Recuperació del segon parcial d'IIP - ETSInf.
Data: 23 de gener de 2015. Duració: 2:30 hores.

1. 6.5 punts Una empresa de missatgeria ha de gestionar la distribució d'enviaments, que poden ser de tipus paquet o document. Cada enviament té com a dades el seu origen (`String`), el seu destí (`String`), una referència (`String`) que l'identifica, un tipus (`int`), i una ubicació (`String`). Un enviament es pot guardar en un magatzem, on se li assigna com a ubicació el lloc concret o estant on es guarda l'enviament mentre està en aquest magatzem. Quan un enviament s'extrau d'un magatzem, se li dóna de baixa canviant-li la ubicació a `"En trànsit"`.

   Per a representar el problema es disposa de la classe `Enviament` de la que es mostra a continuació un resum de la seua documentació:

Field Summary

static int	**DOCUMENT** Constant que indica que l'enviament es un document.
static int	**PAQUET** Constant que indica que l'enviament es un paquet.

Constructor Summary

Enviament(String origen, String desti, String ref, int tipus) Crea un enviament amb origen, desti, referencia i tipus (PAQUET o DOCUMENT) donats; i amb ubicacio per defecte "En transit".

Method Summary

boolean	**equals**(Object o) Comprova si un enviament es igual a un altre o donat, aço es, si les seues referencies coincideixen.
String	**getDesti**() Torna el desti de l'enviament.
String	**getOrigen**() Torna l'origen de l'enviament.
String	**getRef**() Torna la referencia de l'enviament.
int	**getTipus**() Torna el tipus de l'enviament.
String	**getUbicacio**() Torna la ubicacio de l'enviament.
void	**setUbicacio**(String novaUbic) Actualitza a novaUbic la ubicacio de l'enviament.
String	**toString**() Torna un String amb la informacio de l'enviament.

   Observa que un objecte `Enviament` es crea a partir dels `String` que indiquen el seu `origen`, `desti` i `referencia` i del `int` que indica el seu `tipus`. Segons s'ha indicat més amunt, un `Enviament` té també un altre atribut `String ubicacio`, que s'inicia a `"En trànsit"` quan es crea l'objecte.

   La classe té dues constants `int` públiques, `DOCUMENT` i `PAQUET`, iniciades als valors enters que es corresponen amb els respectius tipus d'enviaments.

   Nota que es disposa de mètodes consultors per a totes les dades de l'`Enviament`, i un mètode modificador de la ubicació.

   Es demana: implementa la classe `Magatzem` que representa un magatzem d'enviaments mitjançant els components (atributs i mètodes) que s'indiquen a continuació.

   a) (0.5 punts) Atributs:

   - `MAX_ENVIAMENTS`, una constant de classe (o estàtica) que representa el número màxim d'enviaments que caben en el magatzem, i que ha de valer 2000. Recorda que aquesta constant, així com les de la classe `Enviament`, s'han d'usar sempre que se requerisca.
   - `numE`, un enter en l'interval `[0..MAX_ENVIAMENTS]` que representa el número d'enviaments guardats en el magatzem en cada moment.
   - `llista`, un array de tipus base `Enviament`, de capacitat `MAX_ENVIAMENTS`. Els components d'aquest array contenen els enviaments del magatzem en posicions consecutives des de la `0` fins a la `numE-1`, i per ordre d'arribada, és a dir, `llista[0]` és l'enviament més antic dels que estan al magatzem en un moment donat, i `llista[numE-1]` el més recent. No hi ha enviaments repetits.
   - `numDoc` i `numPaq`, enters que representen el número d'enviaments en el magatzem en un moment donat, que són de tipus document o paquet respectivament.

Els mètodes que emmagatzemen o extraguen un enviament del magatzem han de mantenir els enviaments consecutius en l'array, per ordre d'arribada. Els comptadors d'enviaments de cada tipus s'hauran d'actualitzar adequadament.

b) (0.5 punts) Un constructor per defecte (sense paràmetres) que crea un magatzem buit, amb 0 enviaments.

c) (1 punt) Un mètode amb perfil:

```
private int posicioDe(String ref)
```

que torna la posició en la que apareix l'enviament amb referència `ref` en l'array `llista`, o -1 si no està. És un mètode d'utilitat per als mètodes `emmagatzemar`, `buscar` i `donarBaixa` que es demanen més endavant.

d) (1 punt) Un mètode amb perfil:

```
public boolean emmagatzemar(Enviament e, String estant)
```

que actualitza el magatzem afegint-li l'enviament `e`. En el propi enviament `e` se registra la informació de l'estant on es guarda, per la qual cosa el mètode modifica la ubicació de `e` al `String estant` que se li passa al mètode. L'enviament no s'emmagatzema si existeix un altre igual (amb la mateixa referència) o el magatzem està ple, en tal cas el mètode torna `false`. En cas contrari, torna `true` indicant que l'enviament s'ha pogut guardar.

e) (0.75 punts) Un mètode amb perfil:

```
public Enviament buscar(String ref)
```

que torna l'enviament del magatzem la referència del qual és `ref`. Si no el troba, torna `null`.

f) (1 punt) Un mètode amb perfil:

```
public Enviament donarBaixa(String ref)
```

que busca l'`Enviament` de referència `ref`. Si el troba, canvia la seua ubicació a `"En trànsit"`, l'extrau del magatzem i el torna com a resultat. Si no hi ha cap enviament amb aquesta referència, torna `null`.

