Trabajo Práctico N° 10:
Módulo Objetos (Repaso).

Ejercicio 1.

La UNLP desea administrar sus proyectos, investigadores y subsidios. Un proyecto tiene: nombre, código, nombre completo del director y los investigadores que participan en el proyecto (50 como máximo). De cada investigador, se tiene: nombre completo, categoría (1 a 5) y especialidad. Además, cualquier investigador puede pedir hasta un máximo de 5 subsidios. De cada subsidio, se conoce: el monto pedido, el motivo y si fue otorgado o no.

(a) *Implementar el modelo de clases teniendo en cuenta:*

- *Un proyecto sólo debería poder construirse con el nombre, código, nombre del director.*
- *Un investigador sólo debería poder construirse con nombre, categoría y especialidad.*
- *Un subsidio sólo debería poder construirse con el monto pedido y el motivo. Un subsidio siempre se crea en estado no-otorgado.*

(b) *Implementar los métodos necesarios (en las clases donde corresponda) que permitan:*

- *void agregarInvestigador(Investigador unInvestigador);*
  // agregar un investigador al proyecto.
- *void agregarSubsidio(Subsidio unSubsidio);*
  // agregar un subsidio al investigador.
- *double dineroTotalOtorgado();*
  // devolver el monto total otorgado en subsidios del proyecto (tener en cuenta todos los subsidios otorgados de todos los investigadores).
- *void otorgarTodos(String nombre_completo);*
  // otorgar todos los subsidios no-otorgados del investigador llamado nombre_completo.
- *String toString();*
  // devolver un string con: nombre del proyecto, código, nombre del director, el total de dinero otorgado del proyecto y la siguiente información de cada investigador: nombre, categoría, especialidad y el total de dinero de sus subsidios otorgados.

(c) *Escribir un programa que instancie un proyecto con tres investigadores. Agregar dos subsidios a cada investigador y otorgar los subsidios de uno de ellos. Luego, imprimir todos los datos del proyecto en pantalla.*

Ver proyecto “TP10_E1” de Java.

Ejercicio 2.

Se quiere un sistema para gestionar estacionamientos. Un estacionamiento conoce su nombre, dirección, hora de apertura, hora de cierre y almacena, para cada número de piso (1..N) y número de plaza (1..M), el auto que ocupa dicho lugar. De los autos, se conoce nombre del dueño y patente.

(a) *Generar las clases, incluyendo getters y setters adecuados.*

(b) *Implementar constructores. En particular, para el estacionamiento:*

- *Un constructor debe recibir nombre y dirección e iniciar el estacionamiento con hora de apertura "8:00", hora de cierre "21:00" y para 5 pisos y 10 plazas por piso. El estacionamiento inicialmente no tiene autos.*
- *Otro constructor debe recibir nombre, dirección, hora de apertura, hora de cierre, el número de pisos (N) y el número de plazas por piso (M) e iniciar el estacionamiento con los datos recibidos y sin autos.*

(c) *Implementar métodos para:*

- *Dado un auto A, un número de piso X y un número de plaza Y, registrar al auto en el estacionamiento en el lugar (X, Y). Suponer que (X, Y) son válidos (es decir, están en rango 1..N y 1..M, respectivamente) y que el lugar está desocupado.*
- *Dada una patente, obtener un String que contenga el número de piso y plaza donde está dicho auto en el estacionamiento. En caso de no encontrarse, retornar el mensaje "Auto Inexistente".*
- *Obtener un String con la representación del estacionamiento.*
  Ejemplo: "Piso 1 Plaza 1: libre; Piso 1 Plaza 2: representación del auto; ...; Piso 2 Plaza 1: libre; ..."
- *Dado un número de plaza Y, obtener la cantidad de autos ubicados en dicha plaza (teniendo en cuenta todos los pisos).*

(d) *Realizar un programa que instancie un estacionamiento con 3 pisos y 3 plazas por piso. Registrar 6 autos en el estacionamiento en distintos lugares. Mostrar la representación String del estacionamiento en consola. Mostrar la cantidad de autos ubicados en la plaza 1. Leer una patente por teclado e informar si dicho auto se encuentra en el estacionamiento o no. En caso de encontrarse, la información a imprimir es el piso y plaza que ocupa.*

Ver proyecto "TP10_E2" de Java.

Ejercicio 3.

Un productor musical desea administrar los recitales que organiza, que pueden ser: eventos ocasionales y giras.

