Programación I

Recuperatorio Parcial (turno noche) – 28/11/2019

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

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

a) No es conveniente definir las variables de instancia privadas porque luego no es posible modificarlas desde fuera de la clase.
b) La captura de excepciones en Java se realiza utilizando la instrucción throw.
c) En el siguiente código se produce aliasing.

```
Point p;
Point q = new Point(6,2);
p = q;
```

d) Dados dos algoritmos que resuelven el mismo problema, elegimos entre ellos, el que tenga menor orden de complejidad en caso promedio.

Ejercicio 2 *(20 pts.)*

Escribir una función recursiva **int** cantApariciones(String s, **char** c) que dada una cadena s y un caracter c, devuelve la cantidad de veces que aparece c en s. Por ejemplo:

- cantApariciones(''anana'', 'a') debe devolver 3.
- cantApariciones(''elefante'', 'f') debe devolver 1.
- cantApariciones(''lamina'', 'z') debe devolver 0.
- cantApariciones('''', 'a') debe devolver 0'.
- cantApariciones(''camino'', 'c') debe devolver 1.

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 Modulo7, Laboratorio, PC, Marca y Componente definidas como:

```
public class Modulo7
{
    Laboratorio[] labos;
}

public class Laboratorio
{
    int numero;
    int capacidad;
    PC[] computadoras;
}

public class PC
{
    String serial;
    String modelo;
    String OS;
    Componente[] componentes;
}
```

```
public class Componente
{
    String nombre;
    String tipo;
    Marca marca;
}


public class Marca
{
    String nombre;
    float calidad;
}
```

Estas clases modelan los laboratorios que posee el Modulo7. Cada laboratorio posee computadoras que están compuestas por componentes. Cada Componente es una determinada marca que tiene un estandar de calidad, por lo cual según la marca y especificaciones se le asigna un valor entre 1 y 5 a la calidad del componente. Para la clase Modulo7, se pide:

a) Escribir un método **int** pcsConMinimaCalidad() que devuelva la cantidad de pcs que tienen todos sus componentes con calidad 1 en el módulo 7.
b) Escribir un método LinkedList<PC> pcCompuestasPor(Marca[] ciertasMarcas) que devuelva una lista con todas las PCs del módulo 7 donde todos sus componentes tengan su marca perteneciente al arreglo ciertasMarcas.
c) (*bonus track* 20 pts.) Escribir un método String marcaMasUsuadaMejorCalidad() que devuelva el nombre de la marca de mejor calidad que más componentes provee al módulo 7. Tener en cuenta que la mejor calidad es 5. En caso de tener más de una Marca con estas condiciones, se debe devolver el nombre de cualquiera de ellas.

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;
    ...
}
```

Para la clase ListaInt se pide escribir el método de instancia **void** agregarLargoMesetas() que modifica la lista agregando despues de cada meseta un nodo conteniendo el largo de la meseta. Una meseta es una subsecuencia contigua de elementos del mismo valor de longitud mayor o igual a 1. Por ejemplo:

- Si la lista es [1,1,1,12,3] debe quedar como [1,1,1,3,12,1,3,1]
- Si la lista es [8] debe quedar como [8,1]
- Si la lista es [2,10,12] debe quedar como [2,1,10,1,12,1]
- Si la lista es [1,1,3,7,7,3] debe quedar como [1,1,2,3,1,7,7,2,3,1]
- Si la lista es [ ] debe quedar como [ ]
- Si la lista es [5,5,3,5] debe quedar como [5,5,2,3,1,5,1]

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.