Programación I

Parcial (turno noche) – 13/06/2019

Ejercicio 1 *(20 pts. - 5 pts. cada item.)*

Discutir la veracidad de las siguientes afirmaciones, justificando su respuesta.

a) La invocación a un método de instancia se realiza igual a la invocación de un método de clase.
b) En el siguiente código se produce una excepción:

```
Point p;
Point q = new Point(2,8);
p = q;
q = null;
System.out.print(p.x);
```

c) La implementación de un TAD es transparente para el usuario del TAD.
d) El algoritmo de ordenamiento QuickSort tiene orden de complejidad en peor caso $O(n \log(n))$

Ejercicio 2 *(20 pts.)*

Escribir una función recursiva **boolean** estaIncluida(String s1, String s2) que toma dos String s1 y s2 y devuelve true si los caracteres de la cadena s1 están presentes en la cadena s2, respetando el orden de s1 y considerando las repeticiones. Por ejemplo:

- estaIncluida(''ala'', ''calidad'') debe devolver **true**.
- estaIncluida(''alla'', ''calidad'') debe devolver **false**.
- estaIncluida(''alla'', ''llave'') debe devolver **false**.
- estaIncluida("cajon'', "callejones'') debe devolver **true**.
- estaIncluida('''', ''palabra'') debe devolver **true**.
- estaIncluida(''ala'', ''almidon'') debe devolver **false**.

Se pide resolver **utilizando recursión**. Se pueden dar por hecha la función String resto(String s) que devuelve una cadena igual a s pero sin su primer carácter.

Ejercicio 3 *(35 pts. - 15 pts. item a.- y 20 pts. item b)*

Consideremos las clases Aeropuerto, Aerolinea, Avion, Vuelo y Pasajero definidas como:

```
public class Aeropuerto
{
    Aerolinea[] aerolineas;
    ...
}
public class Aerolinea
{
    String nombre;
    Avion[] aviones;
    ...
}
public class Avion{
    String modelo;
    int anioFabricacion;
    Vuelo[] vuelos;
    ...
}
```

```
public class Vuelo
{
    int nroVuelo;
    Pasajero[] pasajeros;
    Empleado[] empleados;
    String origen;
    String destino;
    int capacidad;
    ...
}
public class Pasajero
{
    int pasaporte;
    int edad;
    String nacionalidad;
}
```

Estas clases modelan un aeropuerto. Un Aeropuerto esta compuesto por un arreglo con aerolineas y cada aerolinea tiene un nombre y un arreglo con los aviones que forman parte de ella. A su vez, cada avion tiene su modelo, el año en que fue fabricado y los vuelos que realiza. Por último, el vuelo tiene los empleados que realizan el vuelos, los pasajeros que van en él, un origen, un destino y la capacidad máxima de pasajeros que puede alojar. Para la clase Aeropuerto, se pide:

a) Escribir un método **int** cantAvionesPasajesDeNac(String nacionalidad) que recibe una nacionalidad y devuelve la cantidad de aviones que trasportan pasajeros de esa nacionalidad en el Aeropuerto.
b) El aeropuerto realiza un control especial a los pasajeros de origen chino, mayores de edad que vuelen a Singapur. Para ello, se pide escribir un método ArrayList<Pasajero> chinosQueViajanASingapur() que devuelva una lista con los pasajeros que vuelen a Singapur, de nacionalidad China y mayores a 18 años de edad. La lista no debe contener elementos repetidos.
c) (*bonus track* 20 pts.) El aeropuerto va a lanzar un beneficio, regalándole un pasaje gratis por día al pasajero que más vuelos haya tomado. Para ello, se pide escribir un método Pasajero masFrecuente() que devuelva el Pasajero que aparezca en la mayor cantidad de vuelos en el Aeropuerto. En caso de haber más de un pasajero con estas características se puede devolver cualquiera de ellos.

Ejercicio 4 *(25 pts.)*

Dadas las clases NodoInt y ListaInt cuyas variables de instancia son las siguientes:

```
public class NodoInt
{
    int elemento;
    NodoInt siguiente;
}
```

```
public class ListaInt
{
    NodoInt primero;
    ...
}
```

Se pide escribir el método de instancia ListaInt extraerPares() que retorna una lista compuesta por aquellos nodos cuyo elemento son pares y modifica la lista original manteniendo únicamente los elementos impares. Por ejemplo:

- Si la lista es [2,5,4,3], la lista deber a quedar como [5,3] y debe devolver [2,4]
- Si la lista es [3,8,3], la lista deber a quedar como [3,3] y debe devolver [8]
- Si la lista es [], la lista deber a quedar como [] y debe devolver []
- Si la lista es [4], la lista deber a quedar como [4] y debe devolver []

Se pide además que el método implementado **sea de orden lineal**, es decir, $O(n)$ donde n es la cantidad de elementos de la lista. Justificar la complejidad del mismo.