Examen de Fundamentos de Informática. 28/01/ 2011

Centro Politécnico Superior

Universidad de Zaragoza

Duración total del examen: 3 horas

Ejercicio 1 [0,0 puntos]

El banco BFI ha almacenado en un fichero de texto, llamado "ingresosBFI.txt", todos los movimientos monetarios de sus clientes desde su fundación. Cada línea del fichero corresponde con la información de un único cliente. Más concretamente, cada línea consta de: un primer entero que determina el número de movimientos realizados por el cliente, una secuencia de enteros formada por el importe de cada movimiento (los enteros positivos representan ingresos y los negativos retiradas de dinero); y, finalmente, el nombre del cliente. Por ejemplo, la siguiente línea representa que Luis Alberto Pérez Martín ha realizado 4 movimientos siendo su saldo final de 4250 euros.

4 3500 1250 -3000 2500 Luis Alberto Pérez Martín

El fichero está libre de errores y el banco conoce su número de clientes, es decir, el número de líneas del fichero.

El departamento de informática del banco BFI ha programado en JAVA la clase `CuentaCorriente`. Un objeto de esta clase almacena la información de una cuenta corriente, concretamente, el titular de la cuenta y su saldo. Para almacenar esta información, la clase consta de dos atributos privados: `titular` y `saldo`. Además, la clase ofrece un constructor y los correspondientes métodos SET y GET.

```
public class CuentaCorriente {

      private String _titular;
      private int _saldo;

      public CuentaCorriente(String t, int s){}
      public void setTitular(String t){}
      public void setSaldo(int s){}
      public String getTitular(){}
      public int getSaldo(){}
}
```

El banco quiere procesar la información del fichero de texto y almacenarla en una estructura arreglo de tipo `CuentaCorriente[]`. Se creará un objeto `CuentaCorriente` con la información de cada cliente almacenada en el fichero. El nombre del titular corresponderá con el nombre del cliente y el saldo será calculado sumando todos sus movimientos. Por tanto, **SE PIDE** programar en Java el siguiente método:

```
public static void CuentaCorriente[] crearCC (String fileName, int max_cc)
```

Los parámetros de entrada del método son el nombre de un fichero de texto con el formato anterior (`fileName`) y el número de clientes (o filas) del fichero (`max_cc`). El método debe devolver como resultado la estructura de tipo arreglo `CuentaCorriente[]` con la información de los clientes del banco.

Ejercicio 2 [0,0 puntos]

La crisis económica actual ha provocado que el banco BFI del ejercicio anterior se haya fusionado con el banco BGT. Utilizado el método implementado en el apartado anterior, cada banco ha almacenado en una estructura de tipo `CuentaCorriente[]` la información de sus respectivos clientes. Dado que BFI y BGT se han fusionado, les interesa tener la información de los dos bancos en una única estructura de tipo `CuentaCorriente[]` que contenga los clientes de ambos bancos.

SE PIDE programar en Java el siguiente método:

```
public static void CuentaCorriente[] mezclarCC (CuentaCorriente[] cb1, CuentaCorriente[] cb2)
```

Los parámetros de entrada del método son las estructuras de tipo arreglo con las cuentas corrientes de los dos bancos (`cb1` y `cb2`, respectivamente). Por simplicidad, un cliente no puede tener una cuenta en ambos bancos. Además, las cuentas corrientes contenidas en cada estructura están ordenadas de menor a mayor según el saldo. Por tanto, el método `mezclarCC` debe crear una nueva estructura de tipo `CuentaCorriente[]` que contenga las cuentas de `cb1` y `cb2` y esté también ordenada de menor a mayor según el saldo.

IMPORTANTE: No puede utilizarse ningún algoritmo predefinido de ordenamiento de arreglos para ordenar por saldo las cuentas de la estructura resultado del método.

Ejercicio 3 [0,0 puntos]

Escribe un programa en Java que pida un entero en base 10 al usuario y lo visualice en todas las bases, desde base 2 hasta base 36. El programa leerá un entero de tipo **long**, y presentará una línea con la representación del número en cada base. Los números representados en las respectivas bases se escribirán en una columna ajustados a la derecha, utilizando como anchura de la columna el número de caracteres necesarios para su representación en base dos. Antes de representar el número en las respectivas bases en cada línea, se escribirá la cadena "Base", la base ajustada a la derecha utilizando 3 caracteres, y el carácter ":".

Previo a la representación del número en las distintas bases, se escribirá el número en base 10, y una línea que contiene tantos caracteres "-" como los necesarios para representar las líneas siguientes. A continuación se muestra un ejemplo de ejecución del programa en el que se presenta el formato de salida descrito que debes implementar.

```
Introduce un entero (base 10): 89898934323425443
Número en Base 10:89898934323425443
------------------------------------------------------------------
Base  2:100111111011000101001101010111110000010001011000010100011
Base  3:                      12101112200220101200200211111222002l
Base  4:                             1033312022122233200202300220З
Base  5:                                 122323121311444324410З233
Base  6:                                    4033103024142224454311
Base  7:                                     1061305454452254164З6
Base  8:                                       4773051527602130243
Base  9:                                       534562635062445807
Base 10:                                        89898934323425443
Base 11:                                        1A580630A39210447
Base 12:                                         5A0291B8B677A397
Base 13:                                         19AA8431920556A8
Base 14:                                          8139918CCDCB49D
Base 15:                                          312D378C6094E2D
Base 16:                                          13F629ABE08B0A3
Base 17:                                           9151EAA94576F3
Base 18:                                           45CE8B3G8F8B07
Base 19:                                           22BDG4B1DHD49H
Base 20:                                           11IJ3H28G183C3
Base 21:                                            C4DDJ67C59HDD
Base 22:                                            6LIGF6AA0GC27
Base 23:                                            4281KB12C7FEK
Base 24:                                            2B1L7KH3NHIMJ
Base 25:                                            1CHGBG9OHO5HI
Base 26:                                             OCLCGO57GOBL
Base 27:                                             G4H2J522DEO7
Base 28:                                             ANE8E8QJ4PIR
Base 29:                                             7AJP8N0NGEQO
Base 30:                                             527A542AQQ1D
Base 31:                                             3GL4TANEP3QJ
Base 32:                                             2FR2JAV0HC53
Base 33:                                             1PMWUD2KUIR7
Base 34:                                             19IL5DEG9EG3
Base 35:                                              WKLS8CBP4XD
Base 36:                                              OL6IGAEGTR7
```