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

Entrega 2: Raw Deal Card Game

Introducción

En esta entrega debes implementar la lógica general del juego sin considerar los efectos de las cartas. Además esta será la primera entrega en que evaluaremos la limpieza de tu código. Tu código debe seguir los principios de los capítulos 2, 3 y 6 del libro *Clean Code*.

Test cases

Esta nueva versión del código base incluye todos los test cases que debes completar para el final del proyecto. Pero en esta entrega solo debes pasar los siguientes test cases:

- `TestInvalidDecks`: Testean que no se puedan usar mazos inválidos.
- `TestNoEffectsDecks`: Testean el funcionamiento del juego sin usar habilidades ni cartas con efectos.
- `TestSuperstarAbilities`: Testean el funcionamiento completo del juego usando cartas sin efectos.

Formato mazos

Los mazos siguen el mismo formato que en la entrega 1. Por ejemplo, este es un mazo de `Kane`:

```
KANE (Superstar Card)
Hip Toss
Escape Move
Arm Bar Takedown
Kane's Chokeslam
Jockeying for Position
Recovery
Step Aside
Kane's Flying Clothesline
Body Slam
Back Breaker
Step Aside
Arm Bar Takedown
Break the Hold
Wrist Lock
Punch
Knee to the Gut
Kane's Return!
Shake It Off
Don't Think Too Hard
Hip Toss
Arm Bar
Kick
Recovery
Shake It Off
Rolling Takedown
Elbow to the Face
```

```
Rolling Takedown
Jockeying for Position
Punch
Arm Bar Takedown
Arm Bar
Roll Out of the Ring
Body Slam
Arm Bar
Roll Out of the Ring
Back Breaker
Punch
Break the Hold
Hip Toss
Reverse DDT
Gut Buster
Jockeying for Position
Gut Buster
Don't Think Too Hard
Escape Move
Gut Buster
Wrist Lock
Escape Move
Reverse DDT
Don't Think Too Hard
Back Breaker
Step Aside
Break the Hold
Wrist Lock
Kick
Kick
Body Slam
Elbow to the Face
Knee to the Gut
Jockeying for Position
```

Se asume que la última carta del mazo representa la carta que está al tope del arsenal. Por lo mismo, la mano inicial de `Kane` (cuyo `hand size` es 7) será la siguiente:

```
Wrist Lock
Kick
Kick
Body Slam
Elbow to the Face
Knee to the Gut
Jockeying for Position
```

Colecciones de cartas

Cada jugador maneja 3 colecciones de cartas:

- `Hand`: Son las cartas que tiene en su mano.
- `Arsenal`: Son las cartas en su mazo.
- `Ringside`: Son las cartas en su pila de descarte.

Para que los test cases funcionen es importante que todas estas colecciones sigan las siguientes reglas. Digamos que estas colecciones son representadas usando listas. Entonces,

- El *tope del arsenal* es la última carta de la lista `arsenal`.
- Al robar una carta del `arsenal` se saca la carta del tope (la última) y se pone al final de la lista `hand`.

- Toda carta que vaya al `ringside` (ya sea porque fue descartada o producto de aplicar daño) se pone al final de la lista `ringside`.
- Nunca desordenes (o revuelvas) ninguna de estas listas.

Ver cartas

La primera funcionalidad extra que deberás implementar es *ver cartas*. Esto permite que el jugador pueda ver las cartas de su mano, ring area y ringside y mirar el ring area y ringside de su oponente. Por ejemplo, acá `HHH` elige *ver cartas* y luego pide ver las cartas de su mano (en el ejemplo `HHH` se enfrenta a `HHH`).

```
-----------------------------------------
Comienza el turno de HHH.
-----------------------------------------
HHH: 0F, tiene 11 cartas en la mano y 49 en el arsenal.
HHH: 0F, tiene 10 cartas en la mano y 50 en el arsenal.
-----------------------------------------
1- Ver cartas
2- Jugar carta
3- Terminar turno
4- Rendirse
(Ingresa un número entre 1 y 4)
INPUT: 1
1- Mi mano
2- Mi ring area
3- Mi ringside pile
4- El ring area de mi oponente
5- El ringside pile de mi oponente
(Ingresa un número entre 1 y 5)
INPUT: 1
-----------------------------------------
0- *Wrist Lock*. Info: 0F/2D, Maneuver, Submission.
1- *Punch*. Info: 0F/3D, Maneuver, Strike.
2- *Hip Toss*. Info: 0F/3D, Maneuver, Grapple.
3- *Chin Lock*. Info: 2F/3D, Maneuver, Submission.
4- *Roundhouse Punch*. Info: 2F/4D, Maneuver, Strike.
5- *Russian Leg Sweep*. Info: 2F/4D, Maneuver, Grapple.
6- *Sit Out Powerbomb*. Info: 3F/6D, Maneuver, Grapple.
7- *Gut Buster*. Info: 4F/5D, Maneuver, Grapple.
8- *Atomic Facebuster*. Info: 4F/6D, Maneuver, Grapple.
9- *Inverse Atomic Drop*. Info: 6F/5D/1SV, Maneuver, Grapple.
10- *Big Boot*. Info: 6F/6D, Maneuver, Strike.
-----------------------------------------
```

