Ejercicio de Feedback Final

Estructura de Datos

Enunciado

Ejercicio 1

Determine y justifique el orden de complejidad de los siguientes métodos:

```
public int metodo1(int x) {
   if (x < 0) {}
   x = -x;
   switch ((int)(Math.random() * 3)) {
      case 0:
         return 0;
      case 1:
         int ac = 1;
         for (int i = 1; i < x; i *= 2) {}
         ac = ac * 2;
         return ac;
      case 2:
         int i = x;
         ac = x;
         while (i > 1) {
            ac = x * i;
         }
         i = i / 2;
         return ac;
   }
}
```

```
public void metodo2(int x) {
   long y = x;
   for (int i = 1; i < x; i++) {
      for (int j = 1; j < x; j *= 2) {}
   }
   y = y * i;
   System.out.println(y);
}
```

Ejercicio 2

Se dispone de la siguiente secuencia de números:

1 0 6 4 8 2 4 7 7

Esta secuencia se va ordenar por el método de inserción directa

1. Escriba cómo queda el array después de cada fase de recorrido en que un dato más queda ordenado (no paso a paso).
2. En el método de ordenación quicksort existe un elemento privilegiado que se denomina pivote. ¿A qué se llama pivote y por qué es tan importante una elección apropiada del mismo?. Ponga un ejemplo en el que la elección del pivote genera complejidad $O(n^2)$.

Ejercicio 3

Para la aplicación de la Universidad se necesita un Comparador para publicar la información de alumnos ordenada de una determinada manera:

```
public class Alumno{
   String nombre;
   String apellidos;
   String provincia;
   int NP;
   // Otros datos del alumno y sus métodos
}
```

a) Se desea un método de comparación permita ordenar los alumnos de forma que se ordenen por provincia, para la misma provincia por apellidos y si coinciden por NP. Escriba un método de la siguiente formaç

```
boolean comparar(Alumno alum1, Alumno alum2)
```

Esta función devuelve true si alum1 va antes que alum2 y false en caso contrario.

b) ¿Qué método de ordenación es preferible y porqué desde el punto de vista de complejidad algorítmica: Inserción directa, Selección directa, Intercambio

directo, si el tiempo de copia se considera despreciable con respecto al tiempo de comparación?.

Ejercicio 4

Dada la siguiente clase Libro:

```
class Libro{
    String autor;
    String numRegistro; // De la forma 1234ABC
    int añoEdición;
    // resto de la clase
}
```

Escriba un método de nombre `librosPares` que tome como parámetro un `ArrayList` de Libros y devuelva un `ArrayList` con todos las libros cuyo nú- mero de registro (sin las letras) es un número par.

Ejercicio 5

Dada la siguiente clase Libro:

```
class Libro{
    String autor;
    String numRegistro; // De la forma 1234ABC
    int añoEdición;
    // resto de la clase
}
```

Por razones de eficiencia se ha pensado manejar todos los libros con un `TreeSet` para mantenerlos ordenados por año de edición y número de registro. Es decir, se ordenan por año de edición y, si hay dos libros con el mismo año de edición éstos se ordenan entre sí por el número de registro. Escriba las modificaciones que hay que realizar y los métodos que hay que añadir en la clase `Libro` indicada al inicio del examen para poder utilizarla junto con el `TreeSet`.

Ejercicio 6

Dada la siguiente clase Libro:

```
class Libro{
    String autor;
    String numRegistro; // De la forma 1234ABC
    int añoEdición;
    // resto de la clase
}
```

Escriba un método que dado un `TreeSet` de objetos `Libro` devuelva cual es el autor que más libros tiene escritos. Para resolver este ejercicio debe crear inicialmente una estructura donde estén todos los autores distintos que tengan libros publicados, y posteriormente calcule usando dicha lista cual es el autor con mayor número de libros publicados.

Ejercicio 7

Suponiendo que se dispone de las siguientes declaraciones de clases que per- miten manejar una lista de Jugadores:

```
public class Jugador {
   private String nombre;
   private String equipo;
   private String posición;
   private int sueldo;
   private int primas;
   public int getSueldo() {
      return sueldo;
   }
   public int getPrimas() {
      return primas;
   }
   public int getSueldoTotal() {
      return sueldo + primas;
   } // El resto de métodos para manejar el producto }
   public class NodoJugador {
      Jugador jugador;
      NodoJugador siguiente;
      // El resto de métodos como en la clase NodoSimple
   }
   public class ListaJugadores {
      NodoJugador cabecera
      // El resto de métodos como en la clase ListaSimple
   }
}
```

Donde la clase `ListaJugador` se codifica como una lista de `NodoJugador`, que guarda una referencia a un jugador. Se pide:

1. Escriba un método de nombre `totalSueldos` que calcule y devuelva la suma de todos los sueldos de todos los jugadores de la lista.

Ejercicio 8

Suponga que se realiza un árbol de búsqueda ordenado por nombre de equipo y nombre de jugador de la siguiente forma:

```
public class NodoJugador {
   Jugador jugador;
   NodoJugador izq;
   NodoJugador der;
   // El resto de métodos como en la clase NodoBúsqueda
}
public class ArbolBúsquedaJugador {
   NodoJugador raiz;
   // El resto de métodos como en la clase ArbolBúsqueda
}
```

1. Escriba un método en la clase `ArbolBúsquedaJugador` de nombre `imprimeSueldosAltos` que reciba por parámetro un valor de sueldo y escribe por pantalla el nombre y equipo de todos los jugadores cuyo sueldo completo el superior al valor dado.

Criterios de evaluación

Los criterios de corrección serán los siguientes:

Ejercicio	Porcentaje
Ejercicio 1	10%
Ejercicio 2	10%
Ejercicio 3	10%
Ejercicio 4	10%
Ejercicio 5	15%
Ejercicio 6	15%
Ejercicio 7	15%

Ejercicio 8	15%

OPENUAX

UNIVERSIDAD ALFONSO X EL SABIO

G R A C I A S

OPENUAX.COM