Segon Parcial d'IIP - ETSInf
Data: 9 de gener de 2017. Duració: 2h 30'

1. 6 punts Es disposa de la classe `Edifici` que representa un edifici de la UPV mitjançant dos tipus d'informació: l'associada a la seua construcció (coordenades GPS i codi d'identificació en un pla) i l'associada a l'ús que se li ha assignat (tipus d'ús i nom de l'entitat que el fa servir). Aquesta classe ja és coneguda i es mostra, a continuació, un resum de la seua documentació:

Field Summary

Fields

Modifier and Type	Field and Description
`static int`	**`DEPARTAMENT`** Constant que representa el tipus d'edifici departamental.
`static int`	**`ESCOLA`** Constant que representa el tipus d'edifici dedicat a docencia com aularis, escoles o laboratoris.
`static int`	**`SERVICIS`** Constant que representa el tipus d'edifici per a altres activitats com cafeteries, oficines, etc.

Constructor Summary

Constructors

Constructor and Description
`Edifici()` Crea un `Edifici` de codi `"1F"`, usat per l'entitat `"DSIC"`, de tipus departamental i en les coordenades `(39.4625, -0.3472)`.
`Edifici(java.lang.String c, java.lang.String e, int t, Punt p)` Crea un `Edifici` de codi `c`, usat per l'entitat `e`, de tipus `t` i en les coordenades donades pel `Punt p`.

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method and Description
`boolean`	**`equals(java.lang.Object o)`** Torna `true` si `o` es un `Edifici` amb el mateix codi i les mateixes coordenades que `this`; en cas contrari, torna `false`.
`java.lang.String`	**`getEntitat()`** Torna l'entitat que utilitza l'edifici `this`.
`int`	**`getTipus()`** Torna el tipus de l'edifici `this`.
`int`	**`mesPropDeRectorat(Edifici e)`** Donat un `Edifici e`, torna: `-1` si l'edifici `this` esta mes prop de rectorat que l'edifici `e`; `1` si `e` esta mes prop de rectorat que `this`; o `0` si tots dos edificis estan igual de prop de rectorat.

Es demana: implementar la classe `PlaVera` per representar els edificis del campus de Vera de la UPV mitjançant les components (atributs i mètodes) que s'indiquen a continuació.

Recorda que has d'usar les constants de les classes `Edifici` i `PlaVera` sempre que es requerisca.

a) (0.5 punts) Atributs, dels quals només és públic el primer:

- `MAX_EDIFS`, una constant de classe (o estàtica) que representa el numero màxim d'edificis que poden haver al pla i que val 50.
- `numEdifs`, un enter en l'interval `[0..MAX_EDIFS]` que representa el número d'edificis del pla en cada moment.
- `edifs`, un array de tipus base `Edifici`, de capacitat `MAX_EDIFS`, per emmagatzemar els edificis del pla en cada moment, disposats en posicions consecutives de l'array, des de la `0` fins la `numEdifs - 1` incloses, **ordenats ascendentment per la seua proximitat a rectorat**, sent `edifs[0]` rectorat, `edifs[1]` el més proper i `edifs[numEdifs - 1]` el més llunyà. Si dos edificis estan **igual de prop** de rectorat, ocuparan en l'array dues posicions consecutives `i` i `i + 1`, $1 \leq$ `i` $<$ `numEdifs - 1`, sent `edifs[i + 1]` un edifici afegit a l'array **amb posterioritat** a `edifs[i]`.
- `numEscoles`, un enter no negatiu que representa el número d'edificis docents que hi ha al pla en cada moment.

b) (1 punt) Un constructor per defecte (sense paràmetres) que crea un objecte `PlaVera` amb 1 únic edifici amb les següents característiques: un edifici de servicis usat per l'entitat `"Rectorat"`, amb codi `"3A"` i coordenades `(39.4823, -0.3457)`.

c) (1.5 punts) Un mètode amb perfil:

```
private int posicioDe(Edifici e)
```

que, donat un `Edifici e`, torna la posició del primer edifici del array (d'índex menor) més llunyà a rectorat que `e`, o `numEdifs` si no hi ha cap edifici més llunyà a rectorat que `e`.

Nota que has d'usar el mètode `mesPropDeRectorat(Edifici)` de la classe `Edifici`.

d) (1.5 punts) Un mètode amb perfil:

```
public boolean afegir(Edifici e)
```

que, donat un `Edifici e` que **no està en el pla**, l'afegeix, si cap, de manera **ordenada** segons la seua proximitat a rectorat, actualitzant els atributos `numEdifs` i, si procedeix, `numEscoles`. El mètode torna `true` si s'ha afegit amb èxit, o `false` si no caben més edificis en el pla.

Nota que has d'usar el mètode privat `posicioDe(Edifici)` per saber la posició de l'array `edifs` en la qual situar l'edifici `e`. Una vegada trobada aquesta posició, cal fer-li un buit a `e` en l'array. Per a això, has d'usar un mètode privat, **ja implementat**, amb el següent perfil:

```
private void desplazarDreta(int ini, int fi)
```

que desplaça una posició cap a la dreta els elements de l'array `edifs` des de la posició `ini` a la posició `fi` incloses ($0 \leq$ `ini` $\leq$ `fi` $\leq$ `numEdifs - 1` $<$ `edifs.length - 1`). Nota que, per precondició, si `ini` $>$ `fi`, no fa cap desplaçament.

e) (1.5 puntos) Un mètode amb perfil:

```
public Edifici[] filtrarTipusEscola()
```

que torna un array d'`Edifici` amb els edificis docents. La longitud d'aquest array serà igual al número d'edificis de tipus docent, o 0 si no hi ha cap edifici d'aquest tipus en el pla.

