Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC2113 Diseño Detallado de Software

Tarea 2: Código limpio

Rodrigo Toro Icarte

1. Introducción

El objetivo de esta tarea es que practiques programando código limpio. Para ello deberás programar *la escoba*. Tu juego además deberá contar con una batería de tests automáticos y conexión web vía sockets.

2. Reglas de la escoba

La escoba es un juego de cartas de baraja española en el que se deben hacer jugadas que sumen 15.[1]

Para empezar

Para jugar a la escoba se usa una baraja española de 40 cartas. Cada carta tiene una pinta y un valor. Las pintas posibles son 4: *oro*, *espada*, *copa* y *bastos*. Los valores posibles son 1, 2, 3, 4, 5, 6, 7, sota, caballo y rey. Al inicio del juego se reparten tres cartas a cada jugador y se colocan cuatro cartas boca arriba en el centro de la mesa.

Jugar las cartas

En su turno cada jugador debe lanzar una de sus cartas e intentar que sume 15 puntos con una o varias de las que hay sobre la mesa y llevárselas. Si no es posible sumar 15 con la carta jugada y alguna(s) de la mesa, la carta se deja descubierta en el centro con las demás cartas.

- **Valor de las cartas**: las figuras (sota, caballo y rey) cuentan 8, 9 y 10 puntos respectivamente, el resto de cartas cuentan lo que indica su índice (1, 2, 3, 4, 5, 6 y 7).
- **Escoba**: si un jugador en su turno al sumar 15 se lleva todas las cartas que haya sobre la mesa, hace una escoba.

Cuando todos los jugadores han jugado sus tres cartas, vuelven a repartirse otras tres cartas a cada uno. Y así hasta que se agoten las cartas de la baraja.

Al acabar la mano (una vez jugadas todas las cartas de la baraja), las cartas que queden en el centro **se las lleva el último jugador que haya logrado llevarse cartas** en su turno.

Al comienzo de la mano, si las cuatro cartas depositadas sobre la mesa suman exactamente uno o dos grupos de 15, **quien reparte se las lleva para sí** sumando una o dos escobas respectivamente.

Recuento de puntos

Una vez acabada una mano se procede al recuento de los puntos:

- 1 punto por cada escoba.

[1] Estas reglas del juego las saqué de aquí: `https://www.ludoteka.com/juegos/escoba/reglas`

- 1 punto por tener el siete de oros.
- 1 punto por tener la mayoría de los sietes de la baraja.
- 1 punto por tener la mayoría de cartas de la baraja.
- 1 punto por tener la mayoría de cartas de pinta *oro* de la baraja.

En caso de empate en cualquiera de los conceptos (sietes, oros, cartas), ambos jugadores suman el punto correspondiente.

Final de la partida

Se juegan tantas manos como hagan falta hasta que alguien alcance los 16 puntos. En ese momento gana quien tenga la mayor cantidad de puntos (siendo posible que existan empates).

Jugar online

Puedes jugar online en esta página: `https://www.ludoteka.com/juegos/escoba/reglas`.

3. Requisitos funcionales

En esta tarea tendrás que entregar un proyecto de C# que incluya 3 subproyectos. Dos de esos proyectos serán proyectos en consola, uno llamado `Cliente` y otro llamado `Servidor`. El tercer proyecto debe ser llamado `Sevidor.Tests`. Este último proyecto contendrá tests para el proyecto `Servidor`.

3.1. Proyecto Servidor

Al correr el servidor lo primero que debe ocurrir es preguntarle al usuario si quiere correr en modo *local* o *modo servidor*. Si selecciona el modo local el usuario debe poder jugar la escoba directo en el servidor (y elegir las acciones de ambos jugadores). Si selecciona el modo servidor se debe esperar a que dos clientes TCP se conecten para comenzar el juego.

El servidor debe permitir que dos jugadores jueguen la escoba. Inicialmente el jugador `0` reparte cartas y parte jugando el jugador `1`. Luego ambos jugadores juegan cartas hasta que se acaben las cartas del mazo. En ese momento se cuentan los puntos ganados y se vuelve a empezar. Ahora reparte el jugador `1` y parte el jugador `0`. Esto se repite hasta que alguien consiga 16 puntos. Nota que saber quién reparte es importante para manejar el caso en que aparezca una escoba al iniciar una partida.

En cada turno el jugador debe elegir una carta de su mano a jugar. En ese momento existen 3 casos que tu programa debe revisar automáticamente:

- **Caso 1**: No existe una combinación de cartas en la mesa que, sumada a la carta bajada, suman 15. En ese caso se le informa al usuario y su carta queda en la mesa.
- **Caso 2**: Existe exactamente una combinación de cartas en la mesa que, sumada a la carta bajada, suman 15. En ese caso se le informa al usuario y el usuario gana las cartas que suman 15.
- **Caso 3**: Existen 2 o más combinación de cartas que, sumada a la carta bajada, suman 15. En ese caso se le debe preguntar al usuaria cuál de esas combinaciones quiere usar y el usuario gana las cartas elegidas.

