Guía de ejercicios POO - Clases y objetos - Parte 2

1. Se solicita implementar una o varias clases que nos permita controlar un tablero electronico de basquet, por lo tanto:
    a. Crear una clase TableroDeBasquet el cual tendrá como atributos: local (string), visitante (string), puntos_local (entero), y puntos_visitante (entero).
    b. Crear un método que adicione puntos de a 3, tanto al local como al visitante
    c. Crear un método que adicione puntos de a 2, tanto al local como al visitante.
    d. Crear un método que adicione puntos de a 1, tanto al local como al visitante.
    e. Analizar si es conveniente guardar el puntaje y el nombre del equipo en la instancia del TableroDeBasquet o si es mejor generar otra clase llamada Equipo y guardar la información en esas instancias de la clase Equipo

2. Se solicita crear una implementación de una baraja española, para ello se solicita:
    a. Crear una clase Carta que posea los atributos de *valor* y *palo*. ACLARACION: solamente los valores entre el 1 y el 12 (8 y 9 no se incluyen)
    b. Crear una clase Baraja que:
        i. Contenga un conjunto de cartas iniciales (40 por defecto). Se recomienda tener algún inicializador de cartas, es decir que entregue todas las cartas correspondientes
        ii. La acción de barajar: cambia de posición todas las cartas aleatoriamente
        iii. La acción de pedir 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.
        iv. Devolver la cantidad de cartas disponibles que aún se puede repartir
        v. La acción de poder dar cartas: 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.
        vi. ACLARACION: tratar a la baraja como si fuese una pila, es decir extraer las cartas desde un solo “lado”

3. Se solicita implementar crear un juego virtual de la ruleta rusa, para ello se deberá:
    a. Crear la clase Revolver que contenga:
        i. La *posición actual* (posición del tambor donde se dispara, puede que esté la bala o no)
        ii. La *posición bala* (la posición del tambor donde se encuentra la bala)
        iii. La acción de *disparar()*: devuelve true si la bala coincide con la posición actual
        iv. La acción de *siguienteBala()*: cambia a la siguiente posición del tambor
        v. La acción de *girar tambor*: deberá de forma aleatoria asignar una nueva posición actual

b. Crear la clase Jugador que contenga:
    i. Un *indentificador del jugador* (del número 1 en adelante)
    ii. Su *nombre*
    iii. Si *está vivo o no*
    iv. La acción de *disparar(revolver)*, si la bala se dispara se deberá cambiar el estado de vivo a no vivo.
c. Crear la clase Juego que contenga:
    i. Un *conjunto de jugadores*
    ii. Un *revolver*
    iii. Un método que chequee si se terminó el juego (finaliza cuando un jugador muere)
    iv. Un método ronda, en el cual cada jugador se va a disparar. Informar el estado de la partida

4. Nos piden realizar una agenda telefónica de contactos.
    a. Un contacto está definido por un nombre y un teléfono (No es necesario de validar). Un contacto es igual a otro cuando sus nombres son iguales.
    b. Una agenda de contactos está formada por un conjunto de contactos (Piensa en que tipo puede ser)
    c. Se podrá crear de dos formas, indicándoles nosotros el tamaño o con un tamaño por defecto (10)
    d. Los métodos de la agenda serán los siguientes:
        i. aniadirContacto(Contacto c): Añade un contacto a la agenda, sino se pueden meter más a la agenda se indicara por pantalla. No se pueden meter contactos que existan, es decir, no podemos duplicar nombres, aunque tengan distinto teléfono.
        ii. existeContacto(Conctacto c): indica si el contacto pasado existe o no.
        iii. listarContactos(): Lista toda la agenda
        iv. buscaContacto(String nombre): busca un contacto por su nombre y muestra su teléfono.
        v. eliminarContacto(Contacto c): elimina el contacto de la agenda, indica si se ha eliminado o no por pantalla
        vi. agendaLlena(): indica si la agenda está llena.
        vii. huecosLibres(): indica cuantos contactos más podemos meter.
    e. Crea un menú con opciones por consola para probar todas estas funcionalidades.