Examen de Estructuras de Datos y Algoritmos
(Ingeniería Informática)

Septiembre 2010

Primera parte (50% nota del examen)

1) Las ventas anuales de los diferentes productos ofertados en un supermercado se registran en objetos de la clase `VentasAnualesProducto`:

```
public class VentasAnualesProducto
                  implements Comparable<VentasAnualesProducto> {

    // atributos y otros métodos no relevantes para el problema

    /** Retorna el nombre del producto
     */
    public String producto() {...}

    /** Retorna el año de la venta
     */
    public int año()  {...}

    /** Retorna las unidades vendidas ese año
     */
    public int unidadesVendidas()  {...}

    /**  Compara en base al año (ordena de menor a mayor)
     */
    @Override
    public int compareTo(VentasAnualesProducto v)  {...}
}
```

Se pide implementar el método:

```
Map<String, Integer> ventasEnPeriodo(
        PriorityQueue<VentasAnualesProducto> ventas,
        int añoIni, int añoFin)
```

El método recibe una cola de prioridad con las ventas de los diferentes productos a lo largo de los años y los años de inicio y finalización del periodo en el que estamos interesados. Retorna un mapa con las ventas de los productos en el periodo comprendido entre el `añoIni` y `añoFin` (ambos inclusive). Las llaves del mapa serán los nombres de los productos vendidos en el periodo indicado y los valores el número de unidades vendidas de cada uno de esos productos en dicho periodo.

Tratar de que el método sea lo más eficiente posible.

¿Cual es la eficiencia del método en el peor caso?

2) Se dispone de una clase `Libro` que permite almacenar el título y los diferentes temas tratados en un libro:

```
public class Libro {
     // los atributos son públicos por simplicidad

     // título del libro
     public String título;

     // temas que trata el libro
     public Set<String> temas;
}
```

Se pide implementar la clase `ÍndiceTemático` que permite agrupar los libros por temas de forma que sea eficiente obtener la bibliografía que trata un determinado tema.

```
public class IndiceTematico {

     private Map<String, HashSet<Libro>> índice =
                        new HashMap<String, HashSet<Libro>>();

     /**
      * Inserta un libro en el índice añadiéndolo al Hashset
      *  de todos los temas que trata
      */
     public void inserta(Libro libro) {...}

     /**
      * Retorna un conjunto con todos los libros que tratan
      * el tema indicado
      */
     public Set<Libro> bibliografíaTema(String tema) {...}
}
```

Indicar la eficiencia de los dos métodos de la clase.

3) Escribir el pseudocódigo de un método al que se le pasa un grafo que utiliza la interfaz de grafos vista en clase y el contenido de un vértice inicial. El método debe retornar un conjunto que contenga los contenidos de todos los vértices alcanzables desde el vértice inicial (los vértices alcanzables son aquellos para los que existe un camino desde el vértice inicial). La cabecera del método será:

```
método <V,A> alcanzables(Grafo<V,A> g, V vInicial)
    retorna Set<V>
```

Examen de Estructuras de Datos y Algoritmos
(Ingeniería Informática)

Septiembre 2010

Segunda parte (50% nota del examen)

4) Se dispone de una implementación de un árbol genealógico realizada con cursores. La clase almacena un `ArrayList` de personas y cada persona tiene como atributos su nombre, y cursores a su padre y su madre (cada cursor indica la casilla del `ArrayList` donde se encuentra el padre o madre, respectivamente). Escribir un método con la cabecera que se indica abajo que retorne una lista de los hijos de una persona. Retornará `null` si la persona no existe en el árbol genealógico.

```
import java.util.*;
public class ArbolGenealogico
{
    // atributos del arbol genealógico
    ArrayList<Persona> lista=new ArrayList<Persona> ();

    // clase interna Persona
    private static class Persona {
        String nombre;
        int padre;
        int madre;
    }
    /**
     * Retorna los hijos de la persona llamada nombre
     */
    public List<String> hijos(String nombre) {...}

    ... resto de los métodos

}
```

5) Escribir el pseudocódigo de un método que permita en un árbol que sigue la interfaz de los árboles vista en clase retornar un conjunto que contenga todos los elementos almacenados en el árbol. La cabecera del método será:

```
método <E> obtenerTodos (Arbol<E> a) retorna Set<E>
```

6) Se dispone de la clase `ColaEnlazada` (vista en clase) que implementa una cola usando una lista enlazada simple. Los elementos se insertan por el final y se eliminan por el principio. Se muestra parte de la implementación abajo. Se pide implementar el método `removeLast()` que elimine el último elemento de la cola. El método tiene la cabecera que se muestra abajo. el método retorna el elemento borrado, o `null` si la cola está vacía. También se pide indicar la eficiencia de este método.

```
public class ColaEnlazada<E> extends AbstractQueue<E>
{

    // atributos de la cola
    private Celda<E> principio,fin;
    private int num;

    // clase privada que define la celda
    private static class Celda<E> {
        E contenido;
        Celda<E> siguiente;

        Celda(E cont) {
            contenido=cont;
        }
    }

    public E removeLast() {...}

   ... resto de los métodos
}
```