Programación Orientada a Objetos. Práctica 3

Juan A. Romero 2014, aromero@uco.es

LEER HASTA EL FINAL ANTES DE COMENZAR EL EJERCICIO

La clase *Ruleta* mantiene la cantidad de dinero en euros que tiene el casino (banca_, de tipo int), el número entre 0 y 36 que sale en cada jugada de forma aleatoria (bola_, de tipo int), una lista de jugadores (jugadores_) y un crupier (crupier_).
La clase *Ruleta* debe cumplir los siguientes requisitos:

1. El constructor de la clase *Ruleta* inicia la bola a -1, y la banca a 1 millón de euros. Como para crear el crupier necesitamos sus datos, hacer que el constructor de la clase *Ruleta* reciba como parámetro un objeto de tipo *Crupier*.
2. Observadores getBanca() y getBola().
3. Modificador setBanca() que solo admite valores positivos. En caso contrario devuelve *false*.
4. El modificador setBola() que solo admite valores entre 0 y 36. En caso contrario devuelve *false*.
5. El observador getCrupier() y el modificador serCrupier().
6. Observador, get*Jugadores()*, que devuelve la lista de jugadores (jugadores_).
7. El método *bool addJugador()* recibe un jugador como parámetro y añade el jugador al final de la lista de jugadores (si no existe previamente un jugador con ese DNI, en cuyo caso no hace nada y devuelve *false*) y crea un fichero tipo texto de apuestas vacío y devuelve *true*. El fichero debe llamarse `DNI.txt`, siendo *DNI* el DNI del jugador. **Si el fichero ya existe**, lo deja como estaba sin modificarlo ni borrarlo.
8. El método int *deleteJugador()* recibe un jugador y lo borra de la lista de jugadores el jugador con mismo DNI que el recibido. Debe devolver 1 si se ha borrado al jugador, -1 si la lista está vacía y -2 si el DNI no se ha encontrado en la lista de jugadores. No debe borrar el fichero con las apuestas de ese jugador.
9. El método int *deleteJugador()* recibe el DNI de un jugador y borra de la lista de jugadores el jugador con ese DNI. Debe devolver 1 si se ha borrado al jugador, -1 si la lista está vacía y -2 si el DNI no se ha encontrado en la lista de jugadores. No debe borrar el fichero con las apuestas de ese jugador.
10. El método void *escribeJugadores()* escribe los datos de la lista de jugadores en un fichero texto denominado `jugadores.txt`. Cada vez que se escribe este fichero se borra todo su contenido anterior. El formato de este archivo debe ser:

    DNI,código,nombre,apellidos,dirección,localidad,provincia,país,dinero
    DNI,código,nombre,apellidos,dirección,localidad,provincia,país,dinero
    . . .

    Si alguno de los campos está vacío el fichero quedaría de la forma:

    DNI,código,nombre,apellidos,,,,,dinero
    DNI,código,,,,,,,dinero
    . . .

    (recordar que DNI y código de jugador era obligatorio para crear un jugador)

11. El método void *leeJugadores()* lee los datos de los jugadores del fichero jugadores.txt y los mete en la lista de jugadores. La lista de jugadores se borra

antes de añadir los jugadores del fichero jugadores.txt

12. El método void *giraRuleta()* simula el giro de la ruleta y la obtención de un número aleatorio entre 0 y 36.
13. El método void *getPremios()* recorre la lista de jugadores y carga sus apuestas de los ficheros correspondientes. Actualiza el dinero de cada jugador con lo que ha ganado o ha perdido en cada apuesta, y actualiza el dinero de la banca con lo que ha ganado o ha perdido en cada apuesta.
14. Preparar un fichero de pruebas unitarias: `ruleta_unittest.cc`
15. Hacer también un programa principal (`ruleta-ppal.cc`) que pruebe el funcionamiento de todo el sistema. Puedes hacerlo todo lo completo que quieras.
16. Podrán crearse los métodos y funciones auxiliares que se consideren.

NOTAS:

- Hacer un `Makefile` para el proyecto.
- Usar los identificadores de las clases y los métodos exactamente como aparecen en el enunciado ya que luego pasaremos unos tests en los que usaremos dichos identificadores.