Programació I - **Examen Final** - 19 de gener de 2012

ENUNCIAT: Es desitja que feu un programa de gestió d'una estació d'esquí sobre la utilització de les seves **pistes d'esquiar** durant un dia que ofereixi les següents opcions:

1. Afegir esquiador a l'estació
2. Canvi d'un esquiador de pista
3. Donat un cert instant de temps (hora, minut i segon) s'informa si tots els esquiadors que han entrat a pistes fa deu minuts estan encara esquiant a la pista "Bosquet"
4. Llistar l'ocupació dels diferents tipus de pistes
5. Sortida d'un esquiador de les pistes
6. Acabar

Per a cada esquiador, com a mínim, s'ha de guardar el seu número de forfait (identificador únic), l'hora, el minut i els segons que indica l'entrada de l'esquiador a les pistes, el nivell de l'esquiador, i l'hora, minut i segons de sortida de l'estació. Com a màxim hi podran haver dos mil esquiadors a l'estació.

Cada pista està identificada per un nom i conté el nivell de dificultat donat pel tipus de pista. El tipus de pista pot ser, per ordre de dificultat, verd, blau, vermell o negre. El nivell de l'esquiador es correspon amb el tipus de pista de màxima dificultat que pot baixar. Per seguretat, un esquiador mai no pot canviar a una pista que sigui de més dificultat que el què indica el seu nivell. Com a màxim a una estació es poden obrir vuitanta pistes.

A l'etapa de disseny s'ha decidit crear quatre classes: **`Esquiador`** (per a mantenir totes les dades d'un sol esquiador), **`Pista`** (que manté les dades d'una pista), **`Estacio`** (per a guardar col.lecció de les pistes que formen l'estació, sent la primera pista d'aquesta col.leció la pista "Bosquet", i per a mantenir la col·lecció d'esquiadors que han passat per l'estació en un dia, ordenada per instant d'entrada de l'esquiador a l'estació) i **`Gestio`** (per a implementar el control principal de l'aplicació). S'adjunta un llistat per a veure part de la implementació de la classe `Gestio`.

Per facilitar la resolució dels problemes següents, considera que:

- Cada esquiador guarda la posició (o índex a la col.lecció de pistes) de la pista on està esquiant.
- La classe `Pista` ja està implementada i disposes dels mètodes `get` i `set` per a tots els seus atributs:

```
public class Pista {
  private String nom;
  private int    nivell;
}
```

- La classe `Hora` ja està implementada i disposes dels següents mètodes públics:

```
public class Hora {
  public Hora( int hora, int minuts, int segons);
  public boolean equals (Hora h);
}
```

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. (*1 punt*) Implementa la classe **`Esquiador`**. Suposa que tots els esquiadors entren a l'estació per la pista "Bosquet"que és verda i està a la posició 0 de la llista de Pistes de l'Estació. L'hora de sortida, per defecte, és 00:00:00.

   (a) Justifica els atributs que caracteritzen aquesta classe i els seus tipus.

   (b) Implementa només els següents mètodes:
   - el contructor de la classe que permeti passar les dades necessàries per a inicialitzar els atributs.
   - el mètode `public String toString()`, tenint en compte si encara està a l'estació o no. que permeti llistar les dades d'un esquiador.

Listing 1: Esquiador.java

```
public class Esquiador {
    private long forfait;      // enter que representa el numero de forfait:
                               // tambe pot ser un int o un String
    private Hora horaEntrada; // H, m, s de entrada
    private Hora horaSortida; // H, m, s de sortida. Si es 0,0,0 vol dir que l'esqui
                               // encara esta dins de l'estacio
    private int nivell;        // enter que indica el nivell de l'esquiador
    private int pistaActual;  // enter que es la posicio dins del Conjunt de Pistes.
                               // Indica a quina pista estÃ  esquiant, si no ha sorti

    public Esquiador() {
        pistaActual = 0;
        horaSortida = new Hora(0, 0, 0);
    }
    public Esquiador(long f, Hora horaEntrada, int nivell) {
        pistaActual = 0;
        horaSortida = new Hora(0, 0, 0);
        forfait = f;
        this.horaEntrada = horaEntrada;
        this.nivell = nivell;
    }

    public String toString() {
        String s;
        s = "Forfait␣numero␣"+forfait+ "␣de␣nivell␣" + nivell + "␣ha␣entrat␣a␣les␣"+
        if (!horaSortida.equals(new Hora(0,0,0))) {
            s = s + "␣i␣ha␣sortit␣a␣les␣"+horaSortida;
        } else {
            s = s + "␣esta␣a␣la␣pista␣" + pistaActual;
        }
        return s;
    }
}
```

2. (*1 punt*) En relació al mètode **obteDadesEsquiador**, contesta les següents preguntes:

   (a) Justifica a quina classe pertany. És un mètode de classe o és un mètode d'objecte? Per què?
   Pertany a la classe Esquiador, que és la classe que té accés a les dades d'un esquiador. Es un mètode d'objecte ja que depén de les dades de l'objecte instanciat en cada moment.

