Examen de Programación
(Grados en Física y Matemáticas)

Junio 2017

Primera parte (1.25 puntos por cuestión, 50% nota del examen)

1) Escribir el código Java de un método que permite calcular la mediana de un array de valores enteros que se pasa como parámetro. La mediana es el valor que ocupa el lugar central de todos los datos cuando éstos están ordenados de menor a mayor.

   En primer lugar el método debe comprobar que los datos del array están ordenados de menor a mayor, comparando cada casilla con la siguiente. Si se comprueba que una casilla es menor que la anterior a ella se lanzará la excepción `NoOrdenados`, que se supone declarada en una clase aparte.

   Para calcular la mediana (*Me*), si el array tiene un número impar de valores la mediana es el valor central del mismo. Ejemplo: si el array es {2, 3, 4, 4, 5, 5, 5, 6, 6} *Me*= 5

   Si el array tiene un número par de valores la mediana es la media entre las dos puntuaciones centrales. Por ejemplo para {7, 8, 9, 10, 11, 12} *Me*= 9.5

2) Hacer un método Java para escribir en un fichero de texto cuyo nombre se pasa como parámetro los contenidos de dos arrays de números reales que también se pasan como parámetros. El primer array contiene frecuencias (Hz) y el segundo voltajes de una señal eléctrica (V). En el fichero se escribirá en primer lugar una línea de encabezamiento que explique los datos que vienen a continuación. A continuación escribir los números de los arrays en columnas, de forma que la columna izquierda contenga los datos del primer array y la columna derecha los del segundo. Cuando uno de los dos arrays (o ambos) se hayan escrito por completo, finalizar el método. Es decir, si un array es más largo que otro su parte final quedará sin escribir.

3) La tabla siguiente muestra la tabla de retenciones del IRPF para 2017. Una persona que gane entre 0 y 12.450€ de salario bruto tendrá una retención del 19% de su salario. Si gana más, por ejemplo 13.450€ se le retendrá el 19% por los primeros 12.450€ y un 24% por los siguientes 1.000 euros. Y así sucesivamente. Escribir un método Java que, dado el salario bruto retorne la retención a aplicar. El método seguirá este pseudocódigo, que está casi completo:

TABLA TRAMOS IRPF		
de	***a***	***Retención***
0,00 €	12.450,00 €	19,00%
12.450,00 €	20.200,00 €	24,00%
20.200,00 €	35.200,00 €	30,00%
35.200,00 €	60.000,00 €	37,00%
60.000,00 €		45,00%

```
método retencionIRPF(real salarioBruto) retorna real
   tablaTramo= array con los datos de la columna izquierda de la tabla
   tablaRetencion= array con datos de la col. dcha. de la tabla divididos entre 100
   real retencion=0
   para i desde 0 hasta tamaño de tablaTramo-2 hacer
      si salarioBruto>tablaTramo[i+1] entonces
         // la retención es sobre el tramo completo y hay que seguir en el bucle
         retencion=retencion+(tablaTramo[i+1]-tablaTramo[i])*tablaRetencion[i]
      si no
```

```
            // la retención es sobre un tramo final parcial y hay que salir del bucle
            retencion=retencion+(salarioBruto-tablaTramo[i])*tablaRetencion[i]
            retorna retencion
        fin si
    fin para
    // falta calcular el último tramo del IRPF, el del 45%
    retencion=retencion+...
    retorna retencion
fin método
```

En este pseudocódigo falta la penúltima instrucción, en la que se añade el último tramo de la retención (el que lleva el 45%) a la variable retencion. Pensar cómo se hace este último cálculo y ponerlo en el código.

4) En un computador con sistema operativo Linux se desea escribir un *script* para agrupar en una carpeta datos de otras tres que están en la carpeta del usuario:
   - experimento1: datos de medidas guardados en ficheros con la terminación .csv
   - experimento2: datos de otras medidas también guardados en ficheros .csv
   - programa: contiene clases java (ficheros .java), documentos (ficheros .txt) y clases compiladas (ficheros .class)

   En primer lugar hacer un esquema de la distribución inicial de carpetas y ficheros

   En segundo lugar escribir el *script,* que debe hacer los siguientes pasos
   - cambiar el directorio de trabajo situándolo en el del usuario
   - crear en el directorio del usuario una carpeta llamada copia_seguridad
   - crear en copia_seguridad una carpeta llamada experimentos
   - copiar en la nueva carpeta experimentos todos los ficheros .csv de experimento1 y de experimento2
   - crear en copia_seguridad una carpeta llamada codigo
   - mover a la nueva carpeta codigo todos los ficheros .java y .txt de la carpeta programa
   - borrar de la carpeta programa todos los ficheros .class, pero mantener la carpeta programa sin borrarla

   En tercer lugar hacer un esquema de la distribución final de carpetas y ficheros.

Examen de Programación
(Grados en Física y Matemáticas)

Junio 2017

Segunda parte (5 puntos, 50% nota del examen)

Se desea hacer parte del software de análisis de las señales de radio captadas por un radiotelescopio provenientes de una supernova. Se dispone para ello de los datos del radiotelescopio almacenados en un fichero con formato "csv".

