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

Entrega 1: Fire Emblem

Francisco Ignacio Gazitúa Requena
Cristian Andrés Hinostroza Espinoza
Estefania Mata Uri-Uri Pakarati Cofré

Introducción

En esta entrega, debes implementar el flujo básico del juego, lo cual abarca la totalidad del juego sin considerar ninguna de las habilidades (desde la Sección 1 hasta la Sección 6 del enunciado del proyecto). Esto incluye implementar:

1. Validación del setup del juego
2. El flujo de turnos de los jugadores
3. Todas las unidades
4. Cálculos de daño considerando tipos de armas, tipos de daño y ventaja del triangulo de armas
5. Flujo completo del combate, considerando todos los ataques (ataque, contraataque y Follow-Up) y cuando termina un combate.

Test cases

El código base incluye todos los tests que debes completar hasta el final del proyecto, sin embargo, para esta entrega solo debes pasar los siguientes tests:

- E1-BasicCombat
- E1-InvalidTeams

Los tests de esta entrega, a grandes rasgos, comprueban que se cumplan los siguientes escenarios:

El primer escenario consiste en que se le pide al usuario elegir los equipos del juego (dentro de un set de posibilidades) y este elige un grupo que contiene al menos un equipo inválido. En ese caso, se muestra un mensaje indicando que la selección es inválida y termina el programa, tal como muestra el siguiente test case:

```
Elige un archivo para cargar los equipos
0: 01.txt
1: 02.txt
2: 03.txt
3: 04.txt
INPUT: 3
Archivo de equipos no válido
```

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 ambos equipos ingresados sean válidos. Si esto ocurre, el juego comenzará, momento en el que los jugadores podrán escoger qué unidades utilizarán para atacar y defender según sea el caso. El juego continuará normalmente hasta que uno de los jugadores quede sin unidades vivas en su equipo. Este caso será explicado con más detalle en las siguientes secciones.

Formato equipos

Los equipos son archivos de texto donde se encuentra la información de cada uno de los equipos del juego. El archivo siempre empezará con una línea que indica que comienza la sección del primer jugador, seguida inmediatamente de los nombres de sus unidades. Luego de las unidades del primer jugador habrá una línea indicado el inicio de la sección del segundo jugador, seguida inmediatamente por el nombre de las unidades de su equipo. Es importante destacar que el archivo siempre contará con un espacio en su última línea.

```
Player 1 Team
Lyn (Def/Res Push)
Player 2 Team
Edelgard (Swift Sparrow,Eclipse Brace)
Veronica
Sigurd
```

Es importante destacar que, en caso que una unidad tenga habilidades, estás se encontrarán en la misma línea del nombre de la unidad entre paréntesis. Si una unidad tiene más de una habilidad, estas se encontrarán dentro del paréntesis delimitadas por una coma sin espacio entre esta y los nombres de las habilidades.

En el ejemplo anterior, Lyn tiene la habilidad Def/Res Push y Edelgard tiene las habilidades Swift Sparrow y Eclipse Brace

Formato Habilidades

La información de las habilidades del juego está en el archivo: `skills.json`. Por cada habilidad se indica su nombre y su descripción.

```
[
    {...},
    {
        "Name": "Sword Agility",
        "Description": "Si la unidad usa espada, otorga Spd+12 a un costo de Atk-6 durante el combate"
    },
    {
        "Name": "Tome Precision",
        "Description": "Otorga Atk/Spd+6 al usar magia."
    },
    {
        "Name": "Resonance",
        "Description": "Si la unidad usa magia, si HP de la unidad >=2, la unidad pierde 1 HP al inicio del combate e inflige +3 de daño por ataque durante el combate."
    },
    {
        "Name": "Lance Power",
        "Description": "Otorga Atk+10 a un costo de Def-10 al usar una lanza"
    },
    {...}
]
```

En esta entrega, el dato más importante de las habilidades es su título, pues identifica las habilidades que tiene equipada una unidad. Recordar que una unidad no puede tener equipada la misma habilidad más de una vez.

Formato Unidades

La información de las unidades del juego se encuentra en el archivo `characters.json`. Las unidades se muestran en el siguiente formato:

```
[
  {...},
  {
    "Name": "Marth",
    "Weapon": "Sword",
    "Gender": "Male",
    "DeathQuote": "Forgive me, my friends...",
    "HP": "54",
    "Atk": "62",
    "Spd": "53",
    "Def": "43",
    "Res": "37"
  },
  {
    "Name": "Tiki",
    "Weapon": "Magic",
    "Gender": "Female",
    "DeathQuote": "No...I don't want to be...alone...",
    "HP": "59",
    "Atk": "61",
    "Spd": "46",
    "Def": "47",
    "Res": "39"
  },
  {
    "Name": "Alm",
    "Weapon": "Bow",
    "Gender": "Male",
    "DeathQuote": "No... Not when we were...so close... Forgive me, Celica...",
    "HP": "54",
    "Atk": "62",
    "Spd": "51",
    "Def": "36",
    "Res": "32"
  },
  {...}
]
```

Los atributos importantes de cada unidad serán su nombre, arma, género y *stats*.

