# Metodologías de Programación I

*Práctica 6.*

*Patrones Composite y Template Method*

## Ejercicio 1

Implemente con el patrón Composite un compuesto de *Alumno*. Este compuesto debe tener el siguiente comportamiento:

- *getNombre*: devuelve el nombre y apellido de todos los componentes-hijos del compuesto.
- *responderPregunta*: devuelve la respuesta más votada por todos los componentes-hijos del compuesto (en caso de empate entre dos o más respuestas, se elige una de ellas al azar).
- *setCalificacion*: le setea la calificación a todos los componentes-hijos del compuesto.
- *mostrarCalificacion*: le envía el mismo mensaje a todos los componentes-hijos del compuesto.
- *sosIgual*: devuelve true si existe un componente-hijo que sea igual al *Alumno* recibido por parámetro.
- *sosMenor*: devuelve true si todos los componentes-hijos son más chicos que el *Alumno* recibido por parámetro.
- *sosMayor*: devuelve true si todos los componentes-hijos son más grandes que el *Alumno* recibido por parámetro.

## Ejercicio 2

Modifique la función *main* del ejercicio 2 de la práctica 5 agregando al aula un alumno compuesto con cinco *Alumno* proxies cualesquiera. ¿Qué objeto necesita ser adaptado en este ejercicio?

Arme una nueva fábrica de alumnos-compuestos para la creación del compuesto correspondiente.

Ejecute ésta función para comprobar el correcto funcionamiento del patrón.

## Ejercicio 3

Implemente con Template Method el algoritmo para jugar a cualquier juego de cartas sabiendo que en cualquier juego de cartas contiene los siguientes "pasos":

- *mezclar el mazo*
- *repartir las cartas iniciales*
- *jugar una mano: que consiste, para cada jugador:*
  - *tomar cartas*
  - *descartar cartas*
- *chequear si existe un ganador*

Implemente esta plantilla en una función que reciba dos *Persona* y devuelva una *Persona* (la que resulte ganadora)

## Ejercicio 4

Implemente dos juegos de cartas cualesquiera, los que el *Alumno* desee, donde en cada paso del algoritmo se imprimen textos en consola.

*Aclaración*: piense en algún mecanismo de simulación que permita jugar, al menos, un par de manos (según el juego implementado) que permita devolver la *Persona* ganadora.

## Ejercicio 5

Implemente en la clase *Program* una función *main* que permita instanciar un juego de cartas y hacer jugar a dos *Persona*. Imprima el nombre de la *Persona* ganadora.

## Ejercicio 6

Realice la siguiente modificación. Haga jugar a las personas todas las partidas que sean necesarias hasta que una obtenga una cierta cantidad de puntos, convirtiéndola en la ganadora de la partida.

¿Qué tuvo que modificar para llevar a cabo esta modificación?

*Este ejercicio, y todos los anteriores que dependen de éste, deben ser entregados en el aula virtual del campus.*

## Ejercicio 7

*Para reflexionar*: ¿qué debería modificar para permitir juegos de más de dos jugadores?

*Opcional*: Implemente la modificación sugerida.

## Ejercicio 8

*Para reflexionar*: Con qué patrón de diseño ya estudiado se podría simular un torneo de cartas entre 16 jugadores enfrentados de a dos. (octavos, cuartos, semifinal y final).

*Opcional*: Implemente la idea planteada.

## Ejercicio 9

*Para reflexionar*: ¿Qué cambia del diseño anterior que el juego de cartas sea uno contra uno, o que en cada partida se enfrenten más de dos jugadores?

*Opcional*: Implemente la idea planteada.

## Ejercicio 10

*Opcional*. Intercambie las clases compuestos, plantillas y juegos concretos implementadas en esta práctica con otro compañero para probar si funcionan clases "externas" en el sistema desarrollado por uno mismo.