Segon Parcial d'IIP (ETSInf). Recuperació
18 de gener de 2019. Durada: 2 hores i 30 minuts

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

NOM:	GRUP:

1. 6 punts Es té un sistema de distribució d'internet i TV en una localitat, i es desitja estudiar les fallades o avaries que es produeixen en el servei al llarg d'un dia. Per a això es disposa de la classe `Failure`, els objectes de la qual representen la notificació d'una avaria i que conté l'identificador de la notificació, i l'instant del dia en què es va produir la fallada. La classe posseeix els següents mètodes:

Class Failure

Constructor Summary

Constructors

Constructor and Description
`Failure(int id, TimeInstant ins)` Crea l'objecte a partir de l'identificador id i de l'instant ins.

Method Summary

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

Modifier and Type	Method and Description
`int`	`getIdent()` Torna l'identificador del Failure.
`TimeInstant`	`getInstant()` Torna l'instant del Failure.

Els instants són de la classe `TimeInstant`, ja coneguda, alguns dels mètodes de la qual són:

Class TimeInstant

Method Summary

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

Modifier and Type	Method and Description
`int`	`getHours()` Torna les hores del TimeInstant.
`int`	`getMinutes()` Torna els minuts del TimeInstant.
`int`	`toMinutes()` Torna el nombre de minuts transcorreguts des de les 00:00 fins l'instant representat per l'objecte en curs.

Es desitja desenvolupar una classe Java que permeta estudiar la distribució de les avaries notificades al llarg d'un dia.

Es demana: Implementar la classe tipus de dades `FailureReport` que representa el conjunt d'avaries produïdes en la localitat al llarg d'un dia, usant els següents atributs i mètodes:

a) (0,5 punts) Atributs:

- `MAX`: atribut de classe públic constant de tipus enter, que indica el màxim nombre d'avaries que poden guardar-se, sent 100 en aquest cas.
- `reported`: atribut d'instància privat de tipus array d'objectes de la classe `Failure` i capacitat `MAX`, per a emmagatzemar les avaries que es van notificant. Cada `Failure` s'emmagatzema en posicions consecutives de l'array, des de la 0 fins a `nFailures - 1`. Un nou `Failure` sempre es disposa en l'array a continuació de l'últim prèviament guardat (no té per què hi haver cap ordenació de cap tipus entre els elements de l'array).
- `nFailures`: atribut d'instància privat de tipus enter que indica el nombre de `Failures` que es porten emmagatzemats.

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

c) (0,75 punts) Un mètode amb perfil:

`public boolean add(Failure f)`

que afegeix `f` al `FailureReport` i torna un `boolean` que indica si l'operació s'ha completat amb èxit. S'ha de situar `f` en la primera posició lliure de `reported`, a la dreta de tots les components ocupades de l'array, sempre que hi haja espai en l'array, en aquest cas es torna `true`. Si l'array estiguera ocupat per complet, no s'afegeix `f`, i es torna `false`.

d) (1,25 punts) Un mètode amb perfil:

`public Failure search(int hour, int iniM, int finM)`

que cerca i torna el primer `Failure` del que l'instant de fallada es corresponga a una hora `hour`, i uns minuts entre `iniM` i `finM`, tots dos inclusivament. Com a precondició se suposa que `0 <= hour <= 23` i `0 <= iniM <= finM <= 59`. Si no en troba cap, torna `null`.

e) (1,5 punts) Un mètode amb perfil:

`public TimeInstant closest(TimeInstant t)`

que torna, de totes les fallades guardades en el `FailureReport`, l'instant més proper a `t` en el qual s'ha produït alguna fallada, és a dir, aquell que difereix un mínim en minuts respecte a `t`. Com a precondició se suposarà que en el `FailureReport` hi ha almenys un `Failure`.

f) (1,5 punts) Un mètode amb perfil:

`public int[] histogram()`

que torna la distribució de fallades al llarg de les 24 hores d'un dia, en forma d'un array d'`int` en el qual la component 0 conté el nombre de fallades l'instant de les quals es troba entre les 00:00 i les 00:59 inclusivament, la component 1 el nombre de fallades entre les 01:00 i les 01:59, ... , i l'última component el nombre de fallades entre les 23:00 i les 23:59.

