Segon Parcial d'IIP - ETSInf
Data: 15 de gener de 2016. Durada: 2:30 hores.

1. [6.5 punts] Es disposa de la classe `Termostat` que representa el controlador de temperatura d'un dispositiu tèrmic instal·lat en alguna zona d'un espai determinat (un habitatge, espai d'oficines, etc.). Cada termòstat es defineix sobre la base de quatre dades: el seu identificador (nom de la zona on se situa), el seu mode (`FRED` per a refrigeració, `CALOR` per a calefacció), la temperatura actual de la zona i la temperatura de confort que desitja l'usuari.

   Aquesta classe ja és coneguda i es mostra a continuació un resum de la seua documentació:

Field Summary

static int	CALOR Constant que representa el mode calefaccio d'un Termostat.
static int	FRED Constant que representa el mode refrigeracio d'un Termostat.
static int	T_IDEAL_CALOR Temperatura ideal en mode calefaccio d'un Termostat.
static int	T_IDEAL_FRED Temperatura ideal en mode refrigeracio d'un Termostat.

Constructor Summary

Termostat() Crea un Termostat per defecte en mode FRED, amb nom "zona d'estar", temperatura de confort T_IDEAL_FRED i com a temperatura actual un valor double aleatori en l'interval [20.0, 40.0[.
Termostat(int m, java.lang.String n, int tC, double tAct) Crea un Termostat amb les dades mode (m), nom (n), temperatura de confort (tC) i temperatura actual (tAct).

Method Summary

int	diferenciaAmbIdeal() Torna un enter que es: - Zero si la temperatura de confort es adequada al mode, es a dir, si es major o igual que la ideal en mode FRED o menor o igual en mode CALOR, - la diferencia en valor absolut entre les temperatures de confort i ideal, en altre cas.
java.lang.String	getNom() Torna el nom del Termostat.

   Es demana: implementar la classe `GestorTermostats` que representa els termòstats existents en un espai mitjançant les components (atributs i mètodes) que s'indiquen a continuació.

   Recordeu que les constants de la classe `Termostat` i de la classe `GestorTermostats` han d'utilitzar-se sempre que es requerisca.

   a) (0.5 punts) Atributs:

   - `MAX_TERMS`, una constant de classe (o estàtica) que representa el nombre màxim de termòstats que poden haver en un espai i que val 15.
   - `numTerms`, un enter en l'interval `[0..MAX_TERMS]` que representa el nombre de termòstats de l'espai a cada moment.
   - `terms`, un array de tipus base `Termostat`, de capacitat `MAX_TERMS`, per a emmagatzemar els termòstats que hi ha en l'espai a cada moment, disposats seqüencialment en posicions consecutives de l'array, des de la `0` fins a la `numTerms - 1` inclusivament.

- `noEficients`, enter que representa el nombre de termòstats que hi ha en l'espai a cada moment que no compleixen amb les normes d'eficiència, és a dir, aquells la temperatura de confort dels quals no és l'adequada per al seu mode de treball (tal com ho comprova el mètode `diferenciaAmbIdeal` de la classe `Termostat`).

b) (0.5 punts) Un constructor per defecte (sense paràmetres) que crea un gestor de termòstats buit, amb 0 termòstats.

c) (1 punt) Un mètode amb perfil:

```
private int termostatEnZona(String nomZona)
```

que, donat el nom `String nomZona` d'una zona de l'espai, retorna la posició de l'array on es troba el termòstat el nom del qual correspon a aquesta zona o -1 si no està.

d) (1.5 punts) Un mètode amb perfil:

```
public boolean instalar(Termostat t)
```

que si no hi ha cap termòstat amb el mateix nom que `t`, l'afegeix al gestor; si existeix un termòstat amb el mateix nom que `t` el reemplaça per `t`. El mètode retorna `true` si s'ha instal·lat o actualitzat amb èxit, o `false` en cas que no es puguen gestionar més termòstats. Noteu que s'ha d'actualitzar, si cal, l'atribut `noEficients`, i que s'ha d'usar el mètode privat `termostatEnZona` per a cercar el termòstat `t` pel seu nom.

e) (1.5 punts) Un mètode amb perfil:

```
public Termostat diferenciaMajor()
```

que retorna el `Termostat` amb major diferència en valor absolut entre les temperatures de confort i ideal, o `null` si no hi ha cap `Termostat`.

f) (1.5 punts) Un mètode amb perfil:

```
public Termostat[] termsNoEficients()
```

que retorna un array de `Termostat` amb els termòstats no eficients. La longitud d'aquest array haurà de ser igual al nombre de termòstats no eficients, o 0 si no n'hi ha cap.