Por ejemplo, considera el siguiente caso. Es el turno del jugador `1`. Las cartas en la mesa son el *rey de bastos*, *uno de copa*, *tres de oro* y el *rey de copas*. El usuario decide bajar el *uno de oro*. En ese caso hay dos formas en que podemos sumar 15 usando esa carta. Por lo mismo debemos preguntarle al usuario qué jugada prefiere:

```
----------------------------------
Juega Jugador 1
Mesa actual: (1) Rey_Bastos, (2) 1_Copa, (3) 3_Oro, (4) Rey_Copa
Mano jugador: (1) Rey_Espada, (2) 1_Oro, (3) 5_Bastos
¿Qué carta quieres bajar?
(Ingresa un número entre 1 y 3)
2

Hay 2 jugadas en la mesa:
1- 1_Oro, 1_Copa, 3_Oro, Rey_Copa
2- 1_Oro, Rey_Bastos, 1_Copa, 3_Oro
¿Cuál quieres usar?
(Ingresa un número entre 1 y 2)
2
Jugador 1 se lleva las siguientes cartas 1_Oro, Rey_Bastos, 1_Copa, 3_Oro
```

Si el jugador se lleva todas las cartas en la mesa se debe indicar que ganó una *escoba* de alguna forma que resalte. Esto con el objetivo de que sea fácil para el ayudante verificar que el conteo de puntos esté funcionando correctamente.

```
----------------------------------
Juega Jugador 1
Mesa actual: (1) 4_Copa, (2) Sota_Espada
Mano jugador: (1) 3_Bastos
¿Qué carta quieres bajar?
(Ingresa un número entre 1 y 1)
1
Jugador 1 se lleva las siguientes cartas 3_Bastos, 4_Copa, Sota_Espada

ESCOBA!********************************************** JUGADOR 1
```

Finalmente cuando termine una mano (es decir, cuando se acaban las cartas en el mazo) se deben mostrar todas las cartas ganadas por cada jugador y sus puntos totales ganados desde el inicio del juego:

```
----------------------------------
Cartas ganadas en esta ronda
   Jugador 0: Caballo_Espada, 6_Bastos, Rey_Oro, 5_Espada, 3_Espada, Caballo_Bastos,
    Sota_Copa, 5_Copa
   Jugador 1: 1_Oro, Rey_Bastos, 1_Copa, 3_Oro, 5_Bastos, Rey_Copa, 2_Oro, 7_Oro, 6_Copa, 2
    _Copa, 4_Espada, Caballo_Oro, 5_Oro, Rey_Espada, 3_Copa, Sota_Oro, 4_Bastos, 1_Bastos,
    Sota_Bastos, 6_Espada, 6_Oro, Caballo_Copa, 3_Bastos, 4_Copa, Sota_Espada, 7_Bastos, 7
    _Copa, 1_Espada, 4_Oro, 2_Bastos, 7_Espada, 2_Espada
----------------------------------
Total puntos ganados
   Jugador 0: 1
   Jugador 1: 5
----------------------------------
```

3.2. Proyecto Cliente

El proyecto `Cliente` debe permitir conectarse al servidor y jugar de manera remota. Sin embargo, por simplicidad basta con que establezcas una conexión con el `localhost`. Es decir, el servidor y ambos clientes pueden estar en el mismo computador.

Cada cliente representa un jugador. Por lo mismo, no deben poder ver la mano del oponente. Pero sí deben poder ver las cartas que juega y va ganando a medida que progresa la partida.

3.3. Proyecto Servidor.Tests

El proyecto `Servidor.Tests` debe ser un `Unit Test Project` escrito en `xUnit`. El proyecto debe incluir tests del proyecto `Servidor`. El objetivo es tener, al menos, 80 % de test coverage. No es necesario testear el código del cliente.

3.4. Generador de números aleatorios

Es importante que podamos replicar los errores que encontremos en tu tarea y compartirlos contigo. Para ello te pedimos que uses la siguiente clase para generar números aleatorias al momento de revolver el mazo de cartas. Esto nos permitirá detectar errores que ocurren para random seed particulares. Este código también se encuentra disponible en canvas:

```
public static class GeneradorNumerosAleatorios
{
    private const int RandomSeed = 0;
    private static Random rng = new Random(RandomSeed);

    public static double Generar() => rng.Next();
}
```

4. Rúbrica

A grandes rasgos, usaremos esta rúbrica para evaluar tu tarea:

- **1 punto**: Test coverge de al menos 80 % del `Servidor`. Además los ayudantes revisarán que tus tests sean apropiados.
- **1 punto**: Sockets. Los ayudantes verificarán que se pueda jugar de dos jugadores vía sockets.
- **4 puntos**: Clean code. Los ayudantes mirarán tu código y evaluarán que siga los principios descritos en *clean code*, capítulos 2 al 10.
- **-6 puntos**: Requisitos funcionales. Los ayudantes también revisarán que tu código funcione correctamente. Por cada error encontrado irán descontando puntos de tu nota.

El objetivo de descontar puntos por funcionalidades no implementadas es incentivarte a que entregues código limpio que funcione :)

5. Consejos

Te recomendamos partir implementando la versión local del servidor usando el patrón MVC. No prestes demasiada atención en escribir código limpio por el momento. Primero enfócate en que tu código funcione. Luego agrega los tests unitario. Técnicamente sería mejor partir creando los tests y luego implementar código en el servidor, pero eso podría no ser una buena idea si no estás acostumbrado a trabajar con test unitarios.

Cuando tengas el código del servidor funcionando (y bien testeado) limpia tu código siguiendo los consejos de clean code. También es posible ir limpiando tu código a medida que lo programas. En cualquier caso tener los tests reducirá la probabilidad de que introduzcas *bugs* mientras limpias el código.

Finalmente, agrega sockets al servidor y crea los clientes. Este último paso debería ser relativamente simple si implementaste MVC correctamente (ver ejemplo del dominó de la clase 6).