Algoritmos y Estructuras de Datos: Examen Julio (2º parcial)

Departamento de Lenguajes, Sistemas Informáticos e Ingeniería de Software

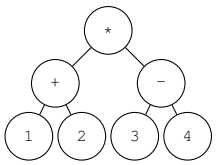
Grado en Ingeniería Informática, Grado en Matemáticas e Informática y
Doble Grado en Ingeniería Informática y Administración y Dirección de Empresas

- Este examen dura **75 minutos** y consta de **?? preguntas** que puntúan hasta **?? puntos**.
- Todas las hojas entregadas deben indicar, en la parte superior de la hoja, **apellidos**, **nombre**, **DNI/NIE** y **número de matrícula**.
- Las calificaciones provisionales de este examen se publicarán el **13 de Julio de 2018** en el Moodle de la asignatura junto con la fecha y lugar de la revisión.

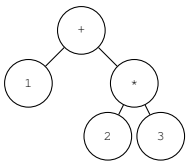
(3½ puntos) 1. **Se pide:** Implementar en Java el método

```
public static String toStringExp (BinaryTree<String> tree)
```

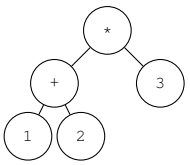
que toma como parámetro un árbol binario `tree` y devuelve un `String` con la expresión matemática en notación *infija* almacenada en el árbol. Será necesario poner paréntesis en cada subexpresión para evitar posibles problemas con la precedencia de operadores. Cada nodo del árbol contendrá un elemento de la expresión en forma de `String`, ya sea operador u operando. El árbol tree no será `null` ni podrá contener elementos `null`. Se asume que tree contendrá expresiones matemáticas correctas y todos los nodos del árbol que contengan un operador tendrán dos hijos. **Se recomienda** implementar el método de **forma recursiva**. A continuación se muestran algunos ejemplos de árboles y los resultados devueltos por el método `toStringExp`:



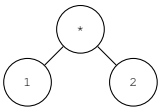
`"((1+2)*(3-4))"`



`"(1+(2*3))"`



`"((1+2)*3)"`



`"(1*2)"`

(3 puntos) 2. Dada la clase

```
class Alumno {
  private Integer dni;
  private String apellidos;
  public Alumno(Integer dni, String apellidos) {
    this.dni = dni;
    this.apellidos = apellidos;
  }
  public Integer getDni() { return dni; }
  public String getApellidos() { return apellidos; }
  public String toString () {return dni + ":" + apellidos;}
}
```

Se pide: Implementar en Java el método

```
public static void imprimirOrdenados(PositionList<Alumno> lista)
```

que imprime en orden ascendiente por DNI los alumnos recibidos en `lista`. La `lista` no contendra elementos `null` ni ninguno de los atributos del alumno serán `null`. Se recomienda utilizar una cola con prioridad usando el interfaz `PriorityQueue<K,V>` para ordenar los elementos. Puede asumirse que se dispone de la clase `SortedListPriorityQueue<K,V>`, que implementa el interfaz `PriorityQueue<K,V>` y que cuenta con el constructor `SortedListPriorityQueue<K,V>()` para crear una cola con prioridad vacía. Nótese que la clase `Integer`, usada para almacenar el DNI, implementa el interfaz `Comparable<Integer>`.

Por ejemplo, dada `lista=[Alumno(4,"a1"),Alumno(2,"a2"),Alumno(0,"a3")]`, la salida por consola será `0:a3 2:a2 4:a1`.

(3½ puntos) 3. Dada la clase `Alumno` de la pregunta anterior, se dispone de un listado en el que están mezcladas todas las entregas de todos los alumnos que han entregado las prácticas individuales. En dicho listado un mismo alumno podrá aparecer múltiples veces, tantas como prácticas haya entregado.

Se pide: Implementar en Java el método

```
public Map<Integer,Integer> contarPracticas (PositionList<Alumno> lentregas)
```

que recibe una lista con las entregas de los alumnos y devuelve un objeto de tipo `Map<Integer,Integer>` cuya clave será el DNI del alumno y cuyo valor será el número de entregas realizadas por dicho alumno. La lista `lentregas` no contendrá elementos `null` ni los atributos de los alumnos contenidos en la lista serán `null`. Se dispone de la clase `HashTableMap<K,V>` que implementa el interfaz `Map<K,V>` y que cuenta con el constructor `HashTableMap<K,V>()` para crear un `Map` vacío.

Por ejemplo, dada `lista=[Alumno(4,"a"),Alumno(2,"b"), Alumno(0,"c"),Alumno(4,"a")]`, el método `contarPracticas(lista)` devolverá un `Map` con los siguientes pares ⟨clave,valor⟩: $\langle 2,1\rangle, \langle 0,1\rangle, \langle 4,2\rangle$, que indica que el alumno con DNI 2 ha entregado 1 práctica, que el alumno con DNI 0 ha entregado 1 práctica y que el alumno con DNI 4 ha entregado 2 prácticas.