   (b) Completa el mètode `opcioAfegir(...)` amb les declaracions i sentències que calen per al bon funcionament del programa:

```
static void opcioAfegir(....) {
  .....
  esquiador.obteDadesEsquiador();
  estacio.afegeixEsquiador(esquiador);
}
```

```
static void opcioAfegir(Scanner sc, Estacio estacio) {
    Esquiador esquiador;

    esquiador = new Esquiador();
    esquiador.obteDadesEsquiador(sc);
    estacio.afegeixEsquiador(esquiador);
}
```

(c) Implementa el mètode **obteDadesEsquiador**.

```
public void obteDadesEsquiador(Scanner sc) {
    System.out.println("Numero␣de␣forfait");
    forfait = sc.nextInt();
    System.out.println("Hora␣d'entrada");
    h = new Hora(sc.nextInt(), sc.nextInt(), sc.nextInt());
    System.out.println("Nivell");
    nivell = sc.nextInt();
}
```

3. (*1 punt*) Implementa la classe **`Estacio`** amb els seus atributs i només implementa el seu constructor.

Listing 2: Estacio.java

```
public class Estacio {
    private Esquiador[] tauEsquiadors;
    private int numEsquiadors;
    private Pista[] conjuntPistes;
    private int numPistes;

    public Estacio() {
        tauEsquiadors = new Esquiador[Gestio.MAX_ESQUIADORS];
        numEsquiadors = 0;
        conjuntPistes = new Pista[Gestio.MAX_PISTES];
        numPistes = 1;
        conjuntPistes[0] = new Pista("Bosquet", 0);

    }

    public Estacio(int maxEsq, int maxPist) {
        tauEsquiadors = new Esquiador[maxEsq];
        numEsquiadors = 0;
        conjuntPistes = new Pista[maxPist];
        numPistes = 1;
        conjuntPistes[0] = new Pista("Bosquet", 0);
    }
}
```

4. (*2 punts*) Identifica les seqüències i esquemes algorísmics necessaris per resoldre els següents mètodes de la classe `Estacio` usats per a canviar de pista un esquiador (**opcioCanviPista**):

```
// idForfait identifica el forfait d'un esquiador
public Esquiador cercaEsquiador(int idForfait);

// esquiador és l'esquiador a qui se li canvia de pista
// idxPista és la posició de la nova pista a la taula de pistes
public boolean canviPista(Esquiador esquiador, int idxPista);
```

(a) Retornes un nou objecte en el mètode **cercaEsquiador**? Raona la teva resposta.
No, ja que sinó no quedaria reflectit el canvi de Pista en la col.lecció d'esquiadors de l'estació

(b) Implementa els dos mètodes.

```
public Esquiador cercaEsquiador(int idForfait) {
    // Cerca per tots els esquiadors de l'estacio
    // 1. Identificacio sequencia:
    // Primer element : i = 0;
    // Seg element: i++;
    // Fi seq: i<numEsquiadors
    // Esquema algorismic: cerca: tauEsquiadors[i].getForfait() == idForfait

    int i;
    boolean trobat;
    trobat = false;
    i= 0;
    while (i<numEsquiadors && !trobat) {
        if (tauEsquiadors[i].getForfait() == idForfait) trobat = true;
        else i++;
    }
    if (trobat) return tauEsquiadors[i];
    else return null;
}

public boolean canviPista(Esquiador esquiador, int idxPista) {
    boolean teProuNivell = false;
    if (esquiador!=null) {
      if (conjuntPistes[idxPista].getNivell()<= esquiador.getNivell()) {
        esquiador.setPista(idxPista);
        teProuNivell = true;
      }
    }
    return teProuNivell;
}
```

5. (*2.5 punts*) Es considera que un esquiador expert baixa la pista inicial "Bosquet" en menys de deu minuts. Es vol saber si en un moment ha entrat un expert. Per a això, donat un cert instant de temps es vol implementar el mètode **hiHaEsquiadorsExperts** a la classe `Estacio`. El mètode calcularà si algun esquiador que ha entrat a l'estació fa deu minuts de l'instant donat, ja ha sortit de la pista "Bosquet". Considera que disposes d'un mètode **`faDeuMinuts()`**, ja implementat a la classe `Gestio`, que retorna cert si la diferència entre dos instants de temps és inferior a deu minuts.

```
public static boolean faDeuMinuts(Hora h1, Hora h2);
```

