordlistan så meddelas detta,





annars visas en översättning av ordet.



Studera slutligen metoderna

- menu
- readDictionary
- list
- program

så du verkligen förstår hur de fungerar.

Uppgift 7 – TreeMap

I Uppgift 7 ska du skriva ett program liknande programmet i Uppgift 6. Men detta program ska översätta svenska ord till skånska. Och använda en TreeMap för att lagra ord-paren.

Din lösning kommer till stora delar likna Uppgift 6. Kopiera filen Exercise6.java till Exercise7.java och ändra Exercise6 till Exercise7 i förekommande fall. Testkör sedan så att Exercise7 fungerar på samma sätt som Exercise6.

Nu ska du byta ut förekomsterna av HashMap till TreeMap och testköra programmet på nytt.

Till slut återstår att ändra i metoden readDictionary så att ordpar <svenskt ord, skånskt ord> lagras i lexikonet. På varje rad i filen SkSvEn.txt är det skånska ordet först, det svenska sedan och det engelska sist, t.ex. hoe,huvud,head. Du måste också anpassa texten i någon dialog.

Uppgift 8 – HashMap och TreeMap

Lägg till metoden
  public String translate(String)
i klasserna Exercise6 respektive Exercise7. Metoden ska översätta argumentet till svenska (Exercise6) och till skånska (Exercise7). Översättningen ska returneras av metoden.

Om ett ord inte finns i ordlistan ska metoden returnera *null*.

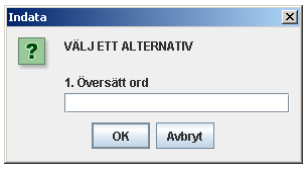
Exempel

```
Exercise6 ex6 = new Exercise6("files/SkSvEn.txt");
System.out.println( ex6.translate( "windmill" ) );   // väderkvarn
System.out.println( ex6.translate( "white" ) );   // null
```

Körresultat

```
väderkvarn
null
```

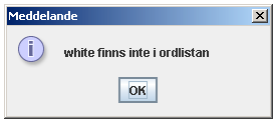
Extra



Uppgift 9 – Engelska -> skånska

Skriv ett program som översätter engelska ord till skånska ord. Programmet ska använda klasserna Exercise6 och Exercise7 för översättning (Du måste lösa Uppgift 8 innan du löser denna uppgift).

Menyn ska endast bestå av ett alternativ och användaren ska meddelas översättning eller att ordet inte finns i ordlistan.



Uppgift 10 – LinkedList

Skriv ett program vilket låter användaren göra följande operationer på en lista:

- Lägga till ord i listan (add)
- Ta bort ord ur listan (remove)
- Skriva ut orden i listan från början till slut (print)
- Skriva ut orden i listan från slutet till början (printReverse)
- Returnerar antalet lagrade ord (size)
- Skriver ut alla ord med minst sex tecken (print6)
- Returnerar antalet ord som börjar på viss bokstav (count)

Du ska använda dig av nedanstående skal till klass:

```
package laboration2;
import java.util.*;

public class Exercise10 {
    private LinkedList<String> list = new LinkedList<String>();

    public void add(String str) {

    }

    public void remove(String str) {

    }

    public void print() {

    }

    public void printReverse() {

    }

    public int size() {
        return 0; // tas bort
    }
```

```
    public void print6() {

    }

    public int count(char chr) {
        return 0; // tas bort
    }
}
```

Testa varje metod noggrant när du är klar med den.

Lösningar

Uppgift 1

A. list.toString()
Objekten som lagras i ArrayListan returneras som en sträng på formen:
`"[elem_0, elem_1, elem2,…]"`
elem_0 är det första objektet i ArrayListan.

B. list.get(2)
Metoden returnerar objektet i position 2 (det tredje objektet) i ArrayListen.

C. list.add(person1)
Objektet *person1* läggs till sist i ArrayListen. Antalet element i ArrayListen ökar med 1.

D. list.add(4, person2)
Objektet *person2* läggs till i 4:e positionen i ArrayListen (blir element nr 5). Elementen från och med postion 4 flyttas en position bakåt i listan för att ge plats för det nya elementet. Antalet element i ArrayListen ökar med 1.

