Tarea 2 - JavaUNO

Profesor: Alexandre Bergel
Auxiliares: Javier Espinoza
Eduardo Riveros
Ayudantes Marco Caballero
Franco Cruces
Matilde Rivas
Juan-Pablo Silva
Fecha de Entrega: 21 de Octubre del 2017

El Juego

UNO es un juego de cartas para dos o más jugadores, jugado con un mazo especial, en el que el objetivo del jugador es quedarse sin cartas en la mano antes que sus rivales, siguiendo un conjunto de reglas de fácil aprendizaje.

El mazo usado para jugar UNO cuenta con l08 cartas, las cuales se muestran en la figura.

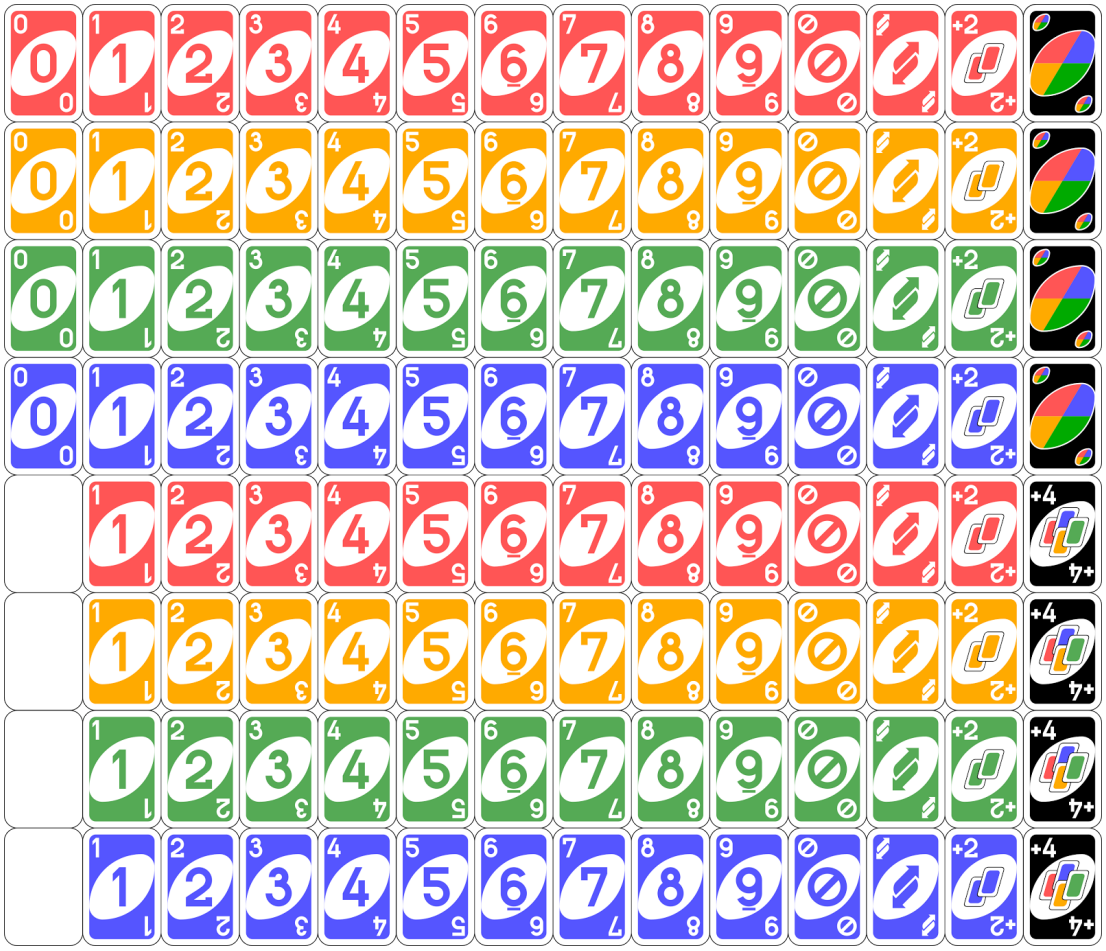

Figura 1.- Cartas en mazo de UNO. (Vectorizado por el usuario de Wikipedia Dmitry_Fomin)

Tipos de Cartas

Las cartas se pueden dividir en 3 grupos fundamentales:

1. **Cartas Numéricas**: Corresponden a las cartas que representan un **número**, del 0 al 9. Además, todas ellas tienen un color asociado, el cual puede ser **azul, verde, rojo** o **amarillo**.
2. **Cartas de Acción:** Corresponden a cartas de colores con **símbolos** en ellas. Permiten realizar acciones especiales que modifican la dinámica del juego. En este grupo se encuentran las cartas de **"saltar turno"**, **"invertir dirección de juego"** y **"robar 2."**
3. **Cartas Comodines** *(Wild)*: Corresponden a cartas de **color negro**. Al igual que las cartas de acción, permiten realizar acciones especiales que modifican la dinámica del juego, pero poseen libertades y restricciones distintas a las *Cartas de Acción*. En este grupo se encuentran las cartas de **"comodín de color"** y **"comodín + robar 4"**