Contesta les següents qüestions:

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

(b) Defineix la capçalera del mètode **hiHaEsquiadorsExperts** i implementa'l.

```
public boolean hihaEsquiadorsExperts(Hora instant) {
    // Cerca per tots els esquiadors de l'estacio
    // 1. Identificacio sequencia:
    // Primer element : i = 0;
    // Seg element: i++;
    // Fi seq: i<numEsquiadors
    // Esquema algorismic: cerca: faDeuMinuts (tauEsquiadors[i].getHoraEntrada()
    //                            &&  tauEsquiadors[i].getPista() != 0
    int i;
    boolean trobat;
    trobat = false;
    i= 0;
    while (i<numEsquiadors && !trobat) {
        if (Gestio.faDeuMinuts (tauEsquiadors[i].getHoraEntrada(), instant) &&
            tauEsquiadors[i].getPista()!=0)
                trobat = true;
        else i++;
    }
    return trobat;
    }
```

(c) Implementa el mètode **opcioHiHaExperts()** de la classe `Gestio`.

```
static public void opcioHihaEsquiadorsExperts(Scanner sc, Estacio estacio) {
    Hora instantActual;

    System.out.println("Entre␣l'instant␣actual");
    instantActual = new Hora(sc.nextInt(), sc.nextInt(), sc.nextInt());
    if (estacio.hihaEsquiadorsExperts(instantActual)) {
        System.out.println("Hi␣han␣esquiadors␣experts") ;
    } else {
        System.out.println("No␣hi␣han␣esquiadors␣experts") ;
    }
}
```

6. (*2.5 punts*) Es vol analitzar l'ocupació dels diferents tipus de pistes en un instant donat. Es vol donar una sortida de l'estil:
A l'hora 12:00:00, l'estat de l'estació és:
Pistes verdes: 200 esquiadors,
Pistes blaves: 210 esquiadors,
Pistes vermelles: 50 esquiadors,
Pistes negres: 10 esquiadors

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

(b) Defineix la capçalera del mètode **analitzaOcupacio** de la classe `Estacio` i implementa'l. Necessites alguna classe addicional?

```
public int[] analitzaOcupacio(Hora instantActual) {
    int [] contador =  {0,0,0,0};
    // Recorregut per tots els esquiadors de l'estacio mirant el nivell
    // de la pista on esta
    // 1. Identificacio sequencia:
    // Primer element : i = 0;
    // Seg element: i++;
    // Fi seq: i<numEsquiadors
    // Esquema algorismic: Recorregut
    for (int i=0; i<numEsquiadors; i++) {
      if (tauEsquiadors[i].getHoraEntrada().mesPetita(instantActual) &&
```

```
                instantActual.mesPetita(tauEsquiadors[i].getHoraSortida()))
            contador[conjuntPistes[tauEsquiadors[i].getPista()].getNivell()]++;
        }
        return contador;
    }
```

Necessito una taula addicional de 4 contadors, un per a cada tipus de pista (Verda, blava, vermell i negra. Suposo que a la classe Hora esta implementat el mètode mesPetita(Hora h) que dona cert si l'hora de l'objecte és abans que l'hora h.

(c) Implementa el mètode **opcioOcupacio()** de la classe `Gestio`.

```
public static void opcioOcupacio(Scanner sc, Estacio estacio) {
    Hora instantActual;

    System.out.println("Entra␣l'instant␣actual");
    instantActual = new Hora(sc.nextInt(), sc.nextInt(), sc.nextInt());
    int [] contadors;
    contadors = analitzaOcupacio(instantActual);
    System.out.println("A␣l'hora␣"+instantActual+ "␣l'estat␣de␣l'estaci\'o␣\'es:
    System.out.println("Pistes␣verdes:␣"+contadors[0]+"␣esquiadors");
    System.out.println("Pistes␣blaves:␣␣"+contadors[1]+"␣esquiadors");
    System.out.println("Pistes␣vermelles:␣␣"+contadors[2]+"␣esquiadors");
    System.out.println("Pistes␣negres:␣"+contadors[3]+"␣esquiadors");
}
```

Llistat de la classe principal de l'aplicació, amb els mètodes que ja teniu implementats: (podeu observar quins són els paràmetres d'entrada i els valors de sortida dels mètodes que heu d'implementar als problemes)

