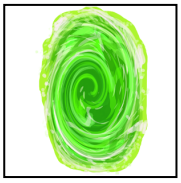
Infinity Game.

Tablero.

Se le pide construir un nuevo juego de mesa titulado “Infinity game”. El juego consiste en un tablero, el cual se debe generar automáticamente luego de que el usuario ingrese el tamaño de este, que no podrá ser inferior a 20 casillas. Existen los siguientes tipos de casilla:

Casilla de partida:
Es la casilla en que inicialmente todos los jugadores se posicionan, en caso de que un jugador caiga en esta casilla luego de iniciado el juego, su salud se verá restaurada a su valor inicial.

Casilla en blanco:
Caer en esta casilla significa terminar el turno actual y pasar al siguiente.



Casilla portal:
Caer en esta casilla implica transportar al jugador a otro portal escogido aleatoriamente.

Casilla de salud:
Aumenta o reduce los puntos de salud del jugador de forma aleatoria en un valor entre 1 y 3.

Casilla de desafío:
Se escogerá un desafío aleatoriamente, los cuales consistirán en avanzar/retroceder (entre 1 y 5 casillas) o agregar/quitar vida a los demás jugadores (entre 1 y 4 puntos de salud).

Casilla final:
El primer jugador en caer en esta casilla será el ganador y el juego terminará.

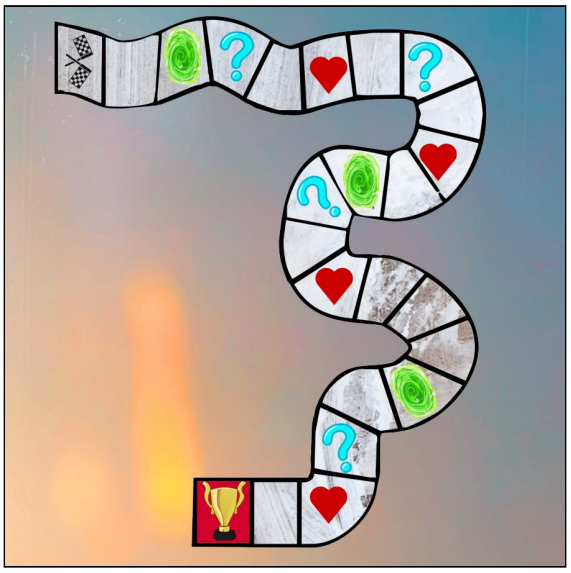


Figura I. Ejemplo de tablero

Jugadores.

Infinity Game puede ser jugado por un mínimo de 1 jugador (modo solitario) y un máximo no definido; el usuario es quien se encarga de especificar la cantidad de jugadores que tendrá la partida. Cada jugador tendrá un Nickname y dispondrá inicialmente de 15 puntos de salud, la cual corresponde a la salud máxima, esto implica que la salud no puede sobrepasar en ningún momento dicho valor. Por otro lado, el jugador es quien llevará la cuenta de su posición actual en el tablero. Inicialmente la forma de la representar a los jugadores será como indica la siguiente tabla (los nicknames no se pueden repetir):

NickName	Puntos de salud	Posición actual
Pedro	15	0
Juan	12	5

Reglas generales.

Al principio de cada turno, los jugadores lanzan dos dados tradicionales (de 6 caras), siendo la suma de las caras resultantes la cantidad de casillas que se moverá en el tablero.

Habrán dos formas de terminar el juego: en primer lugar, si un jugador cae en la casilla final, este ganará el juego quedando en primer lugar, mientras que los demás lugares se ordenarán según los puntos de salud de los jugadores restantes. La otra forma consiste en el último jugador en pie; si todos los demás jugadores mueren (alcanzan 0 puntos de salud), el jugador vivo alcanzará el primer lugar, mientras que los demás lugares se ordenarán por

el orden de muerte. Para ordenar a los jugadores se utiliza un podio, el cual guarda los datos de los jugadores que terminen el juego. La siguiente tabla ilustra su funcionamiento:

1er lugar	2do lugar	….	n° lugar
Jugador victorioso	Jugador en segundo lugar	...	Jugador en último lugar

Notar que cuando un jugador muere, queda automáticamente en el último lugar; si ese puesto está ocupado, pasa a penúltimo lugar y así sucesivamente.

Formato de entrega.

