Metodologías de Programación I

Práctica 4.

Patrones Adapter y Decorator

Ejercicio 1

Modifique la clase Alumno de prácticas anteriores para agregarle una variable de instancia denominada *calificación* con sus correspondientes *getter* y *setter* que representa la última calificación obtenida.

Agregue a la clase los métodos:

```
int responderPregunta(int pregunta)
                    que devuelve un número al azar entre 1
                    y 3 como respuesta (simulando un
                    multiple choice)
string mostrarCalificación()
                    que devuelve un string con el nombre y
                    apellido del alumno junto con la última
                    calificación obtenida.
                    Por ejemplo:
                      Ratón Pérez          6
```

Ejercicio 2

Haga una subclase de *Alumno*, llamada *AlumnoMuyEstudioso* que reimplemente el método *responderPregunta* y que devuelva la respuesta correcta a la pregunta formulada:

```
int responderPregunta(int pregunta)
     return pregunta módulo 3
```

Ejercicio 3

El sistema MDPI, provisto por la cátedra, simula una clase de Metodologías de Programación I, donde un profesor (clase *Teacher*) da una clase (método *teachingAClass*) la cual consiste en pasar lista a los alumnos, tomarles un examen y publicar los resultados ordenados por la calificación obtenida.

Implemente con el patrón Adapter, un adaptador para la clase *Alumno* desarrollada en el ejercicio 1 y que pueda comportarse como un *Student*.

Ejercicio 4

Implemente una función *main* que instancie un *Teacher*, haga llegar a 20 *Students* con el método *goToClass*. Los *Student* son instancias del adaptador implementado en el ejercicio anterior, 10 de ellos adaptaran a una instancia de la clase *Alumno*, mientras que los otros 10 adaptarán a una instancia de *AlumnoMuyEstudioso*.

Luego invoque al método *teachingAClass* de *Teacher* para comprobar el correcto funcionamiento del sistema.

Ejercicio 5

Opcional: Implemente una estrategia para que el listado de calificaciones quede ordenado de mayor a menor por la última calificación.

Ejercicio 6

Implemente con el patrón Decorator distintos decorados para mostrar las calificaciones de los alumnos.

- Un decorado que imprima el legajo:
  Ratón Pérez (12345/6) 6
- Un decorado que imprima la nota en letras:
  Ratón Pérez 6 (SEIS)
- Un decorado que imprima PROMOCIÓN (si la nota es mayor o igual 7), APROBADO (si la nota está entre 4 y 7 ó DESAPROBADO (si la nota es menor a 4):
  Ratón Pérez 6 (APROBADO)
- Un decorado que imprima el resultado dentro de un recuadro con asteriscos:
  ****
  * Ratón Pérez 6 *
  ****
- (Opcional) Un decorado que imprima el número de orden secuencial dentro del listado:
  5) Ratón Pérez 6

Ejercicio 7

Arme los *Students* con todos los decorados implementado en el ejercicio anterior para que se impriman de la siguiente manera:

****
* 5) Ratón Pérez (12345/6) 6 (SEIS) (APROBADO) *
****

Opcional: Haga una subclase de *FabricaDeAlumnos*, llamada *StudentsFactory* que instancie los alumnos decorados y adaptados.

Ejercicio 8

Ejecute la función *main* del ejercicio 4 y compruebe la correcta impresión del listado de calificaciones.

Este ejercicio, y todos los anteriores que dependen de éste, debe ser entregado en el aula virtual del campus.

Ejercicio 9

Opcional. Implemente un adaptador de *Iterable* (ejercicio 6 de la práctica 2) a *Collection*. Implemente un adaptador de *Iterador* (ejercicio 7 de la práctica 2) a *IteratorOfStudent*.

Modifique el *main* del ejercicio 4 e instancie un *Coleccionable* cualquiera (*Pila*, *Cola*, *Conjunto* o *Diccionario*) y, usando el adaptador correspondiente, seteelo a *Teacher* con el método *setStudents*. Luego agregue los *Alumnos* y *AlumnosMuyEstudiosos* con el método goToClass y finalmente invoque al método *teachingAClass*.

Nota: para implementar el método *sort* de la interface *Collection* puede copiar el código dado por la cátedra (método *sort* de la clase *ListOfStudent*).

Ejercicio 10

Opcional. Intercambie las clases de decoradores implementadas en esta práctica con otro compañero para probar si funcionan clases “externas” en el sistema desarrollado por uno mismo.