Solució:

```
public class PlaVera {
    public static final int MAX_EDIFS = 50;
    private int numEdifs;
    private Edifici[] edifs;
    private int numEscoles;

    public PlaVera() {
        edifs = new Edifici[MAX_EDIFS];
        edifs[0] = new Edifici("3A", "Rectorat",
            Edifici.SERVICIS, new Punt(39.4823, -0.3457));
        numEdifs = 1;
        numEscoles = 0;
    }

    private int posicioDe(Edifici e) {
        int i = 1;
        while (i < numEdifs && edifs[i].mesPropDeRectorat(e) <= 0) { i++; }
        return i;
    }

    /** Precondicio: 0 <= ini <= fi <= numEdifs - 1 < edifs.length - 1 */
    private void desplazarDreta(int ini, int fi) {
        for (int pos = fi + 1; pos > ini; pos--) {
            edifs[pos] = edifs[pos - 1];
        }
    }

    /** Precondicio: e no esta en el pla */
    public boolean afegir(Edifici e) {
        boolean res = false;
        if (numEdifs != MAX_EDIFS) {
            int pos = posicioDe(e);
            desplazarDreta(pos, numEdifs - 1);
            edifs[pos] = e;
            numEdifs++;
            if (e.getTipus() == Edifici.ESCOLA) { numEscoles++; }
            res = true;
        }
        return res;
    }
```

```
    public Edifici[] filtrarTipusEscola() {
        Edifici[] aux = new Edifici[numEscoles];
        int k = 0;
        for (int i = 1; i < numEdifs && k < numEscoles; i++) {
            if (edifs[i].getTipus() == Edifici.ESCOLA) {
                aux[k] = edifs[i];
                k++;
            }
        }
        return aux;
    }
}
```

2. 2 punts Siga un enter `n` $\geq$ 2. **Es demana:** implementar un mètode estàtic que, per a tots els enters entre 2 i `n` inclusivament, torne un `String` amb la llista dels seus divisors propis. Recorda que els *divisors propis* d'un enter són tots els seus divisors excepte ell mateix i la unitat. Per exemple, per a `n = 18`, el mètode ha de tornar el següent `String`:

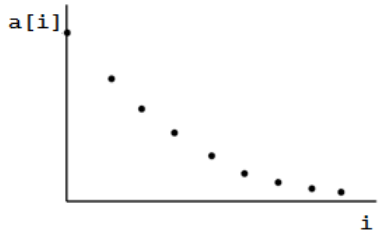
```
Divisors propis de 2:
Divisors propis de 3:
Divisors propis de 4: 2
Divisors propis de 5:
Divisors propis de 6: 2 3
Divisors propis de 7:
Divisors propis de 8: 2 4
Divisors propis de 9: 3
Divisors propis de 10: 2 5
Divisors propis de 11:
Divisors propis de 12: 2 3 4 6
Divisors propis de 13:
Divisors propis de 14: 2 7
Divisors propis de 15: 3 5
Divisors propis de 16: 2 4 8
Divisors propis de 17:
Divisors propis de 18: 2 3 6 9
```

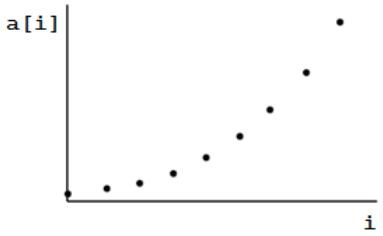
Solució:

```
/** Precondicio: n >= 2 */
public static String divPropis(int n) {
    String res = "";
    for (int i = 2; i <= n; i++) {
        res += "Divisors propis de " + i + ": ";
        for (int j = 2; j <= i / 2; j++) {
            if (i % j == 0) { res += j + " "; }
        }
        res += "\n";
    }
    return res;
}
```

3. 2 punts Siga un array `a` de reals i longitud `n` $\geq$ 2, tal que les seues components s'ajusten al perfil d'una corba còncava, és a dir, hi ha un mínim en una certa posició `k`, `0` $\leq$ `k` $<$ `n` (açò és, els valors en `a[0..k]` són estrictament decreixents i els valors en `a[k..n - 1]` són estrictament creixents); el mínim es pot trobar en un dels extrems de l'array. **Es demana:** implementar un mètode estàtic que, donat l'array,torne la posició del mínim. Per exemple, per als arrays de les següents figures, el mètode hauria de tornar 8, 5 i 0, respectivament.

Solució:






```
/** Precondicio: Les components d'a, a.length >= 2,
 *  s'ajusten al perfil d'una corba concava.
 */
public static int minimConcava(double[] a) {
    int i = 0;
    while (i < a.length - 1 && a[i] > a[i + 1]) { i++; }
    return i;
}
```