Programació I - 13 de gener de 2016

ENUNCIAT: En previsió de les vacances d'estiu, una operadora de vols, que ofereix informació per Internet als seus usuaris, desitja implementar un programa de gestió dels vols. Com a màxim hi hauran 50000 vols programats. La informació de cada vol és un identificador (format per una cadena de caràcters), la ciutat d'origen, la ciutat de destí, el dia i hora de partida i el dia i hora d'arribada. A més a més, per a cada vol es coneix la capacitat màxima de passatge i el nombre de passatgers que han comprat bitllet.

Es vol poder:

1. Afegir vol al quadre de vols de l'operadora.
2. Donat un cert origen, un dia de partida i un destí, es volen saber els vols directes que hi han.
3. Donats un origen i un destí, es vol conèixer els vols programats que fan una única escala (és a dir, que passen per un aeroport intermig)
4. Donats un origen, un destí i una data de partida, es vol conèixer els pics d'ocupació dels vols d'aquella data.
5. Sortir

A l'etapa de disseny s'ha decidit crear 5 classes: **`Data`** (per a mantenir les dades d'una data), **`Hora`** (per a mantenir les dades d'una hora), **`Vol`** (per a mantenir totes les dades d'un sol vol), **`Operadora`** (per a guardar la col.lecció dels vols planificats) i la classe **`Gestio`** (per a implementar el control principal de l'aplicació). S'adjunta un llistat per a veure la implementació de les classes `Data`, `Hora`, `Vol` i part de la implementació de la classe `Gestio`.

PROBLEMES A RESOLDRE I ENTREGAR: Cal entregar CADA PROBLEMA EN UN FULL PER SEPARAT i posar el Nom, Cognoms i DNI en tots els fulls.

1. (*2 punts*) Implementa la classe **`Operadora`** amb els seus atributs, el seu constructor i el mètode **afegeixVol**, considerant que sempre s'afegeix el nou vol a la primera posició de la colecció de vols.

   SOLUCIO:

```
public class Operadora {
   private Vol[] vols;
   private int numVols;

   public Operadora() {
       numVols = 0;
       vols = new Vol[Gestio.MAX_VOLS];
   }
   public Operadora (int maxVols) {
       numVols = 0;
       vols = new Vol[maxVols];
   }
   public void afegeixVol(Vol nouVol) {
       int i;
       if (numVols<Gestio.MAX_VOLS) {
          /*
            Identificació de la seq: enters des de 0 a numVols-1
            Primer():     i=numVols
            Següent(i):   i--
            FiSeq(i):     i <= 0
            Esquema:      recorregut
          */
          for (i = numVols; i>0; i--) {
               vols[i] = vols[i-1];
           }
```

```
            vols[0] = nouVol;
            numVols++;
        }
    }
```

2. (*2,5 punts*) Donada una certa ciutat d'origen, un dia de partida i una ciutat de destí, es volen llistar els vols directes que hi han entre ambdues ciutats. Contesta les següents qüestions:

   (a) Implementa el mètode **quinsVols** amb la següent capçalera:

   ```
   public Operadora quinsVols(String origen, Data partida, String desti)
   ```

   Recorda identificar la seqüència i l'esquema a aplicar per a implementar aquest mètode. Recorda també que al final de l'enunciat està la implementació de la classe Vol.
   SOLUCIO:

```
public Operadora quinsVols(String origen, Data partida, String desti) {
        Operadora auxVols;
        auxVols = new Operadora();
        /*
          Identificació de la seq: enters des de 0 a numVols-1
          Primer():     i=0
          Següent(i):   i++
          FiSeq(i):     i >= numVols
          Esquema:      recorregut
        */
        for (int i=0; i<numVols; i++) {
            if (this.vols[i].getOrigen().equals(origen) &&
                this.vols[i].getDataSort().equals(partida) &&
                this.vols[i].getDesti().equals(desti)) {
                Vol v = new Vol(this.vols[i].getCodi(), origen, desti, partid
                                this.vols[i].getHoraSort(),
                                this.vols[i].getDataArrib(),
                                this.vols[i].getHoraArrib(),
                                this.vols[i].getCapacitat());
                auxVols.afegeixVol(v);
            }
        }
        return auxVols;
    }
```

   (b) A quina classe pertany el mètode **quinsVols**? És un mètode de classe o és un mètode d'objecte? Per què?
   SOLUCIO: Pertany a la classe Operadora i és un mètode de objecte. Cal que existeixi un objecte al qual poder-lo aplicar, depèn de les dades concretes d'una instància de la classe.

