PRG - ETSInf. TEORIA. Curs 2017-18. Recuperació Parcial 2.
19 de juny de 2018. Duració: 2 hores.

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

1. 2.5 punts **Es demana:** implementar un mètode estàtic que recupere les dades enteres d'un fitxer de text, el nom del qual es proporcione com a paràmetre, i els emmagatzeme en una `StackIntLinked`, que en serà el resultat, de manera que la primera dada entera que es trobe en el fitxer ha de quedar en la base de la pila.

   S'ha de tindre en compte el següent:

   - El fitxer pot contindre dades que no siguen enters, per tant, en produir-se l'excepció `InputMismatchException` s'ha de capturar i avançar la línia.
   - Si el fitxer no conté enters, o si no existeix, s'haurà de retornar una pila buida.
   - En cas de que el fitxer no existeixca, l'excepció corresponent ha de tractar-se localment escrivint per la sortida estàndard el missatge `"No s'ha trobat el fitxer"`.
   - En qualsevol cas, si s'ha creat correctament el scanner de lectura, aquest s'haurà de tancar.

Solució:

```
public static StackIntLinked fileToStack(String name) {
    Scanner s = null;
    StackIntLinked stack = new StackIntLinked();
    try {
        s = new Scanner(new File(name));
        while (s.hasNextLine()) {
            try {
                stack.push(s.nextInt());
            } catch (InputMismatchException n) {
                s.nextLine();
            }
        }
    } catch (FileNotFoundException e) {
        System.out.println("No s'ha trobat el fitxer");
    } finally {
        if (s != null) { s.close(); }
    }
    return stack;
}
```

2. 2.5 punts **Es demana:** implementar un mètode estàtic iteratiu que copie els elements de `l`, una llista `ListPIIntLinked` passada com a paràmetre, en una nova llista, resultat a retornar, els elements de la qual estiguen en el mateix ordre, però restant a tots els elements el valor mínim de la llista `l`. Per exemple, si la llista original `l` és `[12] 50 10 120` , el mínim és `10` i, per tant, s'ha d'obtindre la llista `2 40 0 110 [ ]` .

   A més, cal tindre en compte que:

   - Si la llista `l` està buida, s'ha de retornar `null`.
   - En qualsevol cas, el contingut inicial de la llista `l` s'ha de conservar, a excepció de la posició del seu punt d'interés (que es permet modificar).
   - **REQUISIT:** El mètode demanat és d'una classe diferent a `ListPIIntLinked`. Per tant, la seua implementació ha de fer-se usant els mètodes públics de `ListPIIntLinked` exclusivament.

Solució:

```
public static ListPIIntLinked subtractMinimumToList(ListPIIntLinked l) {
    if (l.empty()) { return null; }
    ListPIIntLinked res = new ListPIIntLinked();
    l.begin();
    int min = l.get();
    while (!l.isEnd()) {
        if (l.get() < min) { min = l.get(); }
        l.next();
    }
    for (l.begin(); !l.isEnd(); l.next()) {
        res.insert(l.get() - min);
    }
    return res;
}
```

3. 2.5 punts **Es demana:** implementar un mètode estàtic que reba una seqüència enllaçada `NodeInt` i retorne una altra seqüència enllaçada `NodeInt` que continga solament les dades parelles de la seqüència rebuda. Per exemple, si la seqüència rebuda conté les següents dades: `4 7 2 8 9 3 6`, s'ha de retornar la següent seqüència: `4 2 8 6`. S'ha de tindre en compte que si la seqüència rebuda és `null` o si no conté dades parelles, s'ha de retornar `null`.

Solució:

```
public static NodeInt evenSubsequence(NodeInt seq) {
    NodeInt first = null;
    NodeInt last = null;
    while (seq != null) {
        if (seq.data % 2 == 0) {
            if (first == null) {
                last = new NodeInt(seq.data);
                first = last;
            }
            else {
                last.next = new NodeInt(seq.data);
                last = last.next;
            }
        }
        seq = seq.next;
    }
    return first;
}
```

4. 2.5 punts **Es demana:** implementar en la classe `QueueIntLinked` un mètode d'instància que divideixca una cua en dues mitats, amb perfil `public QueueIntLinked divideQueue()`.
   Tot i tenint en compte que:
   - Precondició: la cua inicial (`this`) té almenys dos elements.
   - La divisió es realitza de forma que la cua inicial es queda amb la primera mitat dels elements, i es retorna una cua amb la resta dels elements.
   - Les cues resultants mantenen l'ordre dels elements en la cua inicial.
   - Si la cua inicial té un quantitat imparell d'elements, serà la cua retornada la que tindrà una longitud superior en una unitat.

   Exemples:

Cua inicial	Cua inicial modificada	Cua retornada
1 2 2 4 3 1	1 2 2	4 3 1
1 2 2 4 3 1 1	1 2 2	4 3 1 1

REQUISIT: En el mètode demanat cal usar exclusivament els atributs de `QueueIntLinked`, el seu constructor, i referències a `NodeInt`. Per tant, NO es permet cap invocació als mètodes de la classe.

Solució:

```
    public QueueIntLinked divideQueue() {
        QueueIntLinked nq = new QueueIntLinked();
        int middle = size / 2;
        NodeInt aux = this.first;
        for (int i = 0; i < middle - 1; i++) {
            aux = aux.next;
        }
        nq.first = aux.next;
        nq.last = this.last;
        aux.next = null;
        this.last = aux;
        nq.size = this.size - middle;
        this.size = middle;
        return nq;
    }
```

ANNEX

Mètodes de les classes `StackIntLinked` i `ListPIIntLinked`, i atributs de la classe `QueueIntLinked`.

```
public class StackIntLinked {
        ...
    public StackIntLinked() { ... }
    public boolean empty() { ... }
    public int size() { ... }
    public void push(int x) { ... }
    public int pop() { ... }
    public int peek() { ... }
    public boolean equals(Object o) { ... }
    public String toString() { ... }
}

public class ListPIIntLinked {
    ...
    public ListPIIntLinked() { ... }
    public boolean empty() { ... }
    public int size() { ... }
    public boolean isEnd() { ... }
    public void begin() { ... }
    public void next() { ... }
    public void insert(int x) { ... }
    public int remove() { ... }
    public int get() { ... }
    public boolean equals(Object o) { ... }
    public String toString() { ... }
}

public class QueueIntLinked {
    private NodeInt first;
    private NodeInt last;
    private int size;
    public QueueIntLinked() { }
    ...
}
```