Recuperació Segon Parcial de IIP (ETSInf)

19 de gener de 2018. Duració: 2 hores i 30 minuts

Nota: Aquest examen s'avalua sobre 10 punts, però el seu pes específic a la nota final de IIP és de **3,6 punts**

NOM:	GRUP:

1. 6 punts Es disposa de la classe `Cita` per a representar cites en l'agenda d'una clínica dental. Cada cita correspon a una intervenció a practicar en la clínica. La franja horària d'obertura de la clínica és de 8 del matí a 8 de la vesprada sense interrupció. Es programen tres tipus d'intervencions: extracció de peça, neteja bucal i revisió d'ortodòncia. La durada estimada de cada intervenció és de 60 minuts per a les extraccions, 30 minuts per a les neteges bucals i de 20 minuts per a les revisions d'ortodòncia. Per a evitar que la clínica tanque més tard de les 20h, l'hora d'inici de les cites podrà ser qualsevol entre les 08:00h i les 19:00h. Es suposa que la classe `Cita` està implementada i llesta per ser utilitzada. A continuació es mostra un resum de la seua documentació, junt amb la de la classe `Instant`:

Classe `Cita`:

Fields

Modifier and Type	Field and Description
`static int`	`EXTRACCIO` Constant per a codificar el tipus d'intervenció extracció de peça, el seu valor és 0.
`static int`	`NETEJA` Constant per a codificar el tipus d'intervenció neteja bucal, el seu valor és 1.
`static int`	`ORTODONCIA` Constant per a codificar el tipus d'intervenció revisió d'ortodòncia, el seu valor és 2.

Constructors

Constructor and Description
`Cita(int tipus, java.lang.String nomPacient, int hora, int minuts)` Crea una cita del tipus d'intervenció `tipus`, amb el nom del pacient `nomPacient`, per a l'hora d'inici indicada pels paràmeters `hora` y `minuts`.

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

Modifier and Type	Method and Description
`int`	`compareTo(Cita altra)` Torna `> 0` si l'hora prevista d'inici de `this` és posterior a `altra`, `< 0` si és anterior, i `0` quan ambdues hores d'inici coincideixen.
`int`	`getDurada()` Torna la duració prevista en minuts segons el tipus d'intervenció.
`Instant`	`getInici()` Torna l'instant d'inici previst de la intervenció.
`java.lang.String`	`getNom()` Torn el nom del pacient de la cita en curs.
`int`	`getTipus()` Torna el tipus d'intervenció.

Classe `Instant`:

Constructors

Constructor and Description
`Instant(int h, int m)` Crea un `Instant` amb `h` hores i `m` minuts.

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

Modifier and Type	Method and Description
`int`	`aMinuts()` Torna el número de minuts transcorreguts des de les 00:00 fins l'`Instant` en curs.

Es demana: Implementar la classe tipus de dades `Agenda` per a representar la programació diària d'una clínica dental en un dia determinat. En aquesta classe s'utilitzaran els següents atributs i mètodes:

a) (0,5 punts) Atributs:

- `MAX_CITES`: atribut de classe públic constant de tipus enter que indica el màxim d'intervencions que poden ser programades en un dia. Aquest màxim és $34 = (19 - 8) * 3 + 1$, tenint en compte que la durada mínima prevista d'una intervenció és de 20 minuts, caben tres per hora i, per tant, caben 33 des de les 8h a les 19h. Falta afegir la darrera del dia que pot programar-se a les 19h per si és una extracció que té una durada de 60 minuts.
- `numCites`: atribut d'instància privat de tipus enter en l'interval `[0..MAX_CITES]` que indica el nombre d'intervencions planificades en la agenda en un moment donat.
- `cites`: atribut d'instància privat, un array de tipus base `Cita` per emmagatzemar les cites d'un dia; les cites s'han de disposar en posicions contigües de l'array, des de la 0 fins la `numCites - 1` ambdues incloses, i ordenades per ordre d'hora prevista d'inici, sense que hi haja superposició temporal entre elles, és a dir, que una cita no acabe més tard de l'hora d'inici de la següent cita en l'array.
- `numOrtodoncies`: atribut d'instància privat de tipus enter que indica el nombre de revisions d'ortodòncia planificades.

b) (0,5 punts) Un constructor per defecte (sense paràmetres) que crea l'array i inicialitza a 0 el nombre de cites planificades i el nombre d'ortodòncies.

c) (2,0 punts) Un mètode amb capçalera o perfil:

```
public boolean inserir(Cita c)
```

que intenta afegir la cita `c` en la agenda, és a dir, intenta inserir la cita en l'array `cites`. Per a simplificar aquest mètode, s'assumirà com a precondició que la nova cita `c` no es superposa temporalment amb cap de les ja existents en l'array. El mètode ha de comprovar si queda espai en l'array i que la cita té una hora d'inici dins de la franja permesa. Assumint que les cites prèviament existents en l'array estan ordenades temporalment, la nova cita ha d'inserir-se en la posició que li corresponga, per tant, si cal, hauran de desplaçar-se cap a la dreta les cites ja existents que tinguen una hora d'inici posterior a la nova cita. El mètode `inserir` ha de tornar `true` quan ha sigut possible inserir la nova cita, i `false` en cas contrari.

d) (1,0 punts) Mètode amb capçalera o perfil:

```
public int getNombreExtraccions()
```

que torna un enter indicant el nombre d'intervencions del tipus extracció de peça planificades per al dia.

e) (2,0 punts) Un mètode amb capçalera o perfil:

```
public String[] getNomsOrtodoncia()
```

que torna un array de tipus base `String` amb els noms dels pacients per a revisió d'ortodòncia. La longitud d'aquest array ha de ser igual al nombre de cites planificades per a revisió d'ortondòncia, o 0 si no hi ha cap.

