Universidad Provincial del Sudoeste
Promoviendo el Desarrollo Armónico de la Región

SEGUNDO PARCIAL

Asignatura: Programación Orientada a Objetos

ATENCIÓN: LA EVALUACIÓN TIENE EN CUENTA LAS CONVENCIONES UTILIZADAS EN LA MATERIA Y LAS TÉCNICAS DE IMPLEMENTACIÓN PRESENTADAS EN CLASE.

Ejercicio 1: Excepciones

a) Implementar una clase ListaDeReproduccion, que contenga los atributos _genero, _listaTemas y _cantidadMaximaTemas. Al crear una instancia, el atributo _listaTemas debe establecerse por defecto en una lista vacía y el atributo _cantidadMaximaTemas tiene valor por defecto 5.

b) Luego implementar el método agregarTema, que recibe los parámetros nombre y género como información del tema a agregar en la lista de reproducción. Este método debe hacer uso de excepciones para las siguientes situaciones:

- El género y el nombre del tema deben ser siempre del tipo cadena de caracteres, en caso contrario se lanza una excepción TypeError.

- No se puede agregar un tema con el mismo nombre que otro ya existente en la lista, en caso contrario se lanza una excepción ValueError.

- El tema a agregar debe ser del mismo género que la lista, en caso contrario se lanza una excepción ValueError.

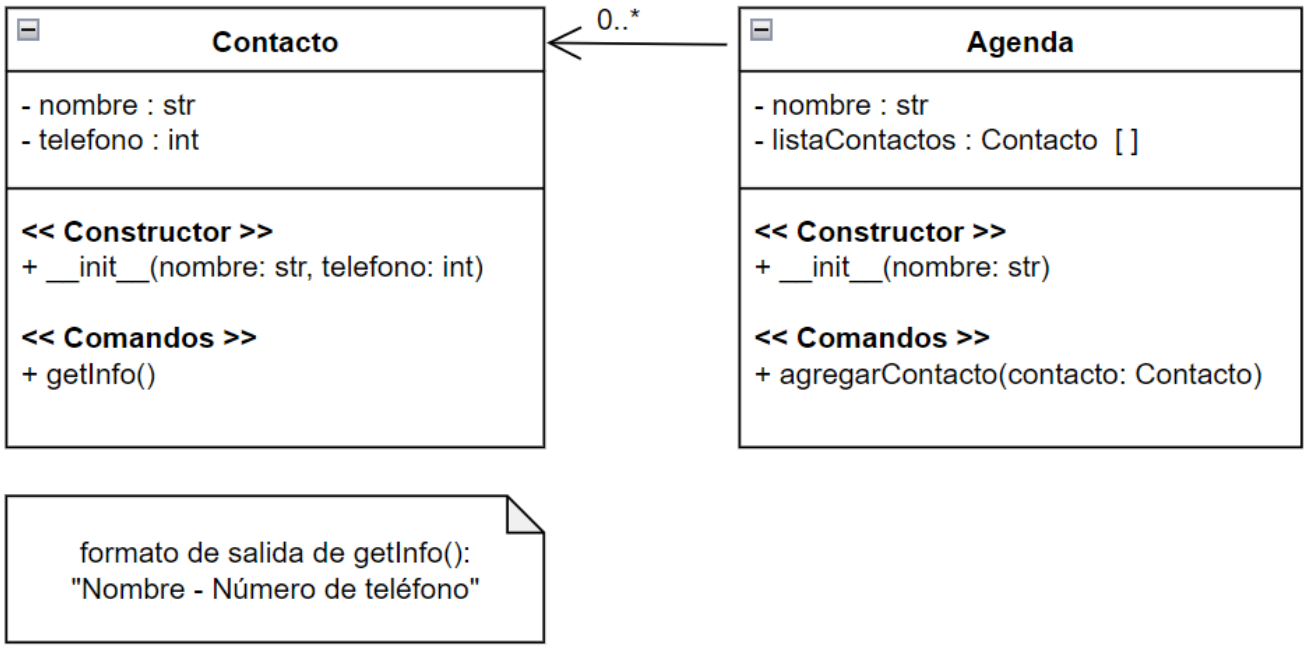
- Si la lista alcanzó la cantidad máxima de temas, al intentar agregar un tema se lanza una excepción personalizada LimiteTemasError.

c) Definir una condición adicional para lanzar otra excepción personalizada (a elección) e implementarla.

IMPORTANTE: dentro de la clase deben lanzarse las excepciones mencionadas, mientras que en el archivo de pruebas debe incluirse un ejemplo de cada una con el control correspondiente.

Ejercicio 2: Métodos mágicos

Se tiene un modelo de clases representado en el diagrama UML.



Realizar la implementación de las clases para que sea posible efectuar las operaciones indicadas en el archivo de prueba.

IMPORTANTE: no es obligatoria la implementación de los métodos de consulta y modificación que no estén indicados en el diagrama, a menos que sean necesarios para completar la implementación de otras funcionalidades.

Ejercicio 3: Herencia

Se cuenta con la implementación de la clase abstracta JuegoDeEquipos, que incluye información de nombre, cantidad de jugadores por equipo y forma en que el juego finaliza (cadena de caracteres, puede ser sólo 'puntos' o 'tiempo').

a) Implementar la clase JuegoEnCancha, derivada de JuegoDeEquipos y con un atributo que describa el tipo de piso ('césped', 'sintético', 'madera', etc). Debe ser también una clase abstracta.

b) Implementar las clases derivadas de JuegoEnCancha, con las siguientes condiciones:

- Clase PartidoVoley: con información de nombres de equipo local y visitante, y marcador en cada set (para ganar un partido, un equipo debe ganar 3 sets, si van empatados en dos sets se juega el quinto y siempre hay un ganador).

- Clase PartidoFutbol: con información de nombres de equipo local y visitante, y marcador del partido (puede ganar un equipo o haber empate).

c) Adicionalmente, se deben realizar las siguientes implementaciones:

i. Cada clase derivada de JuegoEnCancha debe sobrescribir el método detallarJuego(), para incorporar además la información propia de los atributos de clase.

ii. La clase PartidoFutbol debe sobrecargar el método golesDelPartido(), que imprime en pantalla información de los goles del partido, de modo que sea posible utilizarlo en las siguientes alternativas:

golesDelPartido(30, 52, 77): imprime "Los goles del partido fueron a los 30, 52 y 77 minutos"

golesDelPartido([[30, 'V'], [52, 'L'], [77, 'L']]): imprime "Goles del partido: Visitante a los 30min - Local a los 52 min - Local a los 77 min"

golesDelPartido((0,0), (0,1), (1,1), (2, 1)): imprime "El marcador del partido fue 0-0, 0-1, 1-1, 2-1"

La sobrecarga debe funcionar para estas tres alternativas de parámetros, no se requiere considerar casos adicionales pero debe tenerse en cuenta que puede variar la cantidad de goles en el partido.

d) Incluir un ejemplo de código para explicar el concepto de polimorfismo.