Reglas Básicas del Juego

El juego se suele jugar de esta forma[1]:

1. Antes de partir, se revuelve el mazo y se toma la primera carta de él, para colocarla en la pila de descarte:
    a. Si esta carta es una carta **comodín** o **comodín + roba 4**, se devuelve al mazo y se repite este paso.
    b. Si es cualquier otra carta, sus efectos aplican como si se hubiese jugado por un jugador ficticio, anterior al primero.
2. Los jugadores parten con **7 cartas cada uno**.
3. Por turnos, **se puede jugar una carta numérica del mismo color o número que la carta jugada anteriormente**. También se pueden jugar las siguientes cartas especiales:
    a. La carta de acción **saltar turno** se puede jugar si anteriormente se jugó una carta del mismo color o símbolo. Su habilidad permite saltar el turno del jugador siguiente.
    b. La carta de acción **invertir dirección de juego** se puede jugar si anteriormente se jugó una carta del mismo color o símbolo. Su habilidad permite invertir la dirección de juego actual, en caso de 3 o más jugadores. Si solo hay 2 personas jugando, se utiliza de la misma forma que una carta de **saltar turno**.
    c. La carta de acción **robar 2** se puede jugar si anteriormente se jugó una carta del mismo color o símbolo. Su habilidad provoca que el jugador siguiente tenga que robar 2 cartas, perdiendo la posibilidad de jugar en su turno.
    d. La carta **comodín de color** se puede jugar en cualquier momento. Al jugar una de ellas, es necesario que el jugador le asigne un color, el cual servirá para definir las cartas jugables en el turno siguiente.
    e. La carta **comodín + roba 4** se puede jugar en cualquier momento y tiene el mismo efecto que **comodin de color**, sumado a provocar que el siguiente jugador tenga que robar 4 cartas, perdiendo la posibilidad de jugar en su turno.

[1] Para el conocedor de las reglas oficiales del UNO, informamos que como Equipo Docente nos tomamos algunas libertades artísticas para hacer el juego más fácilmente implementable por ustedes. Estas libertades están subrayadas. También, eliminamos la regla oficial que verifica si una jugada de **comodín + roba 4** es *legal* o no lo es, de forma de agilizar el juego.

4. En caso de no tener ninguna carta jugable en su **mano**, el jugador actual debe **robar una carta del mazo**. Si esa carta es jugable, puede jugarla inmediatamente. Si no es jugable, terminará su turno.
5. Si un jugador queda solamente con una carta en su **mano,** deberá gritar "UNO!" lo antes posible. Si otro jugador se da cuenta que él no gritó "UNO!" al momento de tener solo una carta, puede gritar "UNO!" en su lugar, haciendo que el primero robe 7 cartas del **mazo**.
6. Cuando las cartas del **mazo** se acaban, se revuelven todas menos la primera de la **pila de descartes** y se asignan como **mazo**.
7. Gana y termina el juego el primer jugador en quedarse sin cartas en la **mano**.

Estrategia de Juego Básica: RandomPlayer

Para poder probar una mejor interacción en el juego, se les solicitará implementar un **jugador virtual** que juegue cartas válidas en cada turno de forma aleatoria. Este jugador también tendrá que elegir un color aleatorio para jugar en caso de usar un comodín.

Requisitos Tarea 2

Para la tarea 2, implementarán **las reglas básicas** del juego UNO mencionadas en este enunciado.:

1. Su tarea debe permitir jugar UNO con todas **reglas básicas** a entre 2 y 10 jugadores (ya sean humanos o virtuales de tipo *RandomPlayer*) a través de la línea de comandos. El juego debe iniciar con un jugador de tipo *HumanPlayer* y 3 de tipo *RandomPlayer*.
    a. Les recomendamos que los tipos de jugadores sean de fácil extensión para facilitarles la implementación de más jugadores en la próxima tarea.
2. Deben implementar un *Jugador Virtual* que juegue cartas al azar, todas con la misma probabilidad (Asumiendo que *Math.random()* posee una distribución uniforme), robando una carta solo si es necesario. También debe elegir colores al azar al momento de jugar una **carta comodín**.
3. Para esta tarea, el **mazo oficial en el juego principal** debe poseer las cartas en las cantidades especificadas en la figura de la página 1. También deben permitir la implementación nuevos tipos de mazos de forma fácil y extensible usando la interfaz *model.card.deck.ICardStrategy*. Para probar lo anterior, les pediremos que agreguen dos mazos aparte del **mazo oficial** a su código**:**
    a. Un mazo solo con cartas numéricas
    b. Una clase que permita construir mazos para testing, agregando las cartas de a una.
4. Al comienzo de cada turno, el juego debe ir indicando **el jugador actual**. Además, cada vez que se juega una carta, se debe mostrar tanto la carta jugada como un texto que explique su efecto.
5. En el turno del jugador humano, además de lo anterior, se deben mostrar las **acciones que puede realizar el jugador,** numeradas para que éste solo necesite ingresar el número de la opción que desea usar. Las acciones son:
    a. Jugar una carta de su mano o robar una carta del mazo y jugarla si es posible.
    b. Elegir un color luego de jugar algún **comodín.**
6. La lógica del juego **debe detectar automáticamente cuando un jugador tiene una carta**, gritando "UNO!" en su lugar. Por lo tanto, no existirán (por ahora) las penalizaciones asociadas a esa regla.

