EJERCICIOS DE APRENDIZAJE

En este módulo de POO, vamos a empezar a ver cómo dos o más clases pueden relacionarse entre sí, ya sea por una relación entre clases o mediante una herencia de clases.

VIDEOS: Te sugerimos ver los videos relacionados con este tema, antes de empezar los ejercicios, los podrás encontrar en tu aula virtual o en nuestro canal de YouTube.

1. Realizar un programa para que una Persona pueda adoptar un Perro. Vamos a contar de dos clases. Perro, que tendrá como atributos: nombre, raza, edad y tamaño; y la clase Persona con atributos: nombre, apellido, edad, documento y Perro.
Ahora deberemos en el main crear dos Personas y dos Perros. Después, vamos a tener que pensar la lógica necesaria para asignarle a cada Persona un Perro y por ultimo, mostrar desde la clase Persona, la información del Perro y de la Persona.

2. Realizar el juego de la ruleta rusa de agua en Java. Como muchos saben, el juego se trata de un número de jugadores, que, con un revolver de agua, el cual posee una sola carga de agua, se dispara y se moja. Las clases por hacer del juego son las siguientes:

   Clase Revolver de agua: esta clase posee los siguientes atributos: posición actual (posición del tambor que se dispara, puede que esté el agua o no) y posición agua (la posición del tambor donde se encuentra el agua). Estas dos posiciones, se generarán aleatoriamente.

   Métodos:

   - **llenarRevolver():** le pone los valores de posición actual y de posición del agua. Los valores deben ser aleatorios.
   - **mojar():** devuelve true si la posición del agua coincide con la posición actual
   - **siguienteChorro():** cambia a la siguiente posición del tambor
   - **toString():** muestra información del revolver (posición actual y donde está el agua)

   Clase Jugador: esta clase posee los siguientes atributos: id (representa el número del jugador), nombre (Empezara con Jugador más su ID, “Jugador 1” por ejemplo) y mojado (indica si está mojado o no el jugador). El número de jugadores será decidido por el usuario, pero debe ser entre 1 y 6. Si no está en este rango, por defecto será 6.

   Métodos:

   - **disparo(Revolver r):** el método, recibe el revolver de agua y llama a los métodos de mojar() y siguienteChorro() de Revolver. El jugador se apunta, aprieta el gatillo y si el revolver tira el agua, el jugador se moja. El atributo mojado pasa a false y el método devuelve true, sino false.

   Clase Juego: esta clase posee los siguientes atributos: Jugadores (conjunto de Jugadores) y Revolver

   Métodos:

   - **llenarJuego(ArrayList<Jugador>jugadores, Revolver r):** este método recibe los jugadores y el revolver para guardarlos en los atributos del juego.

- **ronda():** cada ronda consiste en un jugador que se apunta con el revolver de agua y aprieta el gatillo. Sí el revolver tira el agua el jugador se moja y se termina el juego, sino se moja, se pasa al siguiente jugador hasta que uno se moje. Si o si alguien se tiene que mojar. Al final del juego, se debe mostrar que jugador se mojó.

  Pensar la lógica necesaria para realizar esto, usando los atributos de la clase Juego.

3. Realizar una baraja de cartas españolas orientada a objetos. Una carta tiene un número entre 1 y 12 (el 8 y el 9 no los incluimos) y un palo (espadas, bastos, oros y copas). Esta clase debe contener un método toString() que retorne el número de carta y el palo. La baraja estará compuesta por un conjunto de cartas, 40 exactamente.

   Las operaciones que podrá realizar la baraja son:

   - **barajar():** cambia de posición todas las cartas aleatoriamente.
   - **siguienteCarta():** devuelve la siguiente carta que está en la baraja, cuando no haya más o se haya llegado al final, se indica al usuario que no hay más cartas.
   - **cartasDisponibles():** indica el número de cartas que aún se puede repartir.
   - **darCartas():** dado un número de cartas que nos pidan, le devolveremos ese número de cartas. En caso de que haya menos cartas que las pedidas, no devolveremos nada, pero debemos indicárselo al usuario.
   - **cartasMonton():** mostramos aquellas cartas que ya han salido, si no ha salido ninguna indicárselo al usuario
   - **mostrarBaraja():** muestra todas las cartas hasta el final. Es decir, si se saca una carta y luego se llama al método, este no mostrara esa primera carta.

EJERCICIOS DE APRENDIZAJE EXTRA

Estos van a ser ejercicios para reforzar los conocimientos previamente vistos. Estos pueden realizarse cuando hayas terminado la guía y tengas una buena base sobre lo que venimos trabajando. Además, si ya terminaste la guía y te queda tiempo libre en las mesas, podes continuar con estos ejercicios extra, recordando siempre que no es necesario que los termines para continuar con el tema siguiente. Por último, recordá que la prioridad es ayudar a los compañeros de la mesa y que cuando tengas que ayudar, lo más valioso es que puedas explicar el ejercicio con la intención de que tu compañero lo comprenda, y no sólo mostrarlo. ¡Muchas gracias!

1. Ahora se debe realizar unas mejoras al ejercicio de Perro y Persona. Nuestro programa va a tener que contar con muchas personas y muchos perros. El programa deberá preguntarle a cada persona, que perro según su nombre, quiere adoptar. Dos personas no pueden adoptar al mismo perro, si la persona eligió un perro que ya estaba adoptado, se le debe informar a la persona.
   Una vez que la Persona elige el Perro se le asigna, al final deberemos mostrar todas las personas con sus respectivos perros.