Para implementar esta funcionalidad debes usar los siguientes métodos a `View`:

- `AskUserWhatSetOfCardsHeWantsToSee()`: Muestra el menú con las 5 opciones de sets de cartas que se pueden ver (mano, ring area, etc) y retorna un `CardSet` que indica la elección del usuario.
- `ShowCards(List<string> cards)`: Muestra la lista de cartas siguiendo el formato requerido por los test cases. Nota que se recibe una lista de strings donde cada string tiene toda la información asociada a la carta. Para transformar una carta en un string siguiendo el formato requerido puedes usar el método `Formatter.CardToString(IViewableCardInfo cardInfo)` que también proveemos en `RawDealView`. Notar que este método recibe cualquier objeto que implemente la interfaz `IViewableCardInfo`.

Jugar carta

La otra funcionalidad que debes implementar es *jugar carta*. Cuando el usuario elige jugar carta solo se deben mostrar las cartas que efectivamente puede jugar. Durante el turno, solo se pueden jugar cartas que

sean de tipo `Maneuver` o `Action` y cuyo `fortitude` es menor o igual al `fortitude` del jugador. Además se debe mostrar el tipo con el que se va a jugar la carta.

Por ejemplo, acá se enfrentan un mazo de `HHH` contra un mazo de `Kane`:

```
-----------------------------------------
HHH: 0F, tiene 11 cartas en la mano y 49 en el arsenal.
KANE: 0F, tiene 7 cartas en la mano y 53 en el arsenal.
-----------------------------------------
1- Ver cartas
2- Jugar carta
3- Terminar turno
4- Rendirse
(Ingresa un número entre 1 y 4)
INPUT: 2
-----------------------------------------
0- [MANEUVER] *Punch*. Info: 0F/3D, Maneuver, Strike.
1- [MANEUVER] *Punch*. Info: 0F/3D, Maneuver, Strike.
2- [MANEUVER] *Punch*. Info: 0F/3D, Maneuver, Strike.
3- [MANEUVER] *Kick*. Info: 0F/5D, Maneuver, Strike. Effect: When successfully played, you
    must take the top card of your Arsenal and put it into your Ringside pile.
4- [MANEUVER] *Kick*. Info: 0F/5D, Maneuver, Strike. Effect: When successfully played, you
    must take the top card of your Arsenal and put it into your Ringside pile.
-----------------------------------------
(Ingresa un número entre -1 y 4)
INPUT: 0
-----------------------------------------
```

En el resumen con la información de los jugadores se debe mostrar primero al jugador que tiene el turno. En este caso es el turno de `HHH` – quien tiene 11 cartas en su mano y no tiene `fortitude` (`0F`). A pesar de tener 11 cartas, cuando `HHH` elige jugar una carta solo le aparecen 3 opciones: jugar `Wrist Lock` como maniobra, jugar `Punch` como maniobra o jugar `Hip Toss` como maniobra. Actualmente, estas son las únicas cartas que `HHH` puede jugar pues son las únicas cartas de su mano que piden `0F`.