   a. El grito de "UNO!" debe realizarse solo una vez por cada vez que un jugador llega a tener una carta. Esto quiere decir que si un jugador tiene una carta dos turnos seguidos (sin pasar por tener más de una carta en algún momento), se debe gritar "UNO!" solo una vez.
   b. Si un jugador tiene una carta, luego más de una y luego vuelve a tener una carta, se debe gritar "UNO!" en ambas ocasiones (la primera y la segunda vez que el jugador tuvo una carta)
7. El juego **debe detectar automáticamente cuando un jugador gana**, declarando el nombre del jugador ganador y terminando el programa.
8. **Es necesario seguir el orden de los paquetes entregados en el código fuente de la tarea, así como también las interfaces y clases ya creadas**. A partir de ellas, ustedes pueden (y deben) extender, siempre cuidando no romper la compatibilidad con lo entregado inicialmente. La explicación de cada paquete e interfaz puede encontrarse en los comentarios de tipo Javadoc adjuntos a ellas.
9. Les recordamos nuevamente que el enfoque de la tarea debe ser **aplicar los contenidos vistos en clase (todos, no solo los de las últimas semanas)** por sobre que simplemente ésta funcione.

Requerimientos Adicionales

Además de una implementación basada en las buenas prácticas y patrones de diseño vistos en clases, usted debe considerar el cumplimiento de los puntos siguientes:

- **Cobertura**: Cree los test unitarios, en JUnit, que sean necesarios para alcanzar un **90%** de coverage por paquete. Estos tests deben estar en el paquete *test* de su proyecto.
- **Javadoc:** Cada clase y método público debe estar debidamente documentado, siguiendo las convenciones de Javadoc.
- **Resumen:** Debe entregar un archivo en formato *.pdf* que contenga una portada con los datos del curso, su nombre, un link al repositorio en que trabajó y un diagrama UML por paquete del proyecto (excluyendo *tests*).
- **Coding Style:** El código y la documentación debe seguir las convenciones de Google.
- **GitHub:** Debe mantener actualizado su repositorio de GitHub, ya que de él se extraerá la tarea que será revisada.

Evaluación

La evaluación utilizará la siguiente rúbrica::

- **Funcionalidad (Descuento máximo de 1,5 puntos):** Corresponde al rendimiento en tests que demuestran que la tarea hace lo que se pide en este enunciado.
- **Diseño (Descuento máximo de 2 puntos)**: Corresponde al uso de patrones de diseño y materia vista en clases de forma adecuada a cada desafío presentado por la tarea.
- **Coverage (Descuento máximo de 1 punto)** correspondiente a cubrir con tests hechos por ustedes mismos, al menos **el 90% del código del paquete** *model*.

- **Resumen (Descuento máximo de 0,5 puntos)** cumpliendo lo mencionado en el punto *Requerimientos Adicionales*.
- **Code Smells (Descuento máximo de 1 punto)** correspondiente a no presentar los *code smells* vistos en clases, usar *Javadoc* donde corresponda y a utilizar el *code style* solicitado.

Recuerden que no es suficiente para una buena evaluación que la tarea funcione, ya que el mayor énfasis en la evaluación corresponde a que estén aplicando los conocimientos vistos en clases.

Entrega

1. Para evitar las confusiones generadas en la tarea anterior, las entregas se realizarán de la siguiente forma:
    a. La implementación de su tarea se entregará **solamente por GitHub**, correspondiendo a la entrega de la carpeta raíz del proyecto en Eclipse (no olviden agregar el archivo *.gitignore* de Material Docente a su proyecto)
    b. El Resumen se entregará **solamente por U-Cursos**.
    c. Tanto la implementación como el informe tienen de plazo de entrega máximo el estipulado en la tarea de U-Cursos.
2. El repositorio **privado** de su tarea debe llamarse **cc3002-tarea2.** De esta forma, si su usuario en Github es *adderou*, su tarea se ubicará en *https://github.com/adderou/cc3002-tarea2*
3. Recuerden invitar a la cuenta *CC3002EquipoDocente* como integrante del repositorio para poder revisar la tarea.
4. **Solo podrán realizar commits hasta la hora límite de entrega de la tarea en U-Cursos.** En caso que detectemos commits en horas posteriores, se les evaluará con nota mínima.

Recomendaciones Finales

- **La tarea 3 requerirá gran parte del código implementado en la tarea 2**, por lo cual se les recomienda hacerla con tiempo y completarla lo más que puedan, para no tener una mayor cantidad de trabajo que la necesaria en la tarea 3.
- Se les solicita que todas las consultas referentes a la tarea las hagan en primera instancia a través del **foro de U-Cursos**. En caso que nos demoremos mucho en contestar (1 día hábil o más), pueden hacernos llegar un correo a los auxiliares o ayudantes para avisarnos de sus consultas.