Informática II – Prepa Tec Campus Eugenio Garza Lagüera
Actividad 5: Ruleta

Diseña y programa un par de clases en Java que permitan modelar el funcionamiento de una mesa de Ruleta. El programa deberá funcionar de la siguiente manera:

Clase Player

1. Constructor

Define un método constructor que reciba como parámetros de entrada el nombre del jugador, y su fecha de nacimiento (mediante una variable de tipo LocalDate). El constructor deberá encargarse de inicializar las siguientes variables:

- `name`: Utilizando el nombre recibido como parámetro
- `age`: Utilizando la fecha recibida como parámetro, deberá calcular la diferencia de años entre la fecha recibida como parámetro, y la fecha de hoy. Revisa la siguiente liga (https://www.leveluplunch.com/java/examples/number-of-years-between-two-dates/).
- `activeAccount`: Será una bandera boleana que almacenará true sólo cuando la persona sea mayor de edad, de lo contrario, almacenará false.
- `balance`: Inicialmente, almacenará un 0, indicando que el jugador no tiene saldo en su cuenta.
- `currentBetAmount:` Inicialmente, se almacenará un -1.
- `currentBetNumber:` Inicialmente, almacenará un -1.

2. public void addFunds(double balance)

Este método depositará saldo en la cuenta del usuario, afectando la variable **`balance`**. Se cobrará una comisión del 4% sobre el monto depositado, por lo que el saldo de la cuenta sólo se incrementará en el monto restante. Adicionalmente, deberán cumplirse las siguientes condiciones:

- Sólo se podrá depositar en cuentas activas.
- El monto mínimo a depositar será de 20 pesos.
- No se admitirán depósitos negativos.

Al finalizar, se imprimirá en pantalla el texto "$ #1 depositados en la cuenta #2", en donde se sustuirá #2 por el nombre del jugador y #1 por el monto agregado a la cuenta.

3. public void withrawFunds(double balance)

Este método retirará saldo en la cuenta del usuario, afectando la variable **`balance.`** Para llevar a cabo el retiro, se deberán cumplirse las siguientes condiciones:

- El jugador deberá contar con suficiente saldo para realizar el retiro.
- No se podrán retirar cantidades negativas.

Al finalizar, deberá imprimir una leyenda que indique si el retiro se pudo realizar o no.

4. public void bet(int betAmount, int betNumber)

Este método llenará las variables `currentBetAmount y currentBetNumber` con la información recibida como parámetro de entrada.

Para poder realizar la apuesta, deberá cumplir con las siguientes condiciones:

- El usuario deberá contar con suficientes fondos en la cuenta.
- Los fondos serán retirados de la cuenta mediante el método **`withrawFunds.`**

Clase `RouletteTable`

La clase RouletteTable contará con el siguiente funcionamiento:

1. **`Constructor`**
   Deberá instanciar el arreglo Player[] players como un arreglo de 5 posiciones. Esto indicará que máximo podrá haber 5 jugadores por cada mesa de ruleta. Adicionalmente, la variable playerCount comenzará con un valor de 0, pues hay 0 jugadores en dicha mesa.

2. **`public void addPlayer(Player p1)`**
   Almacenará un jugador en la primera posición vacía del arreglo `players`. Para esto, previamente deberá haber instanciado un objeto de la clase Player. Si el arreglo no puede almacenar mas jugadores, entonces imprime un mensaje indicándolo.

3. **`public void removePlayer(int playerIndex)`**
   Eliminará el jugador indicado del arreglo players, recorriendo el resto de los objetos del arreglo a la izquierda. Eliminar un jugador significa reemplazarlo con el valor “null”. Adicionalmente, playerCount se disminuirá en 1.
   Sólo se puede eliminar elementos del arreglo `players` en donde haya un jugador presente.

   Ejemplo:
   ```
   // before
   // players {Pepe, Juan, Jose, Humberto, null}
   // playerCount = 4
   removePlayer(2);
   // after
   // players {Pepe, Juan, Humberto, null, null}
   // playerCount = 3
   ```

4. **`public void spin( )`**
   Este método simulará el giro de una ruleta, generando un número aleatorio 1 y 10. El número aleatorio será almacenado en la variable de instancia `lastSpinNumber`.
   Revisa el funcionamiento de un generador de números aleatorios en la siguiente liga (http://tiny.cc/klvjjz).

5. **`public void pay( )`**
   Recorrerá cada jugador almacenado en el arreglo `players`, revisando el contenido de la variable `RouletteTable.lastSpinNumber` y comparándolo contra `Player.lastBetNumber`. Si el jugador apostó al número correcto, se le depositará 10 veces el monto apostado (`Player.lastBetAmount`). Agrega el saldo utilizando el método addFunds() de la clase Player.

Clase RouletteDemo

Utiliza la clase RouletteDemo.java adjuntada en la actividad para probar el funcionamiento de tu programa. Ejemplo de ejecución:

Output

```
96.0 added to Mario's account.
480.0 added to Jose's account.
Maria’s account is inactive.
576.0 added to Rocio's account.
745.92 added to Linda's account.
74.208 added to Jason's account.

Mario sat at the table.
Jose sat at the table.
Maria is an inactive player.
Rocio sat at the table.
Jason sat at the table.
Linda sat at the table.
We can't add more players!

Withrew 50.0 from Mario.
Mario bet on 1.
Withrew 200.0 from Jose.
Jose bet on 2.
Not enough funds!
Withrew 300.0 from Rocio.
Rocio bet on 4.
Withrew 700.0 from Linda.
Linda bet on 5.
Not enough funds!

The roulette drew 1

Mario won 500.0!
480.0 added to Mario's account.
```