E. list.size()
Metoden returnerar antalet element som lagras i ArrayListen.

F. list.indexOf(person2)
Metoden returnerar 4 vilket är objektet *person2*:s position i listan. Om person2 ej finns i listan returneras -1.

G. list.contains(person1)
Metoden returnerar *true* om *person1* finns i listan och annars *false*.

H. list.remove(1)
Metoden tar bort elementet i position 1 ur listan. Elementen från och med position 2 flyttas ett steg framåt i listan

I. list.clear()
Listan töms på alla element. Efter anropet returnerar list.size() värdet 0.

Uppgift 2

```
public void printPersons() {
    for( Person p : list ) {
        System.out.println( "Personnummer: " + p.getId() + ", Namn: " +
p.getFirstName() + " " + p.getLastName() );
    }
}

public int position( String id ) {
    Person p;
    for( int i = 0; i < list.size(); i++ ) {
        p = list.get( i );
        if( id.equals( p.getId() ) )
            return i;
    }
    return -1;
}

public void printName( String id ) {
    int index = position( id );
    if( index >= 0 ) {
        Person p = list.get( index );
        System.out.println( p.getFirstName() + " " + p.getLastName() );
    } else {
        System.out.println( id + " okänd" );
    }
}

public boolean existsFirstName( String firstName ) {
```

```
    Iterator<Person> iter = list.iterator();
    Person p;
    while( iter.hasNext() ) {
        p = iter.next();
        if( firstName.equals( p.getFirstName() ) )
            return true;
    }
    return false;
}

public boolean changeLastName( String id, String lastName ) {
    int pos = position( id );
    if( pos >= 0 ) {
        Person p = list.get( pos );
        p.setLastName( lastName );
    }
    return ( pos >= 0 );
}
```

Uppgift 3

```
public boolean equals(Object o) {
    boolean res = false
    if(o instanceof Person) {
        Person p = (Person)o;
        res = p.id.equals(id);
    }
    return res;
}
```

eller kortare

```
public boolean equals(Object o) {
    return (o instanceof Person && ((Person)o).id.equals(id));
}
```

Uppgift 4

```
public int compareTo(Person p) {
    return id.compareTo(p.getId());
}
```

Uppgift 5

A. population.toString()
   Objekten som lagras i HashMapen skrivs ut mellan måsvingar.

B. population.size()
   Metoden returnerar antalet element som lagras i HashMapen.

C. population.get("Liberia")
   Metoden returnerar *value* som hör till *key* (dvs. "Liberia"). Om *key* inte finns i HashMapen returneras *null*.
   Objektet *pers1* läggs till sist i ArrayListen. Antalet element i ArrayListen ökar med 1.

D. population.containsKey("Sweden")
   Metoden returnerar *true* om nyckeln "Sweden" finns i Hashmapen och annars *false*.

E. population.put("Ladonia", pop1)
   Paret ("Ladonia", pop1) läggs till i HashMapen. Antalet element ökar med 1.

F. population.remove("Sverige")
   Metoden tar bort paret där "Sverige" är *key* ur HashMapen. *value* returneras. Om "Sverige" inte är en nyckel returneras *null*.

G. population.clear()
   Metoden tar bort alla par som lagras i HashMapen. Efter anropet är size=0.

H. Exercise5.info(population)
Ett Iterator-objekt för nycklarna i HashMapen fås på rad ett. Denna ska användas för att gå igenom HashMapen par för par.
Medan det finns fler nycklar
erhåll en nyckel
erhåll ett värde med hjälp av nyckeln
skriv ut innehållet i värde-objektet

Uppgift 6

```
public void translate() {
    String eng, swe;
    eng = JOptionPane.showInputDialog("Ange ett engelskt ord");
    swe = dictonary.get(eng);
    if (swe == null) {
        JOptionPane.showMessageDialog(null, eng + " finns inte i
ordlistan");
    } else {
        JOptionPane.showMessageDialog(null, eng + " = " + swe);
    }
}
```