Luego de mostrar las cartas jugables el usuario pude elegir jugar una de ellas o volver al menú anterior (ingresando `-1`). Digamos que `HHH` decide jugar el `Punch [0F/3D]`. Como resultado, la carta pasa de la mano de `HHH` a su `ring area` y se aplican 3 de daño a `Kane`:

```
-----------------------------------------
HHH intenta jugar la siguiente carta:
[MANEUVER] *Punch*. Info: 0F/3D, Maneuver, Strike.
-----------------------------------------
La carta fue exitosamente jugada.
KANE recibe 3 de daño.
-------------------------------- 1/3
*Kick*. Info: 0F/5D, Maneuver, Strike. Effect: When successfully played, you must take the
    top card of your Arsenal and put it into your Ringside pile.
-------------------------------- 2/3
*Hellfire & Brimstone*. Info: 6F/0D, Action, Kane/Unique. Effect: All players discard all
    cards from their hands. Your opponent places the top 5 cards of his Arsenal into his
    Ringside pile.
-------------------------------- 3/3
*Kane's Chokeslam*. Info: 12F/12D/2SV, Maneuver, Grapple/Kane/Unique.
-----------------------------------------
```

Recuerda que aplicar daño significa pasar la carta del tope del `arsenal` de `Kane` a su pila de descarte (`ringside`). Además se van mostrando una a una las cartas que van pasando. Luego de completado el daño, `HHH` puede seguir jugando cartas (nota que su `fortitude` incrementó a 3[1]):

[1]El `fortitude` de un jugador es igual a la suma de todos los atributos `damage` de las cartas en su `ring area`.

```
-----------------------------------------
HHH: 3F, tiene 10 cartas en la mano y 49 en el arsenal.
KANE: 0F, tiene 7 cartas en la mano y 50 en el arsenal.
-----------------------------------------
1- Ver cartas
2- Jugar carta
3- Terminar turno
4- Rendirse
(Ingresa un número entre 1 y 4)
```

Para implementar esta funcionalidad agregamos los siguientes métodos a `View`:

- `AskUserToSelectAPlay(List<string> plays)`: Muestra la lista de jugadas posibles y retorna el id de la jugada seleccionada. Para mostrar las jugadas en el formato correcto puedes usar el método `Formatter.PlayToString(IViewablePlayInfo playInfo)` que proveemos en `RawDealView`. Se retorna `-1` cuando el jugador decide volver al menú anterior sin jugar una carta.
- `SayThatPlayerIsTryingToPlayThisCard(...)`: Indica que el jugador intenta jugar una carta.
- `SayThatPlayerSuccessfullyPlayedACard(...)`: Muestra que la jugada fue exitosa.
- `SayThatSuperstarWillTakeSomeDamage(...)`: Anuncia que un jugador recibirá cierta cantidad de daño.
- `ShowCardOverturnByTakingDamage(...)`: Muestra el mensaje con la carta que pasó del tope del `arsenal` al `ringside` al aplicar daño.

Fin del juego

En esta entrega deberás implementar las condiciones de término de la partida. La partida termina cuando al finalizar un turno alguno de los jugadores no tiene cartas en su `arsenal`. En ese minuto el jugador que aún tenga cartas en su `arsenal` gana la partida. Existen 2 escenarios comunes de término:

- Un jugador recibe daño teniendo cero cartas en su `arsenal`. En ese caso la partida termina inmediatamente.
- Un jugador roba la última carta de su `arsenal` al inicio de su turno. En ese caso el jugador tiene todo su turno para ganar la partida o recuperar cartas a su `arsenal` y así poder seguir jugando. Pero si termina su turno con cero cartas en su `arsenal` pierde la partida.

Habilidades

Cada superestrella tiene su propia habilidad que deberás implementar. Acá te dejamos una breve descripción de cómo funciona cada habilidad.

HHH Su habilidad es tener un `hand size` muy alto. Fuera de eso no tiene habilidad.

Kane Su habilidad consiste en que al iniciar su turno, antes de robar, el oponente está obligado a pegarse uno de daño. Los métodos de la vista relacionados a aplicar daño te servirán para programar esta habilidad.

The Rock Su habilidad consiste en que al iniciar su turno, antes de robar, puede elegir una carta de su `ringside` y ponerla al fondo de su `arsenal`.[2] Nota que, a diferencia de `Kane`, `The Rock` podría decidir no usar su habilidad en algún turno por lo que se le debe preguntar al usuario si quiere usar su habilidad.

[2]Hay que poner la carta en la posición `0` de la lista que representa el `arsenal`.

Además, si `The Rock` no tiene cartas en su `ringside` no puede usar su habilidad y, por lo mismo, no hay que preguntarle al usuario si quiere usar su habilidad. Para implementar esta habilidad necesitarás usar los siguientes métodos de la vista:

- `DoesPlayerWantToUseHisAbility(...)`: Retorna `true` si el usuario dice que quiere usar su habilidad.
- `SayThatPlayerIsGoingToUseItsAbility(...)`: Indica que el usuario va a usar su habilidad.
- `AskPlayerToSelectCardsToRecover(...)`: Retorna el `id` de la carta que el usuario quiere recuperar de su `ringside`. El `id` es la posición en la lista que es dada de input a este método.