Recorda que el mètode ha de mantenir els enviaments consecutius en les primeres posicions de l'array `llista`, i respectant l'ordre d'arribada, de manera que per tal d'eliminar un enviament de l'array, tots els enviaments a la seua dreta s'han de moure una posició cap a l'esquerra.

g) (1 punt) Un mètode amb perfil:

```
public Enviament[] obtenirEnviaments(int tipus)
```

que torna un array de `Enviament` del tipus que indique el paràmetre `tipus`. La longitud d'aquest array serà igual al número d'enviaments d'aquest tipus o 0 si no hi ha cap (també es torna un array de talla 0 si el tipus no correspon a cap dels tipus vàlids de `Enviament`).

h) (0.75 punts) Un mètode amb perfil:

```
public String toString()
```

que torna un `String` amb la informació de tots els enviaments que hi ha al magatzem, per ordre d'antiguitat, cadascun en una línia distinta.

Solució:

```
public class Magatzem {
    public static final int MAX_ENVIAMENTS = 2000;
    private Enviament[] llista;
    private int numE, numDoc, numPaq;

    public Magatzem() {
        llista = new Enviament[MAX_ENVIAMENTS];
        numE = numDoc = numPaq = 0;
    }

    private int posicioDe(String ref) {
        int i = 0;
        while (i < numE && !ref.equals(llista[i].getRef())) i++;
        if (i < numE) return i; else return -1;
    }

    public boolean emmagatzemar(Enviament e, String estant) {
        if (posicioDe(e.getRef()) == -1 || numE == MAX_ENVIAMENTS) return false;
        e.setUbicacio(estant);
        llista[numE++] = e;
        if (e.getTipus() == Enviament.PAQUET) numPaq++; else numDoc++;
        return true;
    }
```

```
    public Enviament buscar(String ref) {
        int pos = posicioDe(ref);
        if (pos != -1) return llista[pos]; else return null;
    }

    public Enviament donarBaixa(String ref) {
        int pos = posicioDe(ref);
        if (pos == -1) return null;
        Enviament e = llista[pos];
        e.setUbicacio("En transit");
        for (int i = pos + 1; i < numE; i++) llista[i - 1] = llista[i];
        numE--;
        if (e.getTipus() == Enviament.PAQUET) numPaq--; else numDoc--;
        return e;
    }

    public Enviament[] obtenirEnviaments(int tipus) {
        int talla = 0;
        switch (tipus) {
            case Enviament.DOCUMENT: talla = numDoc; break;
            case Enviament.PAQUET: talla = numPaq; break;
        }
        Enviament[] lEnv = new Enviament[talla];
        for (int i = 0, j = 0; j < talla; i++)
            if (llista[i].getTipus() == tipus) {
                lEnv[j] = llista[i];
                j++;
            }
        return lEnv;
    }

    public String toString(){
        String result = "Llistat d'enviaments del magatzem:\n";
        for (int i = 0; i < numE; i++)
            result += llista[i] + "\n";
        return result;
    }
}
```

2. 1.75 punts **Es demana:** Implementa un mètode de classe (o estàtic) tal que, donats un array `a` de `double` (`a.length` $>$ `0`) i un enter `n` (`n` > 0), modifique tots els elements de l'array de forma que el valor màxim del mateix siga `n` i la resta estiguen escalats amb aquest valor. És a dir, si el màxim de `a` és `m`, la relació entre el valor nou i l'antic d'un component de l'array ha de ser `n/m`. Per exemple, si l'array `a` és {`4.5, 27.0, 18.0, 1.5`} i `n = 9`, `a` ha de canviar a {`1.5, 9.0, 6.0, 0.5`}.

Solució:

```
/** a.length > 0, a[i] > 0 ∀i: 0 <= i < a.length, n > 0 */
public static void escalar(double[] a, int n) {
    double maxim = a[0];
    for (int i = 1; i < a.length ; i++)
        if (a[i] > maxim) maxim = a[i];
    for (int j = 0; j < a.length; j++)
        a[j] = (a[j] / maxim) * n;
}
```

3. 1.75 punts El postulat de Bertrand s'enuncia així: "Si n és un número natural major que 3, aleshores sempre existeix un número primer p tal que $n < p < 2 * n - 2$ ". En una certa classe es disposa d'un mètode amb perfil:

```
/** n > 1 */
public static boolean es_primer(int n)
```

que, donat un `n` major que 1, comprova si és un número primer.

Es demana: Implementa un mètode estàtic (que s'escriuria dins de la mateixa classe) que, donat un número natural `n` tal que `n > 3`, determine, usant el mètode `es_primer` anterior, quin número primer `p` compleix el postulat de Bertrand per a `n`. D'haver-ne més d'un, ha de tornar el més menut. Per exemple, per a `n = 8`, els números primers dins de l'interval `[9,13]` són el `11` i el `13`, per tant, el mètode ha de tornar `11`.

Solució:

```
/** n > 3 */
public static int bertrand(int n) {
    int p = n + 1;
    boolean enc = false;
    while (p < (2 * n - 2) && !enc)
       if (es_primer(p)) enc = true;
       else p++;
    return p;
}
```