- *De todo recital, se conoce el nombre de la banda y la lista de temas que tocarán durante el recital.*
- *Un evento ocasional es un recital que, además, tiene el motivo (a beneficio, show de TV o show privado), el nombre del contratante del recital y el día del evento.*
- *Una gira es un recital que, además, tiene un nombre y las "fechas" donde se repetirá la actuación. De cada "fecha", se conoce la ciudad y el día. Además, la gira guarda el número de la fecha en la que se tocará próximamente (actual).*

(a) *Generar las clases necesarias. Implementar métodos getters/setters adecuados.*

(b) *Implementar los constructores. El constructor de recitales recibe el nombre de la banda y la cantidad de temas que tendrá el recital. El constructor de eventos ocasionales, además, recibe el motivo, el nombre del contratante y día del evento. El constructor de giras, además, recibe el nombre de la gira y la cantidad de fechas que tendrá.*

(c) *Implementar los métodos listados a continuación:*

(i) *Cualquier recital debe saber responder a los mensajes:*
- ***agregarTema*** *que recibe el nombre de un tema y lo agrega a la lista de temas.*
- ***actuar*** *que imprime (por consola), para cada tema, la leyenda "y ahora tocaremos…" seguido por el nombre del tema.*

(ii) *La gira debe saber responder a los mensajes:*
- ***agregarFecha*** *que recibe una "fecha" y la agrega adecuadamente.*
- *La gira debe responder al mensaje* ***actuar*** *de manera distinta. Imprimir la leyenda "Buenas noches…" seguido del nombre de la ciudad de la fecha "actual". Luego, debe imprimir el listado de temas como lo hace cualquier recital. Además, debe establecer la siguiente fecha de la gira como la nueva "actual".*

(iii) *El evento ocasional debe saber responder al mensaje actuar de manera distinta:*
- *Si es un show de beneficencia, se imprime la leyenda "Recuerden colaborar con…" seguido del nombre del contratante.*
- *Si es un show de TV, se imprime "Saludos amigos televidentes".*
- *Si es un show privado, se imprime "Un feliz cumpleaños para…" seguido del nombre del contratante.*

Independientemente del motivo del evento, luego, se imprime el listado de temas como lo hace cualquier recital.

(iv) *Todo recital debe saber responder al mensaje* ***calcularCosto*** *teniendo en cuenta lo siguiente. Si es un evento ocasional, devuelve 0 si es a beneficio, 50.000*

si es un show de TV y 150.000 si es privado. Las giras deben devolver 30.000 por cada fecha de la misma.

(d) *Realizar un programa que instancie un evento ocasional y una gira, cargando la información necesaria. Luego, para ambos, imprimir el costo e invocar al mensaje "actuar".*

Ver proyecto "TP10_E3" de Java.

Ejercicio 4.

Una escuela de música arma coros para participar de ciertos eventos. Los coros poseen un nombre y están formados por un director y una serie de coristas. Del director, se conoce el nombre, DNI, edad y la antigüedad (un número entero). De los coristas, se conoce el nombre, DNI, edad y el tono fundamental (un número entero). Asimismo, hay dos tipos de coros: coro semicircular en el que los coristas se colocan en el escenario uno al lado del otro y coro por hileras donde los coristas se organizan en filas de igual dimensión.

(a) *Implementar las clases necesarias teniendo en cuenta que los coros deberían crearse con un director y sin ningún corista, pero sí sabiendo las dimensiones del coro.*

(b) *Implementar métodos (en las clases donde corresponda) que permitan:*

- *agregar un corista al coro.*
  - *En el coro semicircular, los coristas se deben ir agregando de izquierda a derecha.*
  - *En el coro por hileras, los coristas se deben ir agregando de izquierda a derecha, completando la hilera antes de pasar a la siguiente.*
- *determinar si un coro está lleno o no. Devuelve true si el coro tiene a todos sus coristas asignados o false en caso contrario.*
- *determinar si un coro (se supone que está lleno) está bien formado. Un coro está bien formado si:*
  - *En el caso del coro semicircular, de izquierda a derecha, los coristas están ordenados de mayor a menor en cuanto a tono fundamental.*
  - *En el caso del coro por hileras, todos los miembros de una misma hilera tienen el mismo tono fundamental y, además, todos los primeros miembros de cada hilera están ordenados de mayor a menor en cuanto a tono fundamental.*
- *devolver la representación de un coro formada por el nombre del coro, todos los datos del director y todos los datos de todos los coristas.*

(c) *Escribir un programa que instancie un coro de cada tipo. Leer o bien la cantidad de coristas (en el caso del coro semicircular) o la cantidad de hileras e integrantes por hilera (en el caso del coro por hileras). Luego, crear la cantidad de coristas necesarios, leyendo sus datos y almacenándolos en el coro. Finalmente, imprimir toda la información de los coros indicando si están bien formados o no.*

Ver proyecto “TP10_E4” de Java.