   (c) Implementa el mètode **opcioQuinsVols** de la classe `Gestio`. Al final de l'enunciat trobaràs part de la implementació de la classe Gestio.
   SOLUCIO:

```
static void opcioQuinsVols(Operadora oper) {
        Scanner sc;
        String origen;
        String desti;
        Data d;
        int dia, mes, any;
        Operadora auxOper;
        sc = new Scanner(System.in);
        System.out.println("Ciutat␣origen?");
        origen = sc.nextLine();
```

```
        System.out.println("Ciutat␣final?");
        desti = sc.nextLine();
        System.out.println("Data?␣(dia␣mes␣any)");
        dia = sc.nextInt();
        mes = sc.nextInt();
        any = sc.nextInt();
        d = new Data (dia, mes, any);
        auxOper = oper.quinsVols(origen, d, desti);
        if (auxOper != null){
          System.out.println("Vols␣des␣de␣" + origen + "␣a␣" + desti + ":");
          for (int i=0; i<auxOper.getNumVols(); i++) {
            System.out.println(auxOper.getVol(i));
          }
        }
        else
          System.out.println("No␣hi␣han␣vols␣des␣de␣" + origen +
                             "␣a␣" + desti + "␣en␣aquesta␣data");
    }
```

3. (*2,5 punts*) Donats un origen i un destí, es vol conèixer si hi han vols programats que fan una única escala (és a dir, que passen per un aeroport intermig). Implementa per això el mètode **volAmbEscala**. Recorda identificar la seqüència i l'esquema a aplicar per a implementar aquest mètode.:

   ```
   public boolean volAmbEscala(String origen, String desti)
   ```

   Per a implementar **volAmbEscala** feu ús dels següents mètodes:

```
    private int obtenirVolOrigen(String origen) {
       boolean trobat;
       int i;
       trobat = false;
       i=0;
       while (i<numVols && !trobat) {
         if (vols[i].getOrigen().equals(origen)) trobat = true;
         else i++;
       }
       if (!trobat) {
         i=-1;
       }
       return i;
     }
     private int obtenirSeguentVolOrigen(int id, String origen) {
       boolean trobat;
       int i;
       trobat = false;
       i=id+1;
       while (i<numVols && !trobat) {
         if (vols[i].getOrigen().equals(origen)) trobat = true;
         else i++;
       }
       if (!trobat) {
         i=-1;
       }
       return i;
     }
     private boolean obtenirVolDesti(int id, String desti) {
       boolean trobat;
```

```
        int i;
        trobat = false;
        i=0;
        while (i<numVols && !trobat) {
          if (vols[id].getDesti().equals(vols[i].getOrigen()) &&
              vols[i].getDesti().equals(desti)) trobat = true;
          else i++;
      }
        return (trobat);
      }
```

SOLUCIO:

```
public boolean volAmbEscala(String origen, String desti) {
      boolean trobat;
      int id;
      trobat = false;
      /*
        Identificació de la seq: indexes dels vols que surten de "origen"
        Primer():     id = obtenirVolOrigen(origen)
        Següent(id):  id = obtenirSeguentVolOrigen(id, origen)
        FiSeq(id):    id == -1
        Esquema:      cerca  Condició: obtenirVolDesti(id, desti) == true
      */
      id = obtenirVolOrigen(origen);
      while (id!=-1  && !trobat) {
          if (obtenirVolDesti(id, desti)) {
              trobat = true;
          }
          id = obtenirSeguentVolOrigen(id, origen);
      }
      return (trobat);
  }
```