Output del juego

Cuando el archivo de equipos seleccionado no es válido, se imprimirá inmediatamente después de la selección del archivo un mensaje indicando la situación, de la siguiente manera:

```
Elige un archivo para cargar los equipos
0: 01.txt
1: 02.txt
2: 03.txt
3: 04.txt
INPUT: 3
Archivo de equipos no válido
```

En el caso contrario, se preguntará al primer jugador qué unidad desea usar para atacar e inmediatamente después se le preguntará al segundo jugador qué unidad usará para defenderse, esto de la siguiente manera:

```
Player 1 selecciona una opción
0: Hector
INPUT: 0
Player 2 selecciona una opción
0: Marth
1: Eliwood
INPUT: 0
```

Inmediatamente después de finalizar la selección de las unidades, el combate ocurrirá de forma automática. En el combate, se anunciará quién inicio el combate, qué unidad tiene ventaja del triángulo de armas (en caso que alguna la tenga), cada uno de los ataques y, finalmente, el HP restante de cada unidad, siguiendo el siguiente formato:

```
Player 1 selecciona una opción
0: Hector
INPUT: 0
Player 2 selecciona una opción
0: Marth
1: Eliwood
INPUT: 0
Round 1: Hector (Player 1) comienza
Marth (Sword) tiene ventaja con respecto a Hector (Axe)
Hector ataca a Marth con 10 de daño
Marth ataca a Hector con 22 de daño
Marth ataca a Hector con 22 de daño
Hector (17) : Marth (44)
```

Luego de que finalice el combate, se volverán a escoger las unidades para pelear, pero esta vez, el segundo jugador escogerá a la unidad que ataca y el primer jugador a la unidad que se defiende. El combate continuará intercalando quién ataca y quién defiende hasta que un jugador pierda a todas sus unidades. Un ejemplo de lo anterior:

```
Player 1 selecciona una opción
0: Hector
INPUT: 0
Player 2 selecciona una opción
0: Marth
1: Eliwood
INPUT: 0
Round 1: Hector (Player 1) comienza
Marth (Sword) tiene ventaja con respecto a Hector (Axe)
Hector ataca a Marth con 10 de daño
Marth ataca a Hector con 22 de daño
Marth ataca a Hector con 22 de daño
Hector (17) : Marth (44)
Player 2 selecciona una opción
0: Marth
1: Eliwood
INPUT: 0
Player 1 selecciona una opción
0: Hector
INPUT: 0
Round 2: Marth (Player 2) comienza
Marth (Sword) tiene ventaja con respecto a Hector (Axe)
Marth ataca a Hector con 22 de daño
Marth (44) : Hector (0)
Player 2 ganó
```

Si ninguna unidad tiene ventaja del triángulo de armas o ninguna unidad puede hacer un Follow-Up, se anunciará de la siguiente manera:

```
Round 2: Dimitri (Player 2) comienza
Ninguna unidad tiene ventaja con respecto a la otra
Dimitri ataca a Claude con 35 de daño
Claude ataca a Dimitri con 14 de daño
Ninguna unidad puede hacer un follow up
Dimitri (39) : Claude (19)
```

Si una unidad es derrotada antes que se completen todos los ataques, se interrmpirá el combate y se mostrará de inmediato el estado final de la ronda. Esto de la siguiente manera:

```
Round 2: Dimitri (Player 2) comienza
Ninguna unidad tiene ventaja con respecto a la otra
Dimitri ataca a Claude con 35 de daño
Dimitri (39) : Claude (0)
```

En el ejemplo anterior, el HP de Claude llegó a 0 en el primer ataque de Dimitri, por lo que no sucede el primer ataque de Claude y no se anuncia que ninguna unidad puede hacer un Follow-Up

Input-Output

Para que el input-output de tu programa sea consistente con los test cases te proveemos con la clase `View` que contiene métodos básicos:

- `Readline()`: Solicita un string al usuario y retorna el `string` correspondiente.
- `WriteLine(string message)`: Muestra `message` al usuario.

Puedes agregar los archivos que desees al proyecto `Fire-Emblem-View` y editar el archivo `View`. Sin embargo, no puedes editar los otros archivos del proyecto.

Rúbrica

Para evaluar tu entrega usaremos 2 grupos de test cases, llamados `E1-BasicCombat` y `E1-InvalidTeams`, vienen junto al código base de la entrega. Para convertir el porcentaje de test cases pasados en una nota se hace lo siguiente. El primer grupo de test cases vale en total 3.5 puntos y el segundo 1.5 puntos. Además, existe un 0.5 extra por pasar todos los test cases del grupo (estos 0.5 son un todo o nada). Por ejemplo, si pasas el 50 % del grupo `E1-BasicCombat` eso vale 1.75 puntos. En resumen, la rúbrica es la siguiente:

- **[3.5 punto]** Porcentaje de test cases pasados en `E1-BasicCombat`.
- **[0.5 punto]** Pasar todos los test cases en `E1-BasicCombat`.
- **[1.5 punto]** Porcentaje de test cases pasados en `E1-InvalidTeams`.
- **[0.5 punto]** Pasar todos los test cases en `E1-InvalidTeams`.

Por ejemplo, digamos que tu entrega pasa todos los test cases `E1-InvalidTeams` y el 80 % de los test cases `E1-BasicCombat`. Entonces tu nota será un 5,8.

Importante: No está permitido modificar los test cases ni el proyecto `Fire-Emblem.Tests`. Hacerlo puede conllevar una penalización que dependerá de la gravedad de la situación