PRG - ETSInf. TEORIA. Curs 2015-16. Parcial 2.
31 de maig de 2016. Duració: 2 hores.

1. [1.5 punts] Es disposa d'un array `lS` d'objectes de tipus `String`, tal que no té inicialitzats tots els seus elements (és a dir, alguna de les posicions de l'array conté un valor `null` en lloc d'una `String` ben formada).

   Si s'utilitza el mètode següent per imprimir la longitud de totes les `String` realment existents de `lS`:

```
public static void m1(String[] lS) {
    int k = 0;
    boolean fi = false;
    while (!fi) {
        System.out.print("Posició " + k + ": ");
        System.out.println(lS[k].length() + " caràcters");
        k++;
    }
}
```

   es poden produir les excepcions: `NullPointerException` i `ArrayIndexOutOfBoundsException`. Quan, en realitat, es desitjaria una sortida **sense excepcions** com la que es mostra (l'exemple s'efectua amb un array de 6 elements):

```
Posició 0: 4 caràcters
Posició 1: 9 caràcters
Posició 2: String no inicialitzada
Posició 3: 11 caràcters
Posició 4: String no inicialitzada
Posició 5: 0 caràcters
Posició 6: Inexistent. Fi de l'array
```

   Es demana: Reescriure el mètode `m1` perquè, **tractant exclusivament** les dues excepcions indicades (sense fer ús de l'atribut `length` de l'array ni de la constant `null`), resolga el problema demanat efectuant una sortida com la mostrada en l'exemple.

Solució: Es mostren dues solucions alternatives:

```
// Primera solució:
public static void m1(String[] lS) {
    int k = 0;
    boolean fi = false;
    try {
        while (!fi) {
            System.out.print("Posició " + k + ": ");
            try {
                System.out.println(lS[k].length() + " caràcters");
            } catch (NullPointerException np) {
                System.out.println("String no inicialitzada");
            }
            k++;
        }
    } catch (ArrayIndexOutOfBoundsException io) {
        System.out.println("Inexistent. Fi de l'array");
    }
}
```

```
// Segona solució:
public static void m1(String[] lS) {
    int k = 0;
    boolean fi = false;
    while (!fi) {
        System.out.print("Posició " + k + ": ");
        try {
            System.out.println(lS[k].length() + " caràcters");
        } catch (NullPointerException np) {
            System.out.println("String no inicialitzada");
        } catch (ArrayIndexOutOfBoundsException io) {
            System.out.println("Inexistent. Fi de l'array");
            fi = true;
        }
        k++;
    }
}
```

2. 2.5 punts **Es demana** implementar un mètode estàtic (de classe) tal que:

   - Reba com a argument una `String` que conté la ruta i nom d'un fitxer de text.
   - Propague l'excepció `FileNotFoundException` si no fóra possible obrir el fitxer el nom del qual s'ha rebut.
   - Construeixca i retorne un objecte `LlistaPIIntEnla` amb tots els nombres enters continguts en el fitxer.
   - El fitxer de text rebut pot contenir tokens que siguen representacions vàlides de nombres enters i tokens que no ho siguen. Es desconeix quants tokens hi ha al fitxer.
   - Si el token llegit és una representació vàlida d'un nombre enter, aquest nombre s'ha d'inserir en la llista a retornar.
   - Si el token llegit no és una representació vàlida d'un nombre enter, llavors s'ha de capturar la excepció `InputMismatchException` que es genera en aquest cas, mostrant a la consola d'error un missatge que incloga el nom de l'excepció i el valor del token que l'ha generat. Aquesta circumstància no ha d'impedir que continue la lectura del fitxer.

Solució:

```
public static LlistaPIIntEnla llegir(String f) throws FileNotFoundException {
    LlistaPIIntEnla li = new LlistaPIIntEnla();
    Scanner sc = new Scanner(new File(f));
    while (sc.hasNext()) {
        try {
            li.inserir(sc.nextInt());
        } catch (InputMismatchException e) {
            System.err.println(e + "::" + sc.next());
        }
    }
    sc.close();
    return li;
}
```

3. 3 punts En una cua, de vegades, pot ser d'interès treure algun element `x` indesitjat. Per aquest motiu, **es demana** afegir a la classe `CuaIntEnla` un nou mètode amb perfil:

```
public int desencuar(int x)
```

que traga de la cua la primera ocurrència de `x`, i la retorne. En el cas en què la cua estiga buida ha de llançar l'excepció `NoSuchElementException` amb el missatge `Cua buida`, i si `x` no es troba ha de llançar una excepció del mateix tipus amb el missatge `x no es troba a la cua`.

Nota: a la solució no es podran fer servir els mètodes de la classe `CuaIntEnla`.

Solució:

```
public int desencuar(int x) {
    if (this.talla == 0) { throw new NoSuchElementException("Cua buida"); }
    NodoInt ant = null, aux = this.primer;
    while (aux != null && aux.dada != x) {
        ant = aux; aux = aux.seguent;
    }
    if (aux != null) {
        if (this.primer == aux) { this.primer = aux.seguent; }
        else { ant.seguent = aux.seguent; }
        if (aux == this.ultim) { this.ultim = ant; }
        this.talla--;
        return x;
    } else { throw new NoSuchElementException(x + " no es troba a la cua"); }
}
```

4. 3 punts En una classe diferent a `LlistaPIIntEnla`, **es demana** implementar un mètode amb el següent perfil i precondició:

```
/** Precondició: llista1 i llista2 no contenen elements repetits. */
public static LlistaPIIntEnla elimComuns(LlistaPIIntEnla llista1, LlistaPIIntEnla llista2)
```

que retorne una llista amb els elements comuns d'ambdues llistes, eliminant de `llista1` aquests elements comuns.

Exemple:
Siga una `llista1` amb els valors `6 -5 4 8 -9`,
siga una `llista2` amb els valors `21 8 5 -9 -5 16`,
llavors el resultat de `elimComuns(llista1, llista2)` haurà de ser una llista `-5 8 -9`, i haurà de deixar `llista1` amb els elements `6 4`.

Solució:

```
/** Precondició: llista1 i llista2 no contenen elements repetits. */
public static ListaPIIntEnla elimComuns(LlistaPIIntEnla llista1, LlistaPIIntEnla llista2) {
    LlistaPIIntEnla result = new LlistaPIIntEnla();
    llista1.inici();
    while (!llista1.esFi()) {
        int x = llista1.recuperar();
        llista2.inici();
        while (!llista2.esFi() && x != llista2.recuperar()) { llista2.seguent(); }
        if (llista2.esFi()) { llista1.seguent(); }
        else { llista1.eliminar(); result.inserir(x); }
    }
    return result;
}
```