# IIP - Recuperació del Segon Parcial - ETSInf
## Data: 25 de gener de 2013. Duració: 2:30 hores.

1. [2 punts] Escriure un mètode estàtic que donat un valor enter $n \geq 0$ mostre en l'eixida estàndard una figura representant un quadrat de costat $n$. Per exemple, per a $n = 5$ escriuria:

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

**Solució:**

```
/** PRECONDICIÓ: n >= 0. */
public static void quadrat(int n) {
    for (int i=0; i<n; i++) System.out.print('*');
    System.out.println();
    for (int j=1; j<n-1; j++) {
        System.out.print('*');
        for (int k=1; k<n-1; k++) System.out.print(' ');
        System.out.println('*');
    }
    for (int i=0; i<n; i++) System.out.print('*');
    System.out.println();
}
```

2. [2 punts] Donat un array d'enters `a`, amb al menys dues components, es vol conèixer si en ell es compleix que l'element següent de qualsevol amb valor parell té sempre valor senar. Per exemple, per a:

   - {1,3,1,2,1,1,4,1}, es compleix, ja que tots els següents a valors parells són senars,
   - {1,3,2,1,3,6,5,8,2,1}, NO es compleix, ja que després de l'element de valor 8, es troba un amb valor 2 (no senar),
   - {1,5,7,7,9,3}, es compleix, no hi ha cap valor parell,
   - {1,5,3,1,7,2}, es compleix, l'únic parell és l'últim i no té següent.

   **Es demana:** escriure un mètode estàtic que, donat un array de números enters `a`, amb al menys dues components (`a.length`>1), retorne si compleix la propietat enunciada. El mètode ha de realitzar el menor número d'iteracions possible.

**Solució:**

```
/** PRECONDICIÓ: a.length > 1. */
public static boolean parellSenar(int[] a) {
    // El problema és equivalent al de cercar la primera parella de
    // posicions consecutives ocupada per dos números parells:
    boolean enc = false; int i = 1;
    while(i<a.length && !enc)
        if (a[i-1]%2==0 && a[i]%2==0) enc = true;
        else i++;
    return !enc;
}
```

3. 6 punts Una crida telefònica a mòbil es representa mitjançant un objecte de la classe `Crida`. De cada crida es coneix el seu número de telèfon, data, hora i tipus de la crida (entrant/sortint/perduda). El codi següent mostra la classe `Crida` ja implementada:

```
public class Crida {
    public static final int PERDUDA = 0, ENTRANT = 1, SORTINT = 2;
    private String num, data, hora;
    private int tipus; // perduda, entrant o sortint

    /** Constructor */
    public Crida(String num, int tipus, String data, String hora) {
       this.num = num;
       this.tipus = tipus;
       this.data = data;
       this.hora = hora;
    }

    /** Torna el número de la crida */
    public String getNum() { return num; }

    /** Torna el tipus de la crida */
    public int getTipus() { return tipus; }

    /** Torna la informació de la crida, amb el format:
     * "!  nnnnnnnnn dd:mm:aaaa hh:mm" per a una crida perduda,
     * "-> nnnnnnnnn dd:mm:aaaa hh:mm" per a una crida entrant,
     * "<- nnnnnnnnn dd:mm:aaaa hh:mm" per a una crida sortint
     */
    public String toString() {
        String result = "";
        switch(tipus) {
         case PERDUDA: result += "! "; break;
         case ENTRANT: result += "->"; break;
         case SORTINT: result += "<-";
        }
        result += " " + num + " " + data + " " + hora;
        return result;
    }
}
```

**Es demana:** implementar una classe `Historial` que represente l'historial de crides d'un telèfon, és a dir, la llista de crides realitzades o rebudes per aquest telèfon. Les crides es van afegint a l'historial a mesura que es van produint (100 com a màxim), de manera que quan l'historial estiga ple, s'aniran perdent les més antigues.

a) Atributs (0.25 punts):
   - Un atribut públic estàtic constant enter `MAX_CRIDES`, que s'inicie a 100.
   - Un atribut enter `numCrides`, que represente el número de crides de l'historial (valor entre 0 i `MAX_CRIDES`).
   - Un atribut `crides`, array de `Crida`, amb capacitat màxima `MAX_CRIDES`. Els objectes `Crida` s'emmagatzemaran a l'array `crides` en posicions consecutives, des de 0 a `numCrides-1` inclusivament, i per ordre segons es vagen produint i afegint a l'historial.

b) Un constructor (0.25 punts) que inicialice els atributs, creant l'array `crides` amb capacitat `MAX_CRIDES` i sense cap crida inicial.

c) Un mètode (1.5 punts) amb perfil:

```
public void netejar(int m)
```

que, donat un enter `m`, $0 \leq$ `m` $\leq$ `numCrides`, elimine les `m` crides més antigues de l'historial, és a dir, les `m` primeres components de l'array `crides`. Recorda que les crides restants han de quedar, consecutives, en les primeres posicions de l'array.

d) Un mètode (0.5 punts) amb perfil:

```
public void afegir(Crida c)
```

que, donada una `Crida c`, l'afegisca a l'historial. Si l'historial estiguera ple, el netejaria prèviament reduint la seua ocupació a la meitat.

e) Un mètode (1 punt) amb perfil:

```
public Crida cercar(String num)
```

que torne la `Crida` més antiga corresponent al número `num`. Si no hi ha cap crida d'aquest número o hi ha 0 crides a l'historial, torna `null`.

f) Un mètode (1 punt) amb perfil:

```
public String llistar()
```

que torne un `String` amb el llistat de totes les crides per ordre d'antiguitat, començant per la darrera emmagatzemada. El llistat d'una crida es separa del de la següent per `\n`.

g) Un mètode (1.5 punts) amb perfil:

```
public Crida[] filtrarT(int tipus)
```

que compte el nombre de crides del tipus indicat i torne un array amb totes elles, respectant el seu ordre.

**Solució:**

```
public class Historial {
    public static final int MAX_CRIDES = 100;
    private Crida[] crides;
    private int numCrides;

    public Historial() {
       crides = new Crida[MAX_CRIDES];
       numCrides = 0;
    }

    public void netejar(int m) {
        for(int i=m; i<numCrides; i++) crides[i-m] = crides[i];
        for(int j=numCrides-m; j<numCrides; j++) crides[j] = null;
        numCrides = numCrides - m;
    }

    public void afegir(Crida c) {
        if (numCrides==MAX_CRIDES) netejar(MAX_CRIDES/2);
        crides[numCrides++] = c;
    }

    public Crida cercar(String num) {
        int i = 0;
        while(i<numCrides && !crides[i].getNum().equals(num)) i++;
        if (i<numCrides) return crides[i];
        else return null;
    }
```

```
    public String llistar() {
        String str = "";
        for(int i=numCrides-1; i>=0; i--)
            str += crides[i].toString() + "\n";
        return str;
    }

    public Crida[] filtrarT(int tipus) {
        int cont = 0;
        for(int i=0; i<numCrides; i++)
            if (crides[i].getTipus()==tipus) cont++;
        Crida[] aux = new Crida[cont];
        for(int i=0, j=0; i<numCrides; i++)
            if (crides[i].getTipus()==tipus) { aux[j] = crides[i]; j++; }
        return aux;
    }
}
```