Solució:

```
/**
 * Classe <code>Agenda</code> per a gestionar les cites d'una clínica
 * dental. Les cites es mantenen ordenades per hora estimada d'inici.
 * La franja horària d'atenció al públic va de 8 del matí a 8
 * de la vesprada. La darrera cita que pot assignar-se cada dia és
 * a les 19h, per tal d'assegurar que es tanca a les 20h.
 *
 * Hi ha tres tipus de cita, que són: neteja bucal, extracció de
 * peça i ortodòncia. Cada cita tindrà una durada estimada segons
 * el tipus de cita, 20 minuts per a revisions d'ortodòncia,
 * 30 minuts per a neteges bucals i 60 minuts per a extraccions.
 *
 *  @author IIP
 *  @version Curs 2017-18
 */
public class Agenda {
    /** Constant per a indicar el màxim nombre de cites per dia. */
    public static final int MAX_CITES = (19 - 8) * 3 + 1;

    /** Array per a emmagatzemar les cites planificades. */
    private Cita[] cites;
    /** Comptador de cites emmagatzemades a l'array cites. */
    private int numCites;
```

```
    /** Comptador de cites emmagatzemades a l'array cites
        que són del tipus Cita.ORTODONCIA. */
    private int numOrtodoncies;

    /**
     * Crea una agenda per a gestionar les possibles intervencions
     * en una clínica dental per a un mateix dia.
     * A l'inici l'agenda estarà buida, no contindrà cap cita.
     * Les cites aniran afegint-se progressivament.
     */
    public Agenda() {
        numCites = 0;
        numOrtodoncies = 0;
        cites = new Cita[MAX_CITES];
    }

    /**
     * Insereix una nova cita a l'agenda situant-la en la posició que
     * li correspon dins de l'array per tal d'assegurar que les cites
     * queden ordenades en ordre cronològic.
     * Precondició: la cita <code>c</code> no solapa amb cap de les
     * existents en l'array <code>cites</code>.
     */
    public boolean inserir(Cita c) {
        // Si no queda espai no s'insereix i es torna false.
        if (numCites == cites.length) { return false; }

        // Si l'hora d'inici no està dins l'horari d'obertura
        // es torna false.
        if (c.getInici().aMinuts() < 8 * 60
            || c.getInici().aMinuts() > 19 * 60) { return false; }

        // Cerca la posició que li correspon a la nova cita.
        int pos = 0;
        while (pos < numCites  &&  c.compareTo(cites[pos]) > 0) { pos++; }

        // Es desplacen una posició cap a la dreta totes les cites que
        // cronològicament són posteriors a la nova cita, és a dir,
        // aquelles que estan en el subarray [pos, numcites - 1]
        for (int i = numCites; i > pos; --i) { cites[i] = cites[i - 1]; }

        // Es col.loca la nova cita en la posició que li correspon.
        cites[pos] = c;

        // S'incrementa el nombre de cites emmagatzemades en l'agenda.
        ++numCites;

        // S'incrementa el nombre de cites emmagatzemades del tipus
        // Cita.ORTODONCIA si pertoca.
        if (c.getTipus() == Cita.ORTODONCIA) { ++numOrtodoncies; }

        return true;
    }

    /** Torna el nombre d'intervecions del tipus extracció. */
    public int getNombreExtraccions() {
        int comptador = 0;
        for (int i = 0; i < numCites; i++) {
            if (cites[i].getTipus() == Cita.EXTRACCIO) { ++comptador; }
        }
        return comptador;
    }

    /**
     * Torna un array amb els noms dels pacients que tenen
     * revisió d'ortodòncia.
     */
    public String[] getNomsOrtodoncia() {
        String[] noms = new String[numOrtodoncies];
        int k = 0;
        for (int i = 0; i < numCites && k < numOrtodoncies; i++) {
            if (cites[i].getTipus() == Cita.ORTODONCIA) {
                noms[k++] = cites[i].getNom();
            }
        }
        return noms;
    }
}
```

2. [2 punts] **Es demana:** implementar un mètode estàtic què, donat un nombre enter `n` $\geq$ `3`, escriga en l'eixida estàndard una figura de `n` línies. En cada línia s'han d'escriure tres caràcters `'N'` amb la separació adequada per tal que semble la lletra N majúscula. Per exemple, per a `n = 5` s'ha d'escriure:

```
NN    N
N N   N
N  N  N
N   N N
N    NN
```

Solució:

```
/** Precondició: n >= 3 */
public static void lletra(int n) {
    for (int i = 1; i <= n; i++) {
        System.out.print('N');
        for (int j = 1; j < i; j++) {
            System.out.print(' ');
        }
        System.out.print('N');
        for (int j = 1; j <= n - i; j++) {
            System.out.print(' ');
        }
        System.out.println('N');
    }
}
```

3. [2 punts] **Es demana:** implementar un mètode estàtic què, donat un array de nombres enters, determine si és **Par Dominant**, és a dir, si la suma del valor absolut de cadascun dels elements que ocupen posicions parelles, és major que tots i cadascun dels elements, en valor absolut, que ocupen posicions senars.

Solució:

```
public static boolean parDominant(int[] v) {
    boolean continuar = true;
    int suma = 0, j = 1;
    for (int i = 0; i < v.length; i = i + 2) {
        suma = suma + Math.abs(v[i]);
    }
    while (j < v.length && continuar) {
        if (Math.abs(v[j]) >= suma) { continuar = false; }
        else { j = j + 2; }
    }
    return continuar;
}
```