Estructuras de Datos Avanzadas
Examen Parcial
16 de noviembre de 2021

- La duración del examen es de 45 minutos.
- Desde el campus virtual hay que descargar un archivo ZIP que contiene el esqueleto de los dos ejercicios del examen así como los test unitarios correspondientes. Se recomienda crear un proyecto de Netbeans vacío; copiar dentro de la carpeta del proyecto la carpeta src y la carpeta test proporcionadas; añadir las librerías JUnit y Hamcrest; y finalmente cerrar y volver a abrir Netbeans.
- El alumno debe generar un fichero ZIP con el directorio en el que esté el código del alumno y subirá el fichero ZIP a la actividad examen práctico parcial del aula virtual asegurándose de que entrega el código desarrollado durante la prueba.

Ejercicio 1 [3 puntos]

Se desea aumentar la funcionalidad de `LinkedBinaryTree` incorporando la siguiente funcionalidad:

a)[1,5 punto] Se dice que un árbol binario es perfecto si todos los nodos internos tienen dos hijos. Implementar un método `isPerfect`, que determine si un árbol binario es perfecto o no. (si es vacío es perfecto)

b)[1,5 punto] `InternalNodeIterator`: clase que permite recorrer los nodos internos de un árbol binario.

NOTA: Se valorará un buen diseño (de clases y métodos) para obtener la máxima calificación. La cabecera del método `isPerfect` está en el fichero "`MoreFunctionality.java`". Y la clase `InternalNodeIterator` está en el mismo paquete. No está permitido la modificación de las cabeceras de los métodos a implementar, pero sí añadir métodos auxiliares.

Ejercicio 2 [4 puntos]

Se desea desarrollar un sistema de gestión que permita mantener la información de los estudiantes de una escuela. La información que se desea almacenar de cada estudiante es su nombre, su primer apellido y su n.º de DNI. Además cada estudiante podrá estar matriculado en diversas materias. La información de cada materia será únicamente el nombre. El sistema de organización debe ser lo más eficiente posible, a la hora de localizar las asignaturas en las que esté matriculado un estudiante (O(1)). Se pide:

a) [1 punto] Implementar dos constructores de la clase `Organiser`, el primero que genere una estructura vacía y el segundo que reciba una lista de pares asignaturas, estudiantes. Además se deben definir las propiedades de la clase.

b) [0,5 puntos] Implementar la operación `enrolledSubjects`, que recibe un objeto de tipo `DNI` y devuelve la lista de materias en las que esté matriculado el estudiante. En caso de no existir un estudiante con ese `DNI` devuelve `null`. El coste de esta operación debe ser O(1).

c) [1 punto] Implementar la operación `newStudent`, que recibe un objeto de tipo `estudiante` y una lista de asignaturas en las que se desea matricular el estudiante. En caso de que el estudiante ya estuviese matriculado, se tratará como una modificación de matrícula.

d) [1 punto] Implementar la operación `registrationChange`, que recibe un objeto de tipo `estudiante` y una lista de asignaturas en las que se desea matricular el estudiante. En esta ocasión el estudiante ya debe estar matriculado previamente. En caso de no estarlo el sistema debe lanzar un mensaje de error. La nueva lista de asignaturas sustituirá a la antigua.

e) [0,5 puntos] Implementar el método `studentData`, que recibe un objeto de tipo `DNI` y devuelve un `String` con los datos personales del estudiante: nombre, apellido y dni. En caso de no existir un estudiante con ese `DNI` devuelve `null`. El coste de esta operación debe ser O(1).

NOTA: Se valorará un buen diseño (de clases y métodos) para obtener la máxima calificación. La clase `Organiser` está en la carpeta examen parcial. No está permitido la modificación de las cabeceras de los métodos a implementar, pero sí añadir métodos auxiliares.