EXAMEN DE PRÀCTIQUES DE PROGRAMACIÓ 23/06/2015 Duració: 1 hora

A omplir pel professor

Escriu el teu nom i cognoms, i el codi del teu grup de pràctiques, ací:

Nom i cognoms:		Grup:	

Pregunta 1. **(2 punts)**

Es vol completar la implementació d'un mètode **recursiu**, anomenat **`esIniciIFi`**. Aquest mètode, donats **a** i **b** de la classe **`String`**, determina si **a** és prefixe directe i sufixe invers de **b** (és a dir, si els caràcters de **a** es poden llegir, d'esquerra a dreta, en les posicions inicials de **b**, i també es poden llegir, de dreta a esquerra, en les posicions finals de **b**). Com exemples d'ús d'aquest mètode, considera les següents línies de codi:

```
boolean x = esIniciIFi("lasa","lasaladelasal");   // x ← true
boolean y = esIniciIFi("des","desiertodelased");  // y ← true
boolean z = esIniciIFi("los","losdiasdesol");     // z ← true
boolean v = esIniciIFi("susi","susaludasusta");   // v ← false
boolean w = esIniciIFi("dos","dosdiasmalos");     // w ← false
```

Es disposa de la següent implementació parcial del mètode:

```
0    /** precondició: 2 * a.length() <= b.length() */
1    public static boolean esIniciIFi(String a, String b) {
2      if (a.length() == 0) return true;
4      else return a.charAt(0) == ...
5           && a.charAt(0) == ...
6           && esIniciIFi( ...
7    }
```

Es demana: Completar la implementació, afegint el codi necessari en el cas general (on indiquen els punts suspensius).

```
public static boolean esIniciIFi(String a, String b) {
   if (a.length() == 0) return true;
   else return a.charAt(0) == b.charAt(0)
        && a.charAt(0) == b.charAt(b.length()-1)
        && esIniciIFi(a.substring(1), b.substring(1, b.length()-1));
}
```

Pregunta 2. **(2 punts)**

Considera la taula següent amb els temps d'execució (en mil·lisegons) de dos algorismes que resolen el mateix problema:

# Talla n	Algorisme A TA(n)	Algorisme B TB(n)
1000	10148.00	397.81
2000	19753.00	1501.83
3000	25327.00	3312.83
4000	34552.00	5820.27
5000	44145.00	9029.67
6000	51902.00	12936.26
7000	61002.00	17543.27
8000	69827.00	22835.75
9000	77687.00	30385.04
10000	86409.00	36873.96
11000	94799.00	43157.29
12000	103808.00	51282.19
13000	112750.00	59839.54
14000	124301.00	69399.81
15000	131841.00	81848.08
16000	143048.00	90444.16
17000	154218.00	102002.64
18000	166435.00	116535.31
19000	173328.00	127247.52
20000	182737.00	143252.33

Es demana que contestau a les següents qüestions justificant la resposta:

a) Quin és el cost asimptòtic que millor s'ajusta a les dades de la columna **Algorisme A**?
b) Quin és el cost asimptòtic que millor s'ajusta a les dades de la columna **Algorisme B**?
c) Quin algorisme té un comportament asimptòtic millor?

a) El cost asimptòtic que millor s'ajusta a les dades de la columna de l'algorisme A és un cost lineal, doncs a mesura que la talla del problema es multiplica per dos el cost temporal també es multiplica per dos. Si f(2n)/f(n) = 2n/n = 2, aleshores f(n) és lineal.

b) El cost asimptòtic que millor s'ajusta a les dades de la columna de l'algorisme B és un cost quadràtic. El ratio f(2n)/f(n) dona un valor molt proper a 4. Aleshores, si una funció f(n) és quadràtica es compleix que f(2n) = 4 f(n), multiplicant per 2 la talla es multiplica per 4 el valor de la funció. Si per exemple la funció és f(x) = x*x, es veu clarament com f(2*x) = 4*f(x) donat que f(2*x) = (2*x)*(2*x) = 2*2*x*x = 4*x*x = 4*f(x).

c) Evidentment, l'algorisme A té un millor comportament asimptòtic, encara que per a valors xicotets de la talla del problema siga més lent que l'algorisme B.