2. Nos piden hacer un programa sobre un Cine, que tiene una sala con un conjunto de asientos (8 filas por 6 columnas). De Cine nos interesa conocer la película que se está reproduciendo, la sala con los espectadores y el precio de la entrada. Luego, de las películas nos interesa saber el título, duración, edad mínima y director. Por último, del espectador, nos interesa saber su nombre, edad y el dinero que tiene disponible.
   Para representar la sala con los espectadores vamos a utilizar una matriz. Los asientos son etiquetados por una letra y un número la fila A1 empieza al final del mapa como se muestra en la tabla. También deberemos saber si el asiento está ocupado por un espectador o no, si esta ocupado se muestra una X, sino un espacio vacío.

8 A X	8 B X	8 C X	8 D	8 E X	8 F X
7 A X	7 B X	7 C X	7 D X	7 E	7 F X
6 A	6 B X	6 C	6 D X	6 E X	6 F
5 A X	5 B	5 C X	5 D X	5 E X	5 F X
4 A X	4 B X	4 C X	4 D X	4 E X	4 F X
3 A	3 B X	3 C X	3 D	3 E X	3 F X
2 A X	2 B	2 C X	2 D X	2 E X	2 F
1 A X	1 B X	1 C X	1 D X	1 E X	1 F X

   Se debe realizar una pequeña simulación, en la que se generen muchos espectadores y se los ubique en los asientos aleatoriamente (no se puede ubicar un espectador donde ya este ocupado el asiento).
   Los espectadores serán ubicados de uno en uno y para ubicarlos tener en cuenta que sólo se podrá sentar a un espectador si tiene el dinero suficiente para pagar la entrada, si hay espacio libre en la sala y si tiene la edad requerida para ver la película. En caso de que el asiento este ocupado se le debe buscar uno libre.
   Al final del programa deberemos mostrar la tabla, podemos mostrarla con la letra y numero de cada asiento o solo las X y espacios vacíos.

3. Ha llegado el momento de poner de prueba tus conocimientos. Para te vamos a contar que te ha contratado “La Tercera Seguros”, una empresa aseguradora que brinda a sus clientes coberturas integrales para vehículos.

   Luego de un pequeño relevamiento, te vamos a pasar en limpio los requerimientos del sistema que quiere realizar la empresa.

a. **Gestión Integral de clientes**. En este módulo vamos a registrar la información personal de cada cliente que posea pólizas en nuestra empresa. Nombre y apellido, documento, mail, domicilio, teléfono.

b. **Gestión de vehículos**. Se registra la información de cada vehículo asegurado. Marca, modelo, año, número de motor, chasis, color, tipo (camioneta, sedán, etc.).

c. **Gestión de Pólizas:** Se registrará una póliza, donde se guardará los datos tanto de un vehículo, como los datos de un solo cliente. Los datos incluidos en ella son: número de póliza, fecha de inicio y fin de la póliza, cantidad de cuotas, forma de pago, monto total asegurado, incluye granizo, monto máximo granizo, tipo de cobertura (total, contra terceros, etc.). Nota: prestar atención al principio de este enunciado y pensar en las relaciones entre clases. Recuerden que pueden ser de uno a uno, de uno a muchos, de muchos a uno o de muchos a muchos.

d. **Gestión de cuotas:** Se registrarán y podrán consultar las cuotas generadas en cada póliza. Esas cuotas van a contener la siguiente información: número de cuota, monto total de la cuota, si está o no pagada, fecha de vencimiento, forma de pago (efectivo, transferencia, etc.).

Debemos realizar el diagrama de clases completo, teniendo en cuenta todos los requerimientos arriba descriptos. Modelando clases con atributos y sus correspondientes relaciones.

4. Desarrollar un simulador del sistema de votación de facilitadores en Egg-

El sistema de votación de Egg tiene una clase llamada Alumno con los siguientes atributos: nombre completo, DNI y cantidad de votos. Además, crearemos una clase Simulador que va a tener los métodos para manejar los alumnos y sus votaciones. Estos métodos serán llamados desde el main.

- La clase Simulador debe tener un método que genere un listado de alumnos manera aleatoria y lo retorne. Las combinaciones de nombre y apellido deben ser generadas de manera aleatoria. Nota: usar listas de tipo String para generar los nombres y los apellidos.

- Ahora hacer un generador de combinaciones de DNI posibles, deben estar dentro de un rango real de números de documentos. Y agregar a los alumnos su DNI. Este método debe retornar la lista de dnis.

- Ahora tendremos un método que, usando las dos listas generadas, cree una cantidad de objetos Alumno, elegidos por el usuario, y le asigne los nombres y los dnis de las dos listas a cada objeto Alumno. No puede haber dos alumnos con el mismo dni, pero si con el mismo nombre.

- Se debe imprimir por pantalla el listado de alumnos.

- Una vez hecho esto debemos generar una clase Voto, esta clase tendrá como atributos, un objeto Alumno que será el alumno que vota y una lista de los alumnos a los que votó.

- Crearemos un método votación en la clase Simulador que, recibe el listado de alumnos y para cada alumno genera tres votos de manera aleatoria. En este método debemos guardar a el alumno que vota, a los alumnos a los que votó y sumarle uno a la cantidad de votos a cada alumno que reciba un voto, que es un atributo de la clase Alumno.
  Tener en cuenta que un alumno no puede votarse a sí mismo o votar más de una vez al mismo alumno. Utilizar un hashset para resolver esto.

- Se debe crear un método que muestre a cada Alumno con su cantidad de votos y cuales fueron sus 3 votos.

- Se debe crear un método que haga el recuento de votos, este recibe la lista de Alumnos y comienza a hacer el recuento de votos.

- Se deben crear 5 facilitadores con los 5 primeros alumnos votados y se deben crear 5 facilitadores suplentes con los 5 segundos alumnos más votados. A continuación, mostrar los 5 facilitadores y los 5 facilitadores suplentes.