Segon Parcial d'IIP - ETSInf. Recuperació
Data: 24 de gener de 2017. Duració: 2h 30'

1. 6 punts Es disposa de la classe `Parada` que representa una parada d'un tren al llarg del seu trajecte. La informació de la parada ve donada pel nom de la ciutat, el moment del dia en què es fa la parada i el tipus de la mateixa. La parada pot ser de tipus estació, baixador o apartador (un apartador no permet la incorporació de viatgers).

   Es mostra, a continuació, un resum de la seua documentació, amb les seues constants i un extracte dels seus mètodes públics:

Field Summary

static int	APARTADOR
static int	BAIXADOR
static int	ESTACIO

Constructor Summary

Parada(java.lang.String c, iipUtil.Instant h, int t) Crea una Parada en la ciutat c a l'hora h i de tipus t.

Method Summary

. . .	
java.lang.String	getCiutat() Torna la ciutat de la parada this.
iipUtil.Instant	getHoraParada() Torna l'hora de parada de this.
int	getTipus() Torna el tipus de la parada this: ESTACIO, BAIXADOR o APARTADOR.

   Es demana: implementar la classe `RecorregutTren` per representar la seqüència de parades que realitza un tren entre un origen i un destí. Els atributs i mètodes de la classe a implementar són els que s'indiquen a continuació.

   a) (0.5 punts) Atributs, dels quals només és públic el primer:
      - `MAX_PARADES`, una constant de classe (o estàtica) que representa el número màxim de parades que pot haver en el recorregut i que val `25`. Has d'utilitzar aquesta constant i les de la classe `Parada` sempre que es requerisca.
      - `numParades`, un enter en l'interval `[0..MAX_PARADES]` que representa el número de parades que apareixen incloses en el recorregut en cada moment.
      - `trajecte`, un array de tipus base `Parada`, de capacitat `MAX_PARADES`, per tal d'emmagatzemar les parades que apareixen en el recorregut, disposades en posicions consecutives de l'array des de la `0` fins la `numParades - 1` inclusivament, **ordenades cronològicament per l'hora de parada**, sent `trajecte[0]` la parada d'origen i `trajecte[numParades - 1]` la parada de destí.
      - `numPujades`, un enter no negatiu que representa el número de parades que hi ha al recorregut i que permeten la incorporació de viatgers, és a dir, les parades que són estacions o baixadors.

   b) (1 punt) Un constructor que, donada una `Parada origen` i un altra diferent `desti`, sent l'hora de parada de `desti` posterior a la d'`origen`, crea un objecte `RecorregutTren` el trajecte del qual consta inicialment d'aquest parell de parades.

   c) (1.5 punts) Un mètode amb perfil:

```
private int posicio(Instant hP)
```

   que torna `-1` si l'instant `hP` és anterior al de l'origen o posterior al del destí, o ja existeix en `trajecte` una parada en el mateix instant; en un altre cas, torna l'índex de la primera parada de `trajecte` l'instant de la qual és posterior a `hP`.

   Recorda que per a comparar dos instants `t1`, `t2`, pots usar el mètode `compareTo` de la seua classe, de manera que `t1.compareTo(t2)` és < 0, 0 o > 0 si `t1` és anterior, igual o posterior a `t2`, respectivament.

d) (1.5 punts) Un mètode amb perfil:

```
public boolean incloure(Parada p)
```

per tal d'incloure `p` en el recorregut. Si l'hora de la parada `p` no coincideix amb la de cap altra parada del recorregut, és posterior a la de l'origen i anterior a la del destí, aleshores s'inclou `p` ordenadament en el recorregut i es torna `true`. Si `p` no es pot incloure per raons d'horari, o perquè s'excedeix el màxim número de parades, es torna `false`.