Solució:

```
/**
 * Class GestorTermostats: representa un dispositiu que gestiona
 * tots els termostats d'un espai.
 *
 * @author Examen IIP
 * @version Segon Parcial - Curs 2015-2016
 */
public class GestorTermostats {
    /** Constant que representa el nombre maxim de
     *  termostats que poden haver en un espai. */
    public static final int MAX_TERMS = 15;
    // enter en [0..MAX_TERMS] que representa el nombre de
    // termostats que te l'espai en cada moment
    private int numTerms;
    // array d'objectes Termostat, de capacitat MAX_TERMS i
    // on els termostats s'emmagatzemen sequencialment, en
    // posicions consecutives des de la 0 fins la numTerms - 1
    private Termostat[] terms;
    // enter que representa el nombre de termostats que no compleixen
    // les recomanacions d'eficiencia energetica
    private int noEficients;
```

```
    /** Constructor per defecte que crea el gestor buit. */
    public GestorTermostats() {
        terms = new Termostat[MAX_TERMS];
        numTerms = 0;
        noEficients = 0;
    }

    /** Retorna la posicio de l'array en la que es troba el Termostat
     *  de nom nomZona o -1 si no esta.
     *  @param nomZona String, nom de la zona.
     *  @return int, posicio de l'array terms o -1 si no esta.
     */
    private int termostatEnZona(String nomZona) {
        int i = 0;
        while (i < numTerms && !terms[i].getNom().equals(nomZona)) { i++; }
        if (i < numTerms) { return i; }
        else { return -1; }
    }

    /** Si no hi ha cap Termostat amb el mateix nom que t, l'afegeix al gestor;
     *  si existeix un Termostat amb el mateix nom que t, el reemplaça per t.
     *  Torna true si s'ha instal.lat/actualitzat amb exit o false en cas de
     *  que no es puguen gestionar mes termostats.
     *  @param t Termostat, el termostat a instal.lar/actualitzar.
     *  @return boolean.
     */
    public boolean instalar(Termostat t) {
        boolean cap = true;
        int pos = termostatEnZona(t.getNom());
        if (pos != -1) {
            if (terms[pos].diferenciaAmbIdeal() != 0) { noEficients--; }
            terms[pos] = t;
        }
        else if (numTerms < MAX_TERMS) { terms[numTerms++] = t; }
        else { cap = false; }
        if (cap && t.diferenciaAmbIdeal() != 0) { noEficients++; }
        return cap;
    }

    /** Torna el Termostat amb major diferencia en valor absolut entre
     *  les temperatures de confort i ideal o null si no n'hi ha cap.
     *  @return Termostat.
     */
    public Termostat diferenciaMajor() {
        Termostat tMax = null;
        if (numTerms != 0) {
            tMax = terms[0];
            int difMax = tMax.diferenciaAmbIdeal();
            for (int i = 1; i < numTerms; i++) {
                int dif = terms[i].diferenciaAmbIdeal();
                if (dif > difMax) { tMax = terms[i]; difMax = dif; }
            }
        }
        return tMax;
    }

    /** Torna un array de Termostat amb els termostats no
     *  eficients. La longitud d'aquest array es el nombre
     *  de termostats no eficients, o 0 si no n'hi ha cap.
     *  @return Termostat[].
     */
    public Termostat[] termsNoEficients() {
        Termostat[] aux = new Termostat[noEficients];
        int k = 0;
        for (int i = 0; i < numTerms && k < noEficients; i++) {
            if (terms[i].diferenciaAmbIdeal() != 0) {
                aux[k++] = terms[i];
            }
        }
        return aux;
    }
}
```

2. 1.75 punts Es demana escriure un mètode Java estàtic que, donat un enter n ≥ 2, escriga en l'eixida una figura de n línies, amb dues diagonals que es junten en l'última línia, sobre un fons rectangular de `'-'`, de manera que:

   - En cada línia s'han d'escriure dues `'V'` separades per un nombre de `'-'` cada vegada menor,
   - en la primera línia la primera `'V'` apareix pegada al marge esquerre, i la segona en l'extrem dret de la figura.

   Exemple per a n=5:

```
V--------V
-V------V-
--V----V--
---V--V---
----VV----
```

Solució:

```
/** Escriu en l'eixida estandar una figura de n linies (n >= 2),
 *  formada per dues diagonals de 'V' que es junten en l'ultima
 *  linia, sobre un fons rectangular de '-'.
 */
public static void escriuV(int n) {
    // g1 es el nombre de guions que cal escriure
    // en cada linia davant de la primera 'V' i darrere de la segona;
    // g2 es el nombre de guions que cal escriure
    // en cada linia entre les dues 'V';
    int g1 = 0, g2 = 2 * n - 2;
    while (g1 < n) {
        for (int i = 1; i <= g1; i++) { System.out.print('-'); }
        System.out.print('V');
        for (int i = 1; i <= g2; i++) { System.out.print('-'); }
        System.out.print('V');
        for (int i = 1; i <= g1; i++) { System.out.print('-'); }
        System.out.println();
        g1++; g2 = g2 - 2;
    }
}
```

La següent solució alternativa té en compte que per a la línia `i`, el nombre de guions a escriure als extrems i en el centre es poden calcular com `i` i `2*(n-i)-2` respectivament.

```
public static void escriuV(int n) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < i; j++) { System.out.print('-'); }
        System.out.print('V');
        for (int j = 0; j < (2 * (n - i) - 2); j++) { System.out.print('-'); }
        System.out.print('V');
        for (int j = 0; j < i; j++) { System.out.print('-'); }
        System.out.println();
    }
}
```

3. [1.75 punts] Es demana escriure un mètode Java estàtic que, donat un array de `double`, comprove si les components d'índex parell apareixen ordenades ascendentment. Exemples:

Per a

0	1	2	3	4	5
1.5	0.0	**3.0**	-1.0	**3.5**	2.0

i

0	1	2	3	4	5	6
3.0	0.0	**4.5**	-1.0	**6.5**	2.0	**8.5**

ha de donar `true`.

Per a

0	1	2	3	4	5
1.5	0.0	**3.0**	-1.0	**1.5**	2.0

i

0	1	2	3	4	5	6
3.0	0.0	**1.0**	-1.0	**6.5**	2.0	**8.5**

ha de donar `false`.

Solució:

```
/** Comprova si les components d'index parell d'a
 *  estan ordenades ascendentment.
 */
public static boolean enOrdreParells(double[] a) {
    int i = 0;
    while (i < a.length - 2 && a[i] <= a[i + 2]) { i = i + 2; }
    return (i >= a.length - 2);
}
```