Examen de Fundamentos de Computadores y Lenguajes

Examen Final. Septiembre 2007

Cuestiones (5 cuestiones, 5 puntos en total)

1) Se dispone de la siguiente clase enumerada cuyos valores representan diferentes tipos de conversión de unidades de temperatura.

```
public enum UnidadTemp
{
    celsius, kelvin, farenheit
}
```

Las equivalencias entre unas unidades y otras son

$$C = K - 273.15$$
$$C = (F - 32)/1.8$$

Se pide escribir un método al que se le pase un valor de temperatura, un valor que indica las unidades de origen, y un valor que indica las unidades deseadas. El método debe retornar un valor real que corresponda a la temperatura original transformada a las unidades deseadas. El método obedecerá a la siguiente interfaz:

```
public static double convierte(
    double temp, UnidadTemp origen, UnidadTemp deseada)
```

2) Escribir un método al que se le pasa un array de temperaturas y otro array del mismo tamaño con valores de la clase enumerada `UnidadTemp` (ver ejercicio anterior), y que escriba todos los valores contenidos en estos arrays en un fichero de texto llamado `temperaturas.txt`, poniendo en la primera línea la primera temperatura y la primera unidad, en la segunda línea la segunda temperatura y segunda unidad, y así sucesivamente. El método obedecerá a la siguiente interfaz:

```
public static void escribeEnFichero(double[] temp, UnidadTemp[] unidad)
```

3) Escribir en Java el siguiente método descrito mediante pseudocódigo, y que permite calcular de manera eficiente a^n cuando a y n son números naturales.

```
método potencia (entero a, entero n) retorna un número entero
    // Este algoritmo sirve para calcular a elevado a la n
    // cuando a y n son números naturales
    entero i=n
    entero r=1
    entero x=a
    mientras i>0 hacer
        si i es impar entonces
            r=r * x
        fin de si
        x= x * x
        i= i/2
    fin de mientras
    retorna r
fin del método
```

4) Indicar razonadamente el ritmo de crecimiento del tiempo de ejecución del algoritmo siguiente mediante la notación $O(n)$. Este algoritmo obtiene el mismo resultado que el del ejercicio anterior, de una manera diferente. Comparar sus tiempos de ejecución.

```
método potencia (entero a, entero n) retorna un número entero
    // Este algoritmo sirve para calcular a elevado a la n
    // cuando a y n son números naturales
    entero i=n
    entero r=1
    mientras i>0 hacer
        r=r * a
        i= i-1
    fin de mientras
    retorna r
fin del método
```

5) Indicar qué instrucciones son necesarias en un intérprete de órdenes de un sistema operativo Linux para hacer una copia de todos los ficheros fuente Java (los acabados en `.java`) que estén en la carpeta `/home/user3/proyecto1`. La copia debe hacerse en una nueva carpeta llamada `copia_proyecto1` que debe crearse dentro de la carpeta (ya existente) `/home/user3/copias`. Hacer que estas instrucciones funcionen sea cual sea el directorio de trabajo actual.

Examen de Fundamentos de Computadores y Lenguajes

Examen Final. Septiembre 2007

Problema (5 puntos)

Se dispone de una clase ya realizada que sirve para almacenar los datos de un satélite en órbita. Estos datos son sus coordenadas en el espacio tridimensional (x,y,z) y el instante de tiempo en el que se midieron esas coordenadas. El tiempo se mide en segundos desde un instante inicial concreto. La interfaz de la clase se muestra a continuación.

```
/**
 * Clase que contiene las coordenadas espacio-temporales de un satélite
 */
public class DatoOrbital {
    /**
     * Constructor al que se le pasan las coordenadas en metros,
     * y el tiempo en segundos, medido desde un instante inicial concreto
     */
    public DatoOrbital(double x, double y, double z, double t) {...}

    /** Retorna la coordenada x  */
    public double x() {...}

    /** Retorna la coordenada y  */
    public double y() {...}

    /** Retorna la coordenada z  */
    public double z() {...}

    /** Retorna el tiempo  */
    public double tiempo() {...}
}
```

Lo que se pretende es escribir otra clase llamada `Satelite` que permita almacenar una historia de los datos orbitales de un satélite concreto y obtener información a partir de ella. Esta historia se almacena en un array de objetos de la clase `DatoOrbital`. La interfaz de la clase `Satelite` se muestra a continuación:

```
/**
 * Clase que sirve para almacenar los datos orbitales de un satélite
 */
public class Satelite {
    /**
     * Constructor al que se le pasa el número de datos a guardar
     */
    public Satelite(int numDatos, String nombre) {...}

    /** Obtiene un dato orbital del satélite */
    public DatoOrbital leeDatoOrbital() throws NoContesta {...}

    /** Registra un nuevo dato orbital */
    public void registra() {...}

    /** Calcula la posición orbital media entre los instantes tIni y tFin */
    public DatoOrbital posMedia(int tIni, int tFin)  {...}

    /** Retorna la posición orbital en t, o lanza NoExiste si no se encuentra
     */
    public DatoOrbital pos(int t) throws NoExiste  {...}

}
```

Se pide lo siguiente:

- Atributos y Constructor (0.5 puntos). Los atributos son:
  - `historia`: un array de objetos de la clase `DatoOrbital`
  - `actual`: un entero que indica el lugar de la tabla donde se guardará el próximo dato
  - `nombre`: un `String` con el nombre del satélite

  El constructor debe copiar el nombre en el atributo `nombre`, crear el array `historia` del tamaño indicado por `numDatos`, y poner `actual` a cero.
- Método `registra` (2 puntos). Este método llama a `leeDatoOrbital` para obtener un dato orbital que hay que guardar. Si se lanza `NoContesta` se reintenta una vez más la llamada a `leeDatoOrbital`. Si se vuelve a lanzar `NoContesta` se escribe en pantalla un mensaje de error y se finaliza el método. Si alguna de las llamadas a `leeDatoOrbital` tiene éxito, el dato obtenido se guarda en la casilla `actual` del array `historia`, y luego se incrementa `actual`, pero de manera que si su nuevo valor iguala el número de casillas del array, se pone a cero.
- Método `posMedia` (1.5 puntos). Este método debe retornar un objeto de la clase `DatoOrbital`, cuyo valor de *tiempo* sea cero, y cuyos valores x, y, z sean, respectivamente, iguales a la media de los valores x, y, z de los objetos almacenados en el array `historia` cuyo valor de tiempo esté comprendido entre `tMin` y `tMax`, ambos incluidos. Si no hubiese ningún valor que cumpla esa condición, el objeto retornado tendrá todos sus atributos iguales a cero.
- Método `pos` (1 puntos). Esté método debe retornar el primer objeto de la clase `DatoOrbital` almacenado en el array `historia` cuyo valor de *tiempo* sea igual a `t`. Si no se encontrase ningún objeto así, entonces debe lanzarse la excepción `NoExiste`.

Observar que no se pide el método `leeDatoOrbital`(); este método se usa desde `registra`().