A omplir pel professor

Pregunta 3. **(3 punts)**

A omplir pel professor

Considera el projecte de la pràctica 4, que s'ha modificat per tal de permetre un descobert en el compte (és a dir, que el saldo d'un compte bancari puga ser negatiu).

Per això, en la classe **`Compte`**, s'ha afegit un atribut, anomenat **`admetDescobert`**, que indica si s'autoritza un saldo negatiu, i un atribut estàtic, anomenat **`DESCOBERT_MAX`**, amb la quantitat màxima de descobert permès. Per tant, els atributs d'aquesta classe ara són:

```
private static final double DESCOBERT_MAX = 1000.0;
private double saldo;
private int numCompte;
private boolean admetDescobert;
```

A més, s'ha implementat una classe **`DescobertExceditException`**, subclasse d'**`Exception`**, per a gestionar les situacions en les que el saldo negatiu del compte puguera excedir el màxim descobert permès.

Es demana: Implementar, en la classe **`Compte`**, un mètode d'instància, anomenat **`pagarRebut`**, tal que, donat un nombre real (import del rebut), realitze el càrrec del mateix en el compte excepte si es donen les següents circumstàncies:

- Si el compte no admet descobert i l'import del rebut excedeix el saldo del compte, es llançarà l'excepció **`SaldoInsuficientException`** (amb un missatge adequat), i aquesta es propagarà.
- Si el compte admet descobert i l'import del rebut excedeix el saldo i el descobert màxim autoritzat, es llançarà l'excepció **`DescobertExceditException`** (amb un missatge adequat), i aquesta es propagarà.

```
public void pagarRebut(double quantitat)
   throws SaldoInsuficientException, DescobertExceditException {
   if (!admetDescobert && quantitat > saldo)
      throw new SaldoInsuficientException(
      "Rebut impagat perquè no hi ha prou diners al compte!");
   if (admetDescobert && quantitat > saldo + DESCOBERT_MAX)
      throw new DescobertExceditException(
      "Rebut impagat perquè excediria el descobert autoritzat!");
   saldo -= quantitat;
}
```

Pregunta 4. **(3 punts)**

En el projecte de la pràctica 5, es vol obtenir de la concordança totes les paraules que apareguen en una línia donada del text.

Suposa que en la classe **`CuaIntEnla`**, a més de tots els mètodes coneguts, s'ha implementat un mètode amb el següent perfil: **`public boolean conte(int n)`**, que torna **`true`** si el valor **`n`** està en la cua d'enters que l'invoca, i torna **`false`** en cas contrari.

Recorda que les estructures de dades **`Concordanca`** i **`NodeCnc`**, vistes en pràctiques, tenen els atributs següents:

`Concordanca`
```
private NodeCnc prim;
private int talla;
private boolean esOrd;
private String separadors;
```

`NodeCnc`
```
String paraula;
CuaIntEnla numLins;
NodeCnc seguent;
```

Es demana: Implementar, en la classe **`Concordanca`**, un mètodo que, donat un enter **n** que representa un número de línia, torne una cadena de caràcters formada per totes les paraules que apareixen en la línia **n** del text, separades per comes. El mètode ha de funcionar tant si la concordança està ordenada com si no ho està. No importa l'ordre en què les paraules apareguen en la **`String`** resultat.

Exemple: donada la concordança que es mostra en el requadre de la dreta, si el mètode és invocat per aquest objecte **`Concordanca`**, i rep el valor 2, torna la cadena **"Más, alto, afirmar, "**; i si rep el valor 4, torna **"el, sí, y, "**.

```
Más (2):      1  2
azul (2):     1  1
que (2):      1  3
el (4):       1  3  4  4
alto (1):     2
afirmar (1):  2
amor (1):     3
querer (1):   3
sí (3):       3  4  4
y (2):        4  4
```

```
public String paraulesEnLinia(int n) {
   String s = "";
   for (NodeCnc p = prim; p!=null; p = p.seguent)
      if (p.numLins.conte(n)) s += p.paraula + ", ";
   return s;
}
```

A rellenar por el profesor