Se dispone de las clases Modelo y ObsMonocromatica que ya están hechas. Se pueden ver sus diagramas de clases en la figura. Los métodos de la clase Modelo hacen lo siguiente:

Modelo
-double st -double vt -double p

+Modelo (
  double st,
  double vt,
  double p)
+double flujoTeorico(
  double frec)

ObsMonocromatica
-double frec -double flujo -double errFlujo

+ObsMonocromatica(
  double frec, double flujo,
  double errFlujo)
+double frec()
+double flujo()
+double errFlujo()

- *constructor*: Se le pasan los parámetros del modelo teórico, que son st (densidad de flujo máxima en mJy), vt (frecuencia a la que ocurre la densidad de flujo máxima en GHz) y p (exponente de la función de flujo)
- flujoTeorico(): Retorna el flujo monocromático teórico de la supernova (mJy) dada la frecuencia frec en GHz.

La clase ObsMonocromatica es un simple contenedor de tres datos y sus métodos hacen lo siguiente:

- *constructor*: se le pasan los datos de la observación: frecuencia en GHz, flujo en mJy, error del flujo en mJy.
- *observadores*: hay tres observadores, uno para cada atributo, que retornan frecuencia en GHz, flujo en mJy, y error del flujo en mJy.

La clase Observacion recoge en el ArrayList lista los datos de varias observaciones monocromáticas realizadas en el mismo día a diferentes frecuencias. Su diagrama de clases se muestra en la figura. Los métodos añade(), elimina() y leeFichero() están ya implementados (aparecen con fondo más oscuro). Se pide escribir el encabezamiento de la clase, los atributos, el constructor y los métodos indiceConErrorGrande(), chiCuadradoPonderado(), y frecFlujoMax(). La descripción de los métodos es:

- *constructor*: Se le pasa el tiempo transcurrido desde el inicio de la supernova, que se guarda en el atributo tiempo (en días). También crea la lista vacía

- indiceConErrorGrande(): Busca la primera observación que tenga un error relativo (errFlujo/flujo) superior al indicado en errorRelativo. Retorna el índice de la casilla de la lista que contiene esa observación o -1 si no hubiese ninguna.
- chiCuadradoPonderado(): Retorna la estadística *chi cuadrado* ponderada según los errores de las medidas, dados los valores teóricos de estas obtenidos con el método flujoTeorico() del objeto mod. El cálculo se hace con esta fórmula:

$$\chi^2 = \sum_{\forall i} \frac{(M_i - T_i)^2}{\sigma_i^2}$$

  donde M_i es la medida real del flujo, T_i su valor teórico para la misma frecuencia y σ_i es el error del flujo.

Observacion
-ArrayList<ObsMonocromatica> lista -double tiempo
+Observacion(double tiempo) +int indiceConErrorGrande(double errorRelativo) +double chiCuadradoPonderado(Modelo mod) +double frecFlujoMax()
+void añade(ObsMonocromatica obs) +void elimina(int indice) throws IndiceIncorrecto +static Observacion leeFichero(String nombreFichero) throws FileNotFoundException, ErrorDeFormato

- frecFlujoMax(): Retorna la frecuencia a la que el flujo observado es máximo, en GHz.

Los métodos que **no** se piden son:

- añade(): Añade una observación monocromática a la lista.
- elimina(): Elimina de la lista la observación monocromática cuyo índice se indica. Si el índice no está comprendido entre 0 y el número de elementos de la lista menos uno se lanza IndiceIncorrecto
- leeFichero(): Lee del fichero "csv" cuyo nombre se indica los datos de la observación de una supernova y los retorna en un objeto de la clase Observacion. Lanza FileNotFoundException si el fichero no existe o ErrorDeFormato si el formato del fichero csv es incorrecto.

Las excepciones IndiceIncorrecto y ErrorDeFormato están definidas en clases aparte.

Finalmente, se pide hacer un programa principal en una clase aparte que haga lo siguiente:

a. Lee el fichero CSV llamado SN2011dh.csv, obteniendo un objeto de la clase Observacion

b. Busca en la Observacion si existe una casilla con error relativo de flujo mayor que 0.09, lo indica en la pantalla, y en caso de existir elimina esa casilla. Si en este paso se lanzase IndiceIncorrecto se pone un mensaje de error en la pantalla y se continúa con las siguientes instrucciones del main

c. Crea un modelo con st=7.635 vt=4.017 p=3.0 y muestra en pantalla el valor de *chi cuadrado* ponderado usando ese modelo

d. Muestra en pantalla la frecuencia a la que el flujo es máximo

Si en el paso a) se lanzase FileNotFoundException o ErrorDeFormato se abandonarán las instrucciones restantes del main para, a continuación, poner un mensaje en pantalla indicando cuál es el error que ha ocurrido

Valoración:

- encabezamiento de la clase, atributos y constructor: 0.5 puntos
- métodos `indiceConErrorGrande()`, `chiCuadradoPonderado()`, y `frecFlujoMax()`: 1 punto cada uno
- programa principal: 1.5 puntos