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

Entrega 1: Raw Deal Card Game

Introducción

En esta entrega debes implementar la funcionalidad de chequear si un mazo es válido (discutida en la Sección 3 del enunciado del proyecto). Además, en caso de que ambos mazos sean válidos, se debe iniciar una partida y el primer jugador se rendirá de inmediato, terminando el juego.

Test cases

El código base incluye test cases asociados que te ayudarán a ver si tu entrega funciona. A grandes rasgos, se testean tres escenarios distintos.

El primer escenario consiste en que se le pide al usuario elegir su mazo (dentro de un set de posibilidades) y elige un mazo inválido. En ese caso se muestra un mensaje indicando que el mazo es inválido y termina el programa, tal como muestra el siguiente test case:

```
-------------------------
Elige un mazos
1- data/02-InvalidDecks/01.txt
2- data/02-InvalidDecks/02.txt
3- data/02-InvalidDecks/03.txt
4- data/02-InvalidDecks/04.txt
5- data/02-InvalidDecks/05.txt
6- data/02-InvalidDecks/06.txt
7- data/02-InvalidDecks/07.txt
8- data/02-InvalidDecks/08.txt
9- data/02-InvalidDecks/09.txt
10- data/02-InvalidDecks/10.txt
11- data/02-InvalidDecks/11.txt
12- data/02-InvalidDecks/12.txt
(Ingresa un número entre 1 y 12)
INPUT: 3
El mazo ingresado es inválido.
```

Notar que en los test cases se usa el keyword `INPUT:` para denotar que el resto de la línea es un input ingresado por el usuario. Las líneas que no parten con `INPUT:` son prints que tu programa debe mostrar.

El segundo escenario es que el primer mazo ingresado sea válido pero el segundo sea inválido. En este caso se indica que el segundo mazo es inválido y se termina el programa.

```
-------------------------
Elige un mazos
1- data/02-InvalidDecks/01.txt
2- data/02-InvalidDecks/02.txt
3- data/02-InvalidDecks/03.txt
4- data/02-InvalidDecks/04.txt
5- data/02-InvalidDecks/05.txt
6- data/02-InvalidDecks/06.txt
7- data/02-InvalidDecks/07.txt
```

```
8- data/02-InvalidDecks/08.txt
9- data/02-InvalidDecks/09.txt
10- data/02-InvalidDecks/10.txt
11- data/02-InvalidDecks/11.txt
12- data/02-InvalidDecks/12.txt
(Ingresa un número entre 1 y 12)
INPUT: 1
-------------------------
Elige un mazos
1- data/02-InvalidDecks/01.txt
2- data/02-InvalidDecks/02.txt
3- data/02-InvalidDecks/03.txt
4- data/02-InvalidDecks/04.txt
5- data/02-InvalidDecks/05.txt
6- data/02-InvalidDecks/06.txt
7- data/02-InvalidDecks/07.txt
8- data/02-InvalidDecks/08.txt
9- data/02-InvalidDecks/09.txt
10- data/02-InvalidDecks/10.txt
11- data/02-InvalidDecks/11.txt
12- data/02-InvalidDecks/12.txt
(Ingresa un número entre 1 y 12)
INPUT: 6
El mazo ingresado es inválido.
```

El último escenario es que ambos mazos ingresados sean válidos. Si ello ocurre, ambos jugadores roban su mano inicial y comienza jugando quien tenga mayor `superstar value`. En caso de empate partirá el jugador cuyo mazo haya sido ingresado primero. Luego el jugador que parte roba una carta y se rinde, otorgándole la victoria al otro jugador.

```
-------------------------
Elige un mazos
1- data/01-ValidDecks/01.txt
2- data/01-ValidDecks/02.txt
3- data/01-ValidDecks/03.txt
4- data/01-ValidDecks/04.txt
5- data/01-ValidDecks/05.txt
6- data/01-ValidDecks/06.txt
7- data/01-ValidDecks/07.txt
(Ingresa un número entre 1 y 7)
INPUT: 1
-------------------------
Elige un mazos
1- data/01-ValidDecks/01.txt
2- data/01-ValidDecks/02.txt
3- data/01-ValidDecks/03.txt
4- data/01-ValidDecks/04.txt
5- data/01-ValidDecks/05.txt
6- data/01-ValidDecks/06.txt
7- data/01-ValidDecks/07.txt
(Ingresa un número entre 1 y 7)
INPUT: 2

-----------------------------------------
Comienza el turno de HHH.
-----------------------------------------
HHH: 0F, tiene 11 cartas en la mano y 49 en el arsenal.
CHRIS JERICHO: 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: 4

-----------------------------------------
Felicidades, gana CHRIS JERICHO.
```

Notar que en el ejemplo anterior `HHH` y `Jericho` tienen el mismo `superstar value` por lo que normalmente se elegiría al azar quién parte. Pero para que los tests sean determinísticos partirá jugando la superestrella cuyo mazo haya sido ingresado primero (en caso de empate). En este caso, el mazo de `HHH` fue ingresado primero y por eso parte jugando.