Uppgift 7

```
public class Exercise7 {

    private TreeMap<String, String> dictonary = new TreeMap<String,
String>();

    public Exercise7(String filename) {
        readDictionary(filename,dictonary);
    }

    public static void readDictionary(String filename, Map<String, String>
map) {
        try {
            BufferedReader br = new BufferedReader(new
InputStreamReader(new FileInputStream(filename),"ISO-8859-1"));
            String[] parts;
            String swedish, skanish;
            String str = br.readLine();
            while (str != null) {
                parts = str.split(",");
                swedish = parts[1];
                skanish = parts[0];
                map.put(swedish, skanish);
                str = br.readLine();
            }
            br.close();
        } catch (IOException e) {
            System.out.println("readDictionary: " + e);
        }
    }

    public void list() {..}

    public void translate() {
        String swe, ska;
        swe = JOptionPane.showInputDialog("Ange ett svenskt ord");
        ska = dictonary.get(swe);
        if (ska == null) {
```

```
            JOptionPane.showMessageDialog(null, swe + " finns inte i
ordlistan");
        } else {
            JOptionPane.showMessageDialog(null, swe + " = " + ska);
        }
    }

    public static int menu(String[] options) {..}

    public static void main(String[] args) {
        Exercise7 ex7 = new Exercise7("files/SkSvEn.txt");
        String[] menuOptions = {"Översätt ord", "Skriv ut ordlista"};
        int choice = Exercise7.menu(menuOptions);
        while (choice != 0) {
            switch (choice) {
                case 1:
                    ex7.translate();
                    break;
                case 2:
                    ex7.list();
                    break;
            }
            choice = Exercise7.menu(menuOptions);
        }
    }
}
```

Uppgift 8

```
// Exercise6
public String translate(String eng) {
    return dictonary.get(eng);
}
```

```
// Exercise7
public String translate(String swe) {
    return dictonary.get(swe);
}
```

Uppgift 9

```
public class Exercise9 {
    private Exercise6 engSwe;
    private Exercise7 sweSka;

    public Exercise9() {
        engSwe = new Exercise6("files/SkSvEn.txt");
        sweSka = new Exercise7("files/SkSvEn.txt");
    }

    public void translate() {
        String eng, swe;
        eng = JOptionPane.showInputDialog("Ange ett engelskt ord");
        swe = engSwe.translate(eng);
        if (swe == null) {
            JOptionPane.showMessageDialog(null, eng + " finns inte i ordlistan");
        } else {
            JOptionPane.showMessageDialog(null, eng + " = " +
sweSka.translate(swe));
        }

    }

    public void program() {
        String[] menuOptions = {"Översätt ord"};
        int choice = Exercise6.menu(menuOptions);
```

```
        while (choice != 0) {
            switch (choice) {
                case 1:
                    translate();
                    break;
            }
            choice = Exercise6.menu(menuOptions);
        }
    }

    public static void main(String[] args) {
        Exercise9 ex9 = new Exercise9("files/SkSvEn.txt");
        ex9.program();
    }
}
```

Uppgift 10

```
public class Exercise10 {
    private LinkedList<String> list = new LinkedList<String>();

    public void add(String str) {
        list.add(str);
    }

    public void remove(String str) {
        list.remove(str);
    }

    public void print() {
        for (String str : list) {
            System.out.println(str);
        }
    }

    public void printReverse() {
        for (int i = list.size() - 1; i >= 0; i--) {
            System.out.println(list.get(i));
        }
    }

    public int size() {
        return list.size();
    }

    public void print6() {
        for (String str : list) {
            if (str.length() >= 6) {
                System.out.println(str);
            }
        }
    }

    public int count(char chr) {
        int counter = 0;
        for (int i = 0; i < list.size(); i++) {
            if (list.get(i).charAt(0) == chr) {
                counter++;
            }
        }
        return counter;
    }
}
```