Objektorientert Programmering
Ekstraordinær eksamen
2014

Høgskolen i Østfold 2014-01-13

Emnekode	ITF10611
Emne	Objektorientert Programmering
Dato	13.01.2014
Eksamenstid	09:00 - 13:00
Hjelpemidler	To A4-ark(fire sider) med egne notater
Faglærer	Børre Stenseth

Oppgavesettet består av 10 sider inklusive denne forsiden og ett vedlegg på 6 sider. Oppgavesettet er delt på 3 oppgaver. Vedleggene er relevante for oppgave 3.

For hver oppgave er det angitt hvor stor vekt den blir tillagt i vurdering av besvarelsen. Den endelige karakteren blir satt på bakgrunn av denne vektleggingen kombinert med en helhetlig vurdering av besvarelsen.

Du er selv ansvarlig for å kontrollere at oppgavesettet er komplett.
Les gjennom alle oppgavene før du begynner å løse dem.
Husk å bruke enkle kommentarer i den koden du skriver. Dette kan gi uttelling dersom koden inneholder feil, men at du har tenkt riktig.

Lykke til

1 Begreper (20%)

I denne oppgaven skal du bruke dine egne ord og beskrive kort hva du du forstår med de aktuelle begrepene. Du må gjerne bruke korte eksempler for å vise hva du mener.

a) Hva er et **interface** , og hvorfor er dette nyttig ?

b) Forklar bruken av nøkkelordene: **try, catch, finally**

c) Forklar bruken av nøkkelordet: **final**

d) Hva betyr det at en metode er henholdsvis **private, protected** eller **public** ?

2 Javakode (20%)

I denne oppgaven skal du tolke ferdig javakode og besvare noen spørsmål.

a) Hva blir utskriften når vi kaller metoden **doIt1()** med parameter **4** ?

```
public static void doIt1(int n){
        for(int i=0;i < n;i++){
            for(int j=i; j >0;j--)
                System.out.print(" "+j);
           System.out.println();
        }
    }
```

b) Er det noe galt her, i så fall hva ?

```
public interface IA {
    public String dumpMe()
    {
        return "iA";
    }
}
```

c) Hva blir utskriften når vi kaller metoden **doIt2()** med parameter **4** ?

```
public static void doIt2(int n){
       for(int i=0;i< n;i++){
            switch(i){
               case 0: System.out.println(i);break;
               case 1: System.out.println(i);break;
               case 2: System.out.println(i);
               case 3: System.out.println(i);break;
               default:System.out.println('*');
            }
        }
}
```

d) I klassedefinisjonen nedenfor er det to feil. Analyser koden og beskriv hva som er galt. Hvordan ville du endre koden for at den skal kompilere ?

```
public class class B {
    private final int ID=0;
    public B(String id) {
        ID=Integer.parseInt(id);
    }
    static void printReport(String S){
        System.out.println("Reporting "+S+" from: "+ID);
    }
}
```

3 Konstruksjon (60%)

Din oppgave er å bidra i første fase av et prosjektet der det skal utvikles et system for et firma som driver med formidling av tolketjenester. Firmaet har tilknyttet seg en rekke tolker som behersker ulike språk. Kundene til firmaet er offentlige etater (politi, sykehus, høgskoler etc) og private firmaer som trenger hjelp med oversettelser i forbindelse med utenlandske besøk el.

Fokus i denne første fasen er å finne egnede tolker som skal møte fram fysisk og bistå med oversettelse. Det vil si at vi ser bort fra oversettelse pr telefon, Skype eller lignende og vi ser bort fra skriftlig oversettelse av dokumenter.

Oppgaven er å finne tolker som har de ønskede språkkunnskapene og som bor nærmest det stedet der jobben, oppdraget, skal gjøres.

I vedlegget finner du følgende:

•class Tolk.java

•class Oppdrag.java

•interface iGeografi.java

•class Test.java

Du skal gjøre følgende:

a) Skrive konstruktoren i class Tolk

b) Skrive metoden matchOppdrag() i class Tolk

c) Skrive metoden toString() i class Tolk

d) Skrive metoden findTolkerISammeKommune() i class Test

e) Skrive metoden findTolkerISammeFylke() i class Test

f) Skrive metoden findTolker () i class Test

Vedlegg

Vedlegg til oppgave 3

- class Tolk.java
- class Oppdrag.java
- interface iGeografi.java
- class Test.java
- Dataformat