Solució:

```
public class FailureReport {
    public static final int MAX = 100;
    private Failure[] reported;
    private int nFailures;

    public FailureReport() {
        reported = new Failure[MAX];
        nFailures = 0;
    }

    public boolean add(Failure f) {
        if (nFailures == MAX) { return false; }
        reported[nFailures] = f;
        nFailures++;
        return true;
    }

    /** Precondició: 0 <= hour <= 23, 0 <= iniM <= finM <= 59. */
    public Failure search(int hour, int iniM, int finM) {
        int i = 0; boolean found = false;
        while (i < nFailures && !found) {
            TimeInstant t = reported[i].getInstant();
            int h = t.getHours(), m = t.getMinutes();
            if (h == hour && iniM <= m && m <= finM) { found = true; }
            else { i++; }
        }
        if (found) { return reported[i]; }
        else { return null; }
    }

    /** Precondició: En this hi ha almenys un Failure. */
    public TimeInstant closest(TimeInstant t) {
        TimeInstant tI = reported[0].getInstant(), minT = tI;
        int tToMinutes = t.toMinutes();
        int min = Math.abs(tI.toMinutes() - tToMinutes);
```

```
        for (int i = 1; i < nFailures; i++) {
            tI = reported[i].getInstant();
            int dif = Math.abs(tI.toMinutes() - tToMinutes);
            if (dif < min) { min = dif; minT = tI; }
        }
        return minT;
    }

    public int[] histogram() {
        int[] result = new int[24];
        for (int i = 0; i < nFailures; i++) {
            int h = reported[i].getInstant().getHours();
            result[h]++;
        }
        return result;
    }
}
```

2. 2 punts Donat un enter $n \geq 0$, es desitja calcular l'invertit de n, és a dir, un altre enter que continga les mateixes xifres que n però en ordre invers. Per a això, s'escriuran un parell de mètodes que se suposaran en la mateixa classe, de manera que un puga usar a l'altre en els seus càlculs.

   Es demana:

   1. (1 punt) Realitzar un mètode estàtic que calcule el nombre de xifres d'un enter donat ≥ 0. Per exemple, per a 2347 el mètode ha de tornar 4, per a 8 ha de retornar 1, per a 0 ha de tornar 1.
   2. (1 punt) Usant el mètode anterior, realitzar un mètode estàtic que calcule l'invertit d'un enter donat ≥ 0. Per exemple, per al 2347 haurà de calcular el 7432.

      Noteu en l'exemple que $7432 = 7 \cdot 1000 + 4 \cdot 100 + 3 \cdot 10 + 2 \cdot 1$.

      Es podrà usar el mètode `Math.pow(a, b)` predefinit de Java, que torna un `double`: $\texttt{a}^{\texttt{b}}$.

Solució:

```
    /** Calcula el nombre de xifres de n, n >= 0. */
    public static int digits(int n) {
        int count = 1;
        while (n > 9) {
            n = n / 10;
            count++;
        }
        return count;
    }

    /** Calcula l'invertit de n, n >=0. Versió 1 */
    public static int reversed(int n) {
        int i = digits(n) - 1, add = 0;
        while (i >= 0) {
            add = add + (n % 10) * (int) (Math.pow(10, i));
            n = n / 10; i--;
        }
        return add;
    }

    /** Calcula l'invertit de n, n >=0. Versió 2 */
    public static int reversed(int n) {
        int i = digits(n), add = 0;
        while (i > 0) {
            add = add * 10 + (n % 10);
            n = n / 10; i--;
        }
        return add;
    }
```

3. 2 punts **Es demana:** escriure un mètode estàtic que tinga com a paràmetre un array `a` de `char` i que escriga en l'eixida estàndard, línia a línia, tots els *sufixos* de la cadena de caràcters en `a`, des del més curt endavant. S'entén per sufix qualsevol subcadena que comprèn els caràcters des d'un donat fins a l'últim inclusivament. Per exemple, si `a` és {'s','t','a','t','i','c'}, s'ha d'escriure:

```
c
ic
tic
atic
tatic
static
```

Solució:

```
/** Versió 1 */
public static void suffixes(char[] a) {
    String s = "";
    for (int i = a.length - 1; i >= 0; i--) {
        s = a[i] + s;
        System.out.println(s);
    }
}

/** Versió 2 */
public static void suffixes(char[] a) {
    for (int i = a.length - 1; i >= 0; i--) {
        for (int j = i; j < a.length; j++) {
            System.out.print(a[j]);
        }
        System.out.println();
    }
}
```