Nota que, en cas de que `p` es puga incloure, has d'usar el mètode privat `posicio` per tal de descobrir on situar `p` en l'array `trajecte`. Una vegada trobada aquesta posició, cal fer-li un forat a `p` en l'array. Per a això, has de fer servir 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 `trajecte[ini..fi]`, sent $0 \leq$ `ini` $\leq$ `fi` $\leq$ `numParades - 1` $<$ `trajecte.length - 1`.

e) (1.5 punts) Un mètode amb perfil:

```
public Parada[] pujades()
```

que torna un array amb les parades del recorregut que permeten la pujada de viatgers, és a dir, són estacions o baixadors. La longitud d'aquest array serà igual al número de parades en les que poden pujar viatgers, o 0 si no hi hagués cap.

Solució:

```
import iipUtil.Instant;
public class RecorregutTren {
    public static final int MAX_PARADES = 25;
    private Parada[] trajecte;
    private int numParades;
    private int numPujades;

    /** Precondicio: origen i desti son parades distintes, sent
     *  l'hora de parada de desti posterior a la d'origen */
    public RecorregutTren(Parada origen, Parada desti) {
        trajecte = new Parada[MAX_PARADES];
        trajecte[0] = origen;
        trajecte[1] = desti;
        numParades = 2;
        if (origen.getTipus() != Parada.APARTADOR) { numPujades++; }
        if (desti.getTipus() != Parada.APARTADOR) { numPujades++; }
    }

    private int posicio(Instant hP) {
        if (trajecte[0].getHoraParada().compareTo(hP) >= 0
            || trajecte[numParades - 1].getHoraParada().compareTo(hP) <= 0) {
            return -1;
        }
        int i = 1;
        while (i < numParades - 1 && trajecte[i].getHoraParada().compareTo(hP) < 0) {
            i++;
        }
        if (trajecte[i].getHoraParada().compareTo(hP) == 0) { return -1; }
        else { return i; }
    }

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

```
    public boolean incloure(Parada p) {
        if (numParades == MAX_PARADES) { return false; }
        int pos = posicio(p.getHoraParada());
        if (pos == -1) { return false; }
        desplazarDreta(pos, numParades - 1);
        trajecte[pos] = p;
        numParades++;
        if (p.getTipus() != Parada.APARTADOR) { numPujades++; }
        return true;
    }

    public Parada[] pujades() {
        Parada[] result = new Parada[numPujades];
        int j = 0;
        for (int i = 0; i < numParades && j < numPujades; i++) {
            if (trajecte[i].getTipus() != Parada.APARTADOR) {
                result[j] = trajecte[i];
                j++;
            }
        }
        return result;
    }
}
```

2. 2 punts Siga un enter `a` $>$ `1`. **Es demana:** implementar un mètode estàtic que, usant `'*'`, mostre per pantalla una figura composada per un triangle rectangle isòsceles d'altura `a` i la seua imatge especular, encarats per la hipotenusa i amb les seues bases separades per un espai en blanc. Per exemple, per a `a = 4`, el mètode ha de produir la següent figura:

```
*       *
**     **
***   ***
**** ****
```

Solució:

```
/** Precondicio: a > 1 */
public static void dibuixar(int a) {
    int b = a * 2 + 1; // base de la figura a dibuixar
    for (int i = 1; i <= a; i++) {
        for (int j = 1; j <= i; j++) { System.out.print('*'); }
        int blancs = b - i;
        for (int j = i; j < blancs; j++) { System.out.print(' '); }
        for (int j = 1; j <= i; j++) { System.out.print('*'); }
        System.out.println();
    }
}
```

3. 2 punts **Es demana:** implementar un mètode estàtic que, donats dos arrays de caràcters `a` i `b`, tots dos sense repetits, torne el nombre de caràcters comuns. Per exemple, si `a` és {`'C','T','A','G'`} i `b` és {`'T','U','C'`}, el mètode ha de tornar 2.

Solució:

```
/** Precondicio: a sense repetits, b sense repetits */
public static int numCharComuns(char[] a, char[] b) {
    int comuns = 0;
    for (int i = 0; i < a.length; i++) {
        int j = 0;
        while (j < b.length && a[i] != b[j]) { j++; }
        if (j < b.length) { comuns++; }
    }
    return comuns;
}
```