4. (*3 punts*) Donat un cert origen, un cert destí i una data de partida, es desitja conèixer els vols que són pics d'ocupació dels vols que aquell dia surten amb aquell origen i aquell destí. L'ocupació d'un vol és el percentatge entre el nombre de passatgers que han comprat bitllet per a aquell vol i el nombre total de passatge admès en aquell vol. Es considera que un vol és un pic d'ocupació, quan la seva ocupació és superior a l'ocupació del vol que surt abans i a l'ocupació del vol que surt després. Encara que el mètode afegeixVol no afegeix els vols ordenats, per implementar aquest mètode suposa que els vols estan ordenats per origen, i els de mateix origen, estan ordenats per destí, i els de mateix destí, estan ordenats per data i hora. Per exemple, amb les dades de la següent taula:

Taula 1: Exemple pics ocupació

IdVol	Origen	Desti	Data	Hora	OcupacioMax	Passatgers
1	AL	BCN	1 1 2016	7 30	200	50
2	BCN	MAD	8 1 2016	8 00	200	100
3	BCN	MAD	8 1 2016	8 20	200	120
4	BCN	MAD	8 1 2016	9 00	100	80
5	BCN	MAD	8 1 2016	9 30	100	60
6	BCN	MAD	8 1 2016	10 00	120	45
7	BCN	MAD	8 1 2016	10 30	120	57
8	BCN	MAD	8 1 2016	11 00	120	30
9	MAD	GRA	8 1 2016	8 30	120	30

quan es cridi el mètode picOcupacio amb BCN, MAD, 8 01 2016, donarà la informació següent:

4 BCN MAD 8 1 2016 9 00 100 80

7 BCN MAD 8 1 2016 8 30 120 57

Contesta les següents preguntes:

(a) Identifica la seqüència i l'esquema a aplicar per a implementar aquest mètode.

(b) Defineix la capçalera del mètode **picOcupacio** a la classe `Operadora` i implementa'l. Què retornarà aquest mètode? Justifica el tipus de dades que retornarà.

SOLUCIO:

El métode retorna una Operadora, que conté els vols que són pic d'ocupació.

```
private boolean esVol(int i, String origen, String desti, Data partida) {
        return (this.vols[i].getOrigen().equals(origen) &&
                this.vols[i].getDataSort().equals(partida) &&
                this.vols[i].getDesti().equals(desti));
    }

private boolean picTres(int i1, int i2, int i3) {
        return(this.vols[i1].getOcupacio() < this.vols[i2].getOcupacio() &&
               this.vols[i3].getOcupacio() < this.vols[i2].getOcupacio());
    }

public Operadora picOcupacio(String origen, String desti, Data partida) {
        Operadora auxVols = null;
        boolean trobat;
        int i;
        trobat = false;

        // cerca del primer vol amb l'origen, desti i la data d'entrada
        /*
          Identificació de la seq: enters des de 0 a numVols-1
          Primer():     i=0
          Següent(i):   i++
          FiSeq(i):     i >= numVols
          Esquema:      cerca Condic: esVol(this.vols[i], origen, desti, partida
        */
        i = 0;
        while (i<numVols && !trobat) {
            if (esVol(i, origen, desti, partida)) {
                trobat = true;
            } else i++;
        }
        if (trobat) {
            // Ara es va a analitzar els pics d'ocupacio
            // Primer element: son tres elements que cumpleixin la condicio
            int vol1, vol2, vol3;
            vol1 = i;
            if (i<numVols-2) {
                if (esVol(i+1, origen, desti, partida)) {
                    vol2 = i+1;
                    vol3 = i+2;
                    /*
                        Identificació de la seq: enters que identifiquen tres vo
```

```
                                Primer():     vol1=i; vol2=i+1; vol3=i+2;
                                Següent(i):   vol1=vol2; vol2=vol3; vol3=vol3+1;
                                FiSeq(i):     vol3 >= numVols
                                Esquema:      cerca d'un vol que no té l'origen Condició
    !esVol(vol3, origen, desti, partida)
                            */

                            while (vol3<numVols && esVol(vol3, origen, desti, partida))
                                if (picTres(vol1, vol2, vol3)) {
                                    auxVols = new Operadora();
                                    Vol v = new Vol(this.vols[vol2].getCodi(), origen, de
                                            partida,
                                            this.vols[vol2].getHoraSort(),
                                            this.vols[vol2].getDataArrib(),
                                            this.vols[vol2].getHoraArrib(),
                                            this.vols[vol2].getCapacitat());
                                    auxVols.afegeixVol(v);
                                }
                                vol1 = vol2;
                                vol2 = vol3;
                                vol3 = vol3+1;
                            }
                        }
                    }
                }
                return auxVols;
            }
```

