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.