Módulo Objetos
Repaso

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.

i) Implemente el modelo de clases teniendo en cuenta:
   a. Un proyecto sólo debería poder construirse con el nombre, código, nombre del director.
   b. Un investigador sólo debería poder construirse con nombre, categoría y especialidad.
   c. Un subsidio sólo debería poder construirse con el monto pedido y el motivo. Un subsidio siempre se crea en estado no-otorgado.

ii) Implemente los métodos necesarios (en las clases donde corresponda) que permitan:
   a. void agregarInvestigador(Investigador unInvestigador);
      // agregar un investigador al proyecto.
   b. void agregarSubsidio(Subsidio unSubsidio);
      // agregar un subsidio al investigador.
   c. double dineroTotalOtorgado();
      //devolver el monto total otorgado en subsidios del proyecto (tener en cuenta todos los subsidios *otorgados* de todos los investigadores)
   d. void otorgarTodos(String nombre_completo);
      //otorgar todos los subsidios no-otorgados del investigador llamado *nombre_completo*
   e. 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.

iii) Escriba un programa que instancie un proyecto con tres investigadores. Agregue dos subsidios a cada investigador y otorgue los subsidios de uno de ellos. Luego imprima todos los datos del proyecto en pantalla.

2- Se necesita diseñar 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: el nombre del dueño y patente.

a) Genere las clases, incluyendo getters y setters adecuados.

b) Implemente 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) Implemente 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. Suponga 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* … etc”
- Dado un número de plaza Y, obtener la cantidad de autos ubicados en dicha plaza (teniendo en cuenta todos los pisos).

d) Realice un programa que instancie un estacionamiento con 3 pisos y 3 plazas por piso.
Registre 6 autos en el estacionamiento en distintos lugares.
Muestre la representación String del estacionamiento en consola.
Muestre la cantidad de autos ubicados en la plaza 1.
Lea una patente por teclado e informe 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.

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 o show de TV), 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) Genere las clases necesarias. Implemente métodos getters/setters adecuados.

b) Implemente 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) Implemente 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.
- Devolver el string para cada tema con 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.
- Devuelve el string de manera distinta: Imprime 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 devolver el string de manera distinta:

- Si es un show de beneficencia devuelve la leyenda "Recuerden colaborar con…" seguido del nombre del contratante.
- Si es un show de TV devuelve "Saludos amigos televidentes"

Independientemente del motivo del evento, luego se devuelve 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 y 50000 si es un show de TV. Las giras deben devolver 30000 por cada fecha de la misma.

d) Realice un programa que instancie un evento ocasional y una gira, cargando la información necesaria. Luego, para ambos, imprima el costo y su información correspondiente.

4- Una aerolínea registra las reservas de pasajes de sus clientes con la siguiente información: número de reserva, aeropuerto de origen y aeropuerto destino del vuelo, fecha de vuelo y el importe a abonar.

Existen diferentes tipos de reservas:

- La **reserva clásica**, que consiste en la reserva de un boleto de avión para un único pasajero. Por lo que se asocia a este tipo de reserva un número de DNI del pasajero y un número de asiento.
- La **reserva grupal**, que consiste en aquellas que involucran un grupo de viajeros. Por lo que, tienen asociado un conjunto de DNIs de los pasajeros que tomarán el vuelo (a lo sumo 30).

Se pide:

a. Implementar la clase **Reserva** con sus respectivos atributos, constructores y métodos para acceder y modificar sus atributos. Además de todos los elementos necesarios para manejar los diferentes tipos de reserva.
b. La aerolínea quiere establecer una promoción, que será aplicada de la siguiente manera:

   - En el caso de las **reservas clásicas**, se aplicará un descuento del 10%.
   - En el caso de las **reservas grupales**, se aplicará un descuento del 5% si la reserva no supera los 6 pasajeros o de la bonificación de un pasaje gratis cada 10 pasajeros.

c. Implementar los métodos necesarios para imprimir toda la información de las reservas.

d. Implementar un programa principal que cargue dos **reservas grupales** (una con 5 pasajeros y una con 10 pasajeros) y una **reserva clásica**. Después de la carga, aplicar el descuento y hacer la impresión de los datos utilizando lo implementado en el inciso c).