Llistat de les classes **Data**, **Hora**, **Vol** i la classe principal de l'aplicació, **Gestió**, amb els mètodes que ja teniu implementats:

```
class Data {
    private int dia;
    private int mes;
    private int any;
    public Data(int d, int m, int a) {
          if (d>=1 && d<=31) dia = d;
          if (m>=1 && m<=12) mes = m;
          if (a>2015) any = a;
    }
    public boolean equals(Data d) {
          return(d.dia == dia && d.mes == mes && d.any == any);
    }
    public String toString(){
        return ("Dia:\t" + dia + "␣Mes:\t" +  mes + "␣Any:\t" +  any);
    }
}
class Hora {
  private int hora;
  private int minuts;

  public Hora (int h, int m) {
    if (h>=0 && h<=23) hora = h;
    if (m>=0 && m<=59) minuts = m;
  }
  public String toString(){
    return ("Hora:\t" + hora + "␣Minut:\t" +  minuts);
  }
}
```

```
public class Vol {
  private String codi;
  private String origen;
  private String desti;
  private Data dataSort;
  private Hora horaSort;
  private Data dataArrib;
  private Hora horaArrib;
  private int capacitatMax;
  private int bitlletsComprats;
  public Vol (String codi, String origen, String desti, Data dataSort,
                    Hora horaSort, Data dataArrib, Hora horaArrib, int capacitatMax) {
          this.codi = codi;
          this.origen = origen;
          this.desti = desti;
          this.dataSort = dataSort;
          this.horaSort = horaSort;
          this.dataArrib = dataArrib;
          this.horaArrib = horaArrib;
          this.capacitatMax = capacitatMax;
          this.bitlletsComprats = 0;
    }
    String getCodi() {
        return codi;
    }
    String getOrigen() {
          return (origen);
    }
    String getDesti() {
          return (desti);
    }
    Data getDataSort() {
          return dataSort;
    }
    Data getDataArrib() {
          return (dataArrib);
    }
    Hora getHoraSort() {
          return (horaSort);
    }
    Hora getHoraArrib() {
          return horaArrib;
    }
    int getCapacitat() {
          return capacitatMax;
    }
    int getBitlletsComprats() {
          return (bitlletsComprats);
    }
    float getOcupacio() {
          return (float)(capacitatMax)/(float)(bitlletsComprats);
    }
    public String toString() {
          return("***Vol␣des␣de␣"+ origen + "␣a␣" + desti +
               "\nData␣sortida:\t" + dataSort + "\nData␣arribada:\t" + dataArrib +
               "\nHora␣sortida:\t" + horaSort + "\nHora␣arribada:\t" + horaArrib +
               "\nPlaces␣lliures:␣" + (capacitatMax-bitlletsComprats));
    }
}

class Gestio {
  public  static final int AFEGIR         = 1;
  public  static final int QUINS_VOLS     = 2;
```

```
  public  static final int VOL_ESCALA     = 3;
  public  static final int PIC_OCUP       = 4;
  public  static final int SORTIR       = 5;
  public  static final int MAX_VOLS= 50000;
  public static void main (String [] args) {
    Operadora operadora = new Operadora(MAX_VOLS);
    int     op;
    op = pintaMenuPreguntaOpcio();
    while (op!=SORTIR) {
      switch(op) {
        case AFEGIR:     opcioAfegir(operadora); break;
        case QUINS_VOLS: opcioQuinsVols(operadora); break;
        case VOL_ESCALA:   opcioVolAmbEscala(operadora); break;
        case PIC_OCUP:  opcioPicOcupacio(operadora); break;
        default: break;
      }
      op = pintaMenuPreguntaOpcio();
    }
  }
}
```