Examen de repaso 5

Ejercicio 1

Implementar la clase **Estudiante** la cual está compuesta por un campo de tipo cadena que representa el nombre, un campo de tipo entero que representa la edad y un campo de tipo cadena que representa su dni.

Se debe codificar el **constructor vacío**, donde el nombre tomará el valor por defecto "Sin nombre", la edad cero y el dni "Sin dni". **Constructor parametrizado** cuyo único parámetro obligatorio será el dni, siendo el resto opcional y tomando como valores por defecto los indicados anteriormente. Se debe controlar que, si alguna cadena recibida está vacía, se asignará el valor por defecto y no cadena vacía, mientras que, si la edad recibida es menor que cero se asignará cero, es decir, el valor por defecto.

Implementar además los observadores y modificadores:

- **getNombre(), getEdad() y getDNI()**
- **setNombre(), setEdad() y setDNI()**

Nota: l*os modificadores deben ser booleanos y retornar true o false dependiendo de si han modificado la variable privada o no. Si las cadenas son vacías o si la variable entera es menor que cero, la variable privada no se modificará y se retornará false. En caso contrario, se modificará y se devolverá true.*

Por último, implementar un método **getInfo()** que retorne una cadena de caracteres con el siguiente formato:

"El alumno con nombre XXX tiene YYY años y su DNI es ZZZ."

Donde XXX corresponde al nombre, YYY a la edad y ZZZ a su dni.

Ejercicio 2

Implementar la clase **EstudianteGraduado** que deriva de la clase **Estudiante**. Esta clase dispondrá además de un campo de tipo cadena que representa la especialidad y un campo de tipo entero que representa el año de finalización o graduación.

Debe implementar un **constructor vacío**, el cuál establecerá la especialidad por defecto a "Sin indicar" y el año 2011.

Constructor parametrizado que recibirá como parámetros obligatorios el dni del estudiante, la especialidad y el año de graduación. El resto de los valores se tomarán por defecto los indicados anteriormente.

Observador **getEspecialidad()** y **getAnioGraduacion(),** ambos por referencia.

Modificadores **setEspecialidad()** y **setAnioGraduacion().** Ambos serán de tipo booleano y controlará que el año sea superior al 2011 y que la especialidad sea "Mención en Computación", "Mención en Computadores" o "Mención en Software". En caso de recibir un valor permitido, este se asignará a la variable privada y se retornará true, en caso contrario, no se asignará y se retornará false.

Nota: *en el constructor parametrizado, si no se recibe como valores de especialidad o año de graduación los permitidos e indicados en los modificadores, estos tomarán por defecto los valores indicados en el constructor vacío.*

Ejercicio 3

Implemente, en los ficheros matriculaAsignatura.cc y matriculaAsignatura.h, la clase **MatriculaAsignatura**. Esta clase define todos los alumnos que han sido matriculados en una determinada asignatura. La clase está definida por una cadena que representa el nombre de la asignatura, un número máximo de alumnos que se pueden matricular en ella y una lista de la STL formada por los estudiantes matriculados en dicha clase.

Codificar el **constructor parametrizado**, el cuál reciba como únicos parámetros obligatorios el nombre de la asignatura y el número máximo de alumnos que se pueden matricular en dicha asignatura.

Observador **getAsignatura()** que retorna el nombre de la asignatura y **getMaxAlumnos()** que retorna el número máximo de alumnos.

Método **addAlumno(),** que recibe un alumno y lo añade a la lista siempre y cuando este no se encuentre ya en ella. Si se añade el método retornará true, en caso contrario false.

Método **getN()**, que retorna el número de alumnos matriculados en la asignatura.