Vedlegget er på 6 sider inklusive denne forsiden

class Tolk

```
package tolktest.tolk;

import java.util.ArrayList;

/**
 * Beskrivelse av en Tolk
 * @author bs
 */
public class Tolk {
    // tolkens navn
    private String name;
    // tolkens tlf nummer
    private String phone;
    // tolkens bostedskommune
    private String kommune;
    // språkene tolken behersker
    private ArrayList<String> languages;

    /**
     * konstruktor
     * @param oneLine En string med data for en tolk,se vedlegg
     * @throws an exception hvis data har feil format
     */
    public Tolk(String oneLine) throws Exception{

    }

    // tilgang til attributter
    public String getName(){return name;}
    public String getPhone(){return phone;}
    public String getKommune(){return kommune;}
    public ArrayList<String>getLanguages(){return languages;}

    /**
     * Sjekker at de nødvendig språkene er på
     * plass for å ta et oppdrag
     * @param job Oppdraget som skal utføres
     * @return true hvis tolken kan de to språkene, false ellers
     */
    public boolean matchOppdrag(Oppdrag job){
        System.out.println("not implemented yet");
        return false;
    }


    @Override
    public String toString(){
        return "not implemented yet";
    }
}
```

class Oppdrag

```
package tolktest.tolk;

/**
 * Beskrivelse av et oppdrag, en tolkejob
 * @author bs
 */
public class Oppdrag {
    // de to aktuelle språkene
    private String language1;
    private String language2;
    // hvor oppdraget skal gjøre
    private String kommune;

    /**
     * konstruktor
     */
    public Oppdrag(String language1, String language2,String kommune){
        this.language1=language1;
        this.language2=language2;
        this.kommune=kommune;
    }


   // tilgang til attributter
   public String getKommune(){return kommune;}
   public String[] getLanguages(){
      return new String[]{language1,language2};
    }

    @Override
    public String toString(){
        return "Oppdrag i "+kommune+" :"+
                language1+","+language2;
    }
}
```

interface iGeografi

```
package tolktest.tolk;

/**
 * Et interface som beskriver den funksjonaliteten vi
 * vil ha når vi skal se på nærhet mellom tolkens bosted
 * og tolkejobben
 * @author bs
 *
 * MERK: du skal ikke implementere dette interfacet
 *
 */

public interface iGeografi {
    // I hvilket fylke ligger en kommune
    String getFylke(String Kommune);

    // Avstanden mellom to kommuner, basert på kommunesenter
    // og reiseavstand
    long getDistance(String Kommune1,String Kommune2);
}
```

class Test

```
package tolktest.tester;

import java.util.ArrayList;
import tolktest.tolk.Geografi;
import tolktest.tolk.Oppdrag;
import tolktest.tolk.Tolk;
import tolktest.tolk.iGeografi;

/**
 * En testklasse som setter opp et datasett med tolker
 * og gjør det mulig å teste noen metoder for å finne
 * hvilke tolker som passer til et oppdrag
 * @author bs
 */
public class Test {
    // geografisk informasjon
    private iGeografi geo;

    // alle tolkene som brukes i testmaterialet
    private ArrayList<Tolk>alleTolker;

    // konstruktor
    public Test(){
        // Geografi implementerer iGeografi
        // Det er ikke din jobb å implementere Geografi
        // du kan stole på at noen andre har gjort det
        geo=new Geografi();

        // liste av alle tilgjengelige tolker
        alleTolker=new ArrayList();

        // testdata, vil bli erstattet med data fra fil
        String[] data=new String[]{
            "Ole Brum;666666;spansk,tysk,norsk;Halden",
            "Yashmin Tivari;555555;urdu,norsk;Re",
            "John Brown;121341;engelsk,russisk,norsk;Gjøvik"};

        for(int ix=0;ix < data.length;ix++){
            try{
                alleTolker.add(new Tolk(data[ix]));
            }
            catch(Exception e){
                System.out.println("feil i data: "+data[ix]);
            }
        }
    }

    //-----------------------------------------------
    // De tre metodene nedenfor har det til felles at
    // skal finne og returnere en liste av tolker
    // som kan gjøre oppdraget, theJob,
    // under litt forskjellige betingelser
    // De skal også skrive ut oppdraget og
    // de utvalgte tolkene på System.out

    // Alle tolker som kan gjøre jobben
    // der tolk og oppdrag hører hjemme i samme kommune
    public ArrayList<Tolk>findTolkerISammeKommune(Oppdrag theJob){
        System.out.println("not implemented yet");
        return null;
    }
```

```
    // Alle tolker som kan gjøre jobben
    // der tolk og oppdrag hører hjemme i samme fylke
    public ArrayList<Tolk>findTolkerISammeFylke(Oppdrag theJob){
        System.out.println("not implemented yet");
        return null;
    }

    // Alle tolker som kan gjøre jobben og som bor mindre enn
    // maxDistanse fra oppdraget, helst sortert på
    // avstand mellom tolk og oppdrag
    public ArrayList<Tolk>findTolker(Oppdrag theJob,long maxDistanse){
        System.out.println("not implemented yet");
        return null;
   }
}
```

Dataformat.

Tekstlinjer som brukes i konstruktøren i class Tolk er definert slik:

navn;telefon;språkliste;kommune

altså 4 komponenter skilt med ; (semikolon)

Språkliste er i seg selv en list med språk skilt med , (komma). En tolk kan i prinsipp beherske et vilkårlig antall språk. Ett språk er dog ikke særlig nyttig.

Eksempel på en komplett tekstlinje:

Ole Brum;666666;spansk,tysk,norsk;Halden