Listing 3: Gestio0.java

```
import java.util.Scanner;

public class Gestio0 {

  public  static final int AFEGIR       = 1;
  public  static final int CANVI_PISTA  = 2;
  public  static final int EXPERTS    = 3;
  public  static final int OCUPACIO   = 4;
  public  static final int SORTIDA    = 5;
  public  static final int SORTIR     = 0;
  public  static final int MAX_ESQUIADORS= 2000;
  public  static final int MAX_PISTES= 80;
  public  static final int VERD= 0;
  public  static final int BLAU= 1;
  public  static final int VERMELL= 2;
  public  static final int NEGRE = 3;

  public static void main (String [] args) {
      Scanner sc;
      Estacio1 estacio = new Estacio1(MAX_ESQUIADORS, MAX_PISTES);
      int     op;

      sc = new Scanner(System.in);
      op = pintaMenuPreguntaOpcio(sc);
      while (op!=SORTIR) {
          switch(op) {
          case AFEGIR: opcioAfegir(sc, estacio); break;
          case CANVI_PISTA: opcioCanviPista(sc, estacio); break;
          case EXPERTS:   opcioHiHaExperts(sc, estacio); break;
          case OCUPACIO:  opcioOcupacio(sc, estacio); break;
          case SORTIDA:   opcioSortidaEstacio(sc, estacio); break;
          default: break;
          }
          op = pintaMenuPreguntaOpcio(sc);
      }
  }

    static int pintaMenuPreguntaOpcio(Scanner sc) {
        System.out.println("Tria una opcio de les seguents ");
        System.out.println("1. Afegir Esquiador ");
        System.out.println("2. Canvi de Pista ");
        System.out.println("3. Hi ha experts? ");
        System.out.println("4. Ocupacio? ");
        System.out.println("5. Sortir de l'Estacio ");
        System.out.println("0. Acabar ");
        System.out.println("Quina opcio vols?");
        return (sc.nextInt());
    }

    static void opcioAfegir(Scanner sc, Estacio1 estacio) {
        Esquiador1 esquiador;

        esquiador = new Esquiador1();
        esquiador.obteDadesEsquiador(sc);
        estacio.afegeixEsquiador(esquiador);
    }

    static void opcioCanviPista(Scanner sc, Estacio1 estacio) {
        int idForfait;
```

```
        Esquiador1 esquiador;
        String novaPista;
        int idxPista;

        System.out.println ("Numero␣de␣forfait␣de␣l'esquiador?");
        idForfait = sc.nextInt();
        esquiador = estacio.cercaEsquiador(idForfait);

        System.out.println ("Nom␣de␣la␣nova␣pista?");
        novaPista = sc.next();
        idxPista = estacio.cercaPista(novaPista);

        if (idxPista!=-1) {
            if (estacio.canviPista(esquiador, idxPista)) {
                System.out.println("Esquiador␣amb␣forfait␣" + idForfait +"␣ha␣canviat␣a"
                                + "␣la␣pista␣" + novaPista);
            } else {
                System.out.println("Esquiador␣amb␣forfait␣" + idForfait +"␣no␣tÃ©" +
                                "␣el␣nivell␣suficient");
            }
        } else {
            System.out.println("No␣existeix␣la␣pista␣de␣nom␣"  + novaPista);
        }
    }

    static public void opcioHiHaExperts(Scanner sc, Estacio1 estacio) {
        Hora instantActual;

        System.out.println("Entra␣l'instant␣actual");
        instantActual = new Hora(sc.nextInt(), sc.nextInt(), sc.nextInt());
        if (estacio.hihaEsquiadorsExperts(instantActual)) {
            System.out.println("Hi␣han␣esquiadors␣experts") ;
        } else {
            System.out.println("No␣hi␣han␣esquiadors␣experts") ;
        }
    }

    public static void opcioOcupacio(Scanner sc, Estacio1 estacio) {
        Hora instantActual;

        System.out.println("Entra␣l'instant␣actual");
        instantActual = new Hora(sc.nextInt(), sc.nextInt(), sc.nextInt());
        int [] contadors;
        contadors = estacio.analitzaOcupacio();
        System.out.println("A␣l'hora␣"+instantActual+ "␣l'estat␣de␣l'estaci\'o␣\'es:");
        System.out.println("Pistes␣verdes:␣"+contadors[0]+"␣esquiadors");
        System.out.println("Pistes␣blaves:␣␣"+contadors[1]+"␣esquiadors");
        System.out.println("Pistes␣vermelles:␣␣"+contadors[2]+"␣esquiadors");
        System.out.println("Pistes␣negres:␣"+contadors[3]+"␣esquiadors");
    }
}
```