Undertaker Su habilidad solo puede ser usada una vez por turno y `Undertaker` debe tener al menos 2 cartas en su mano. Esta habilidad tiene dos pasos. Primero `Undertaker` descarta dos cartas. En este juego descartar significa poner cartas de la mano en el `ringside`. Luego, `Undertaker` elige cualquier carta de su `ringside` y la pone en su mano. Para implementar esta habilidad necesitarás usar los siguientes métodos:

- `SayThatPlayerIsGoingToUseItsAbility(...)`: Indica que el usuario va a usar su habilidad.
- `AskPlayerToSelectACardToDiscard(...)`: Retorna el `id` de la carta que el usuario quiere descartar.
- `AskPlayerToSelectCardsToPutInHisHand(...)`: Retorna el `id` de la carta que el usuario quiere pasar de su `ringside` a su mano.

Jericho Su habilidad solo puede ser usada una vez por turno y `Jericho` debe tener al menos una carta en su mano. Esta habilidad también tiene dos pasos. Primero `Jericho` descarta una carta. Luego su oponente descarta una carta. Para implementar esta habilidad necesitas llamar a algunos de los métodos usados en la habilidad del `Undertaker`.

Mankind Su habilidad es automática y tiene dos partes. Por un lado, toda carta que dañe a `Mankind` hace 1 menos de daño. Por ejemplo, si `HHH` le juega un `Punch [0F/3D]` a `Mankind` la carta hace 2 de daño en vez de 3. Además de esto, `Mankind` roba 2 cartas en vez de una al inicio de su turno. La única excepción a esta regla es que si a `Mankind` solo le queda una carta en el `arsenal` al inicio de su turno entonces robará 1 carta en vez de 2. Este rebuscado caso sale en uno de los test cases :)

Stone Cold Su habilidad solo se puede usar una vez por turno y si `Stone Cold` aún tiene cartas en su `arsenal`. La habilidad consiste en robar una carta del `arsenal` (i.e., poner la carta del tope del `arsenal` en la mano) y luego elegir una carta de la mano y ponerla al fondo de su `arsenal` (en la posición `0`). Para implementar esta habilidad necesitarás usar los siguientes métodos de la vista:

- `SayThatPlayerIsGoingToUseItsAbility(...)`: Indica que el usuario va a usar su habilidad.
- `SayThatPlayerDrawCards(...)`: Indica que el jugador robó una carta.
- `AskPlayerToReturnOneCardFromHisHandToHisArsenal(...)`: Retorna al `id` de la carta en la mano del jugado que se debe poner al fonde del `arsenal`.

Finalmente, solo debes darle la opción de usar su habilidad al jugador si es que efectivamente puede usarla. Para ello la vista incluye un menú con y sin la opción de usar la habilidad:

- `AskUserWhatToDoWhenUsingHisAbilityIsPossible()`
- `AskUserWhatToDoWhenHeCannotUseHisAbility()`

Rúbrica

Esta entrega tiene puntaje por funcionalidad y por limpieza de código. El puntaje por funcionalidad es asignado de la siguiente manera:

- **[1.0 punto]** Porcentaje de test cases pasados en `TestInvalidDecks`.
- **[2.0 puntos]** Porcentaje de test cases pasados en `TestNoEffectDecks`.
- **[3.0 puntos]** Porcentaje de test cases pasados en `TestSuperstarAbilities`.

El puntaje por limpieza de código es asignado de la siguiente manera:

- **[1.5 puntos]** Sigue los principios del cap. 2 de *clean code*.
- **[2.5 puntos]** Sigue los principios del cap. 3 de *clean code*.
- **[2.0 puntos]** Sigue los principios del cap. 6 de *clean code*.

Finalmente, tu nota final será igual al promedio geométrico entre el puntaje por funcionalidad y el puntaje por limpieza de código (más el punto base), donde el promedio geométrico entre x e y es igual a $\sqrt{xy}$.

Por ejemplo, si tienes 3 puntos por funcionalidad y 5 puntos por limpieza de código entonces tu nota será $\sqrt{3 \cdot 5} + 1 = 4{,}9$. Pero si tienes 6 puntos en funcionalidad y solo 1 punto en limpieza de código entonces tu nota será $\sqrt{6 \cdot 1} + 1 = 3{,}5$.