Nota también que en la tabla resumen siempre aparece primero el jugador que comenzó la partida:

```
-----------------------------------------
HHH: 0F, tiene 11 cartas en la mano y 49 en el arsenal.
CHRIS JERICHO: 0F, tiene 7 cartas en la mano y 53 en el arsenal.
-----------------------------------------
```

Finalmente, no consideraremos las `superstar abilities` en esta entrega. Por lo mismo, superestrellas como `Mankind` que normalmente roban dos cartas al inicio del turno robarán una sola carta. Tampoco hay que dar la opción de usar la habilidad de la superestrella en el menú de acciones posibles al iniciar un turno.

Formato mazos

Los mazos son archivos de texto donde la primera línea indica la superestrella y las siguientes líneas tienen los nombres de las cartas que contiene el mazo. Los mazos están ordenados y no deben ser revueltos al inicio de la partida (eso permite que podamos corregir el proyecto de manera automática). Por ejemplo, acá se muestra un mazo de `Kane` con sus 60 cartas:

```
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
```

Formato cartas

La información de las cartas del juego están en el archivo: `cartas.json`. Por cada carta se indica su título, tipo, subtipos, fortitude, daño, stune value y efecto.

```
[
    {...},
    {
        "Title": "Choke Hold",
        "Types": [
            "Maneuver"
        ],
        "Subtypes": [
            "Submission",
            "Heel"
        ],
        "Fortitude": "6",
        "Damage": "5",
        "StunValue": "0",
        "CardEffect": "When successfully played, opponent must discard 1 card."
```

```
    },
    {...}
]
```

En esta entrega los datos más relevantes son el título de la carta, pues identifican las cartas que están en un mazo, y sus subtipos. Dependiendo de los subtipos de una carta existen restricciones que definen si un mazo es válido o inválido. Por ejemplo, no se pueden tener cartas con los subtipos `heel` y `face` a la vez en un mazo. El resto de restricciones son detalladas en la Sección 3 del documento del proyecto.

Input-Output

Para facilitar que el input-output de tu programa sea consistente con los test cases te proveemos con la clase `View` que contiene todos los métodos que necesitas para generar el output esperado por los test cases. En particular, incluye los siguientes métodos:

- `SayThatDeckIsInvalid()`: Indica que el mazo es inválido.
- `SayThatATurnBegins(string superstarName)`: Indica que el turno de `superstarName` comienza.
- `ShowGameInfo(PlayerInfo player1, PlayerInfo player2)`: Muestra la información resumen del estado del juego (el fortitude de cada jugador y el número de cartas en su mano y arsenal). Primero se muestra la información de `player1` y luego la de `player2`.
- `CongratulateWinner(string winnersName)`: Muestra el mensaje que felicita al ganador del juego.
- `AskUserToSelectDeck(string folder)`: Muestra el menú para elegir un mazo y retorna el path al mazo elegido por el usuario.
- `AskUserWhatToDoWhenItIsNotPossibleToUseItsAbility()`: Muestra el menú con alternativas de cosas que puede hacer el usuario en su turno. Retorna la acción elegida por el usuario. En esta entrega, el usuario siempre elige rendirse.

Rúbrica

Para evaluar tu entrega usaremos 3 grupos de test cases. Los primeros dos grupos, llamados `TestValidDecks` y `TestInvalidDecks`, vienen junto al código base de la entrega. El último grupo es un conjunto de test cases secretos. Para convertir el porcentaje de test cases pasados en una nota se hace lo siguiente. Cada grupo de test cases vale en total 2 puntos. Los primeros 1.5 puntos salen del porcentaje de test cases pasados. Por ejemplo, si pasas el 50 % del grupo `TestValidDecks` eso vale 0.75 puntos. Después existe un 0.5 extra por pasar todos los test cases del grupo (estos 0.5 son un todo o nada). En resumen, la rúbrica es la siguiente:

- **[1.5 punto]** Porcentaje de test cases pasados en `TestValidDecks`.
- **[0.5 punto]** Pasar todos los test cases en `TestValidDecks`.
- **[1.5 punto]** Porcentaje de test cases pasados en `TestInvalidDecks`.
- **[0.5 punto]** Pasar todos los test cases en `TestInvalidDecks`.
- **[1.5 punto]** Porcentaje de test cases secretos pasados.
- **[0.5 punto]** Pasar todos los test cases secretos.

Por ejemplo, digamos que tu entrega pasa todos los test cases `TestValidDecks` y `TestInvalidDecks`, y el 80 % de los test cases secretos. Entonces tu nota será un 6,2.

Importante: No está permitido modificar los test cases ni el proyecto `RawDeal.Tests`.