Debe crear un repositorio en GitHub y adjuntar en el campus el link de este.

Tarea 1. Fecha de entrega: 16/09/2018

Crear el menú básico del programa según las especificaciones del enunciado, considerando la representación de los jugadores y del tablero. Adicionalmente, construir los siguientes métodos:

-Lanzar los dos dados y devolver el resultado (la suma de ambas caras).

-Generar tablero. Debe recibir como parámetro la cantidad de casillas y devolver un arreglo que represente al tablero (sugerencia: el tablero se puede representar como un arreglo de tipo Char, de esta manera, cada tipo de casilla puede representarse por una letra).

-Procesar la caída de un jugador en la casilla de desafío. Para esto, el método debe recibir el índice del jugador que cae en la casilla y modificar los valores del o los jugadores afectados (ver detalles de los desafíos en el enunciado).

-Procesar la caída de un jugador en la casilla portal. Para esto, el método debe recibir el índice del jugador que cae en la casilla y modificar la posición del jugador hacia la posición de otro portal en el tablero.

-Procesar la caída de un jugador en la casilla de salud. Para esto, el método debe recibir el índice del jugador que cae en la casilla y modificar el valor de su salud según corresponda.

Tarea 2. Entrega 30/09/2018

A partir del resultado de la tarea 1, se le pide agregar las pruebas unitarias los siguientes métodos:

- Lanzar dado, en el cual debe asegurarse que el valor resultante pertenece al rango posible correspondiente a la suma de las dos caras resultantes.
- Caer en la casilla salud, donde debe probar que la salud del jugador sea modificada como se indica en el enunciado.
- Caer en la casilla portal, en el cual debe probar que al caer en dicha casilla, la posición del jugador afectado coincida con otra casilla portal.

OBS:

- Para los dados debe considerar que el resultado se encuentre entre 2 y 12.
- Para la casilla salud, considerar que la salud no puede ser menor a 0, esto se debe contemplar como posibilidad dentro de la prueba unitaria.
- La casilla de portal, se debe probar que al cambiar al jugador de posición, la nueva ubicación de este corresponda a otra casilla de portal.

Tarea 3. Fecha de entrega: 7/10/2018

- Validar el ingreso del usuario al momento de crear el tablero (cuando se le pregunta el tamaño, este debe ser mayor o igual a 20, adicionalmente, debe anteponerse al caso en que se le ingrese cualquier cosa menos un número entero).
- Hacer lo mismo con el ingreso de la cantidad de jugadores. En este caso, debe ser a lo menos un jugador, además que se debe contemplar el caso en que el usuario no ingrese un valor numérico.
- Cuando sea el turno de un jugador, este tendrá dos opciones: lanzar dados o meditar.

- Si un jugador escoge la opción de meditar, este podrá elegir su posición dentro de 3 casillas a la redonda de su posición actual, adicionalmente recuperará 1 punto de salud.
- Durante todo el juego, cada jugador dispondrá de un total de 5 opciones de meditar.
- Si un jugador que no cuenta con más opciones de meditar escoge esta opción, mantendrá su posición pero se le restará un punto de salud.
- Debe anteponerse a todas las situaciones posibles, por ejemplo, ¿Qué hará si el usuario escoge una casilla fuera del tablero?

OBS: Considere que el juego debe seguir "funcionando" aún cuando se presenten situaciones inesperadas. En caso de una situación extrema de no poder seguir haciéndolo, debe informar al usuario y "cerrarse honorablemente".

Tarea 4. Fecha de entrega: 21/10/2018

Dado el enunciado del problema y considerando el avance hasta la fecha, construya el problema en base a una estructura de clases.

pauta:
https://docs.google.com/document/d/1hXVs7CLDRpZiO49ubr9hFEx94fIbtiDoTV2r8YWKI9s/edit?usp=sharing

Tarea 5. Fecha de entrega 28/10/2018

Utilizando como cimiento lo elaborado hasta el momento:

- Construya el diagrama de clases en Visual Paradigm en base a lo avanzado en la tarea anterior, considerando los métodos y atributos de cada clase. Indique en la herramienta las relaciones de dependencia que encuentre.
- Update: se incluye un objeto mágico al juego: reliquia. Al momento de que un jugador lanza los dados, existe 1% de probabilidad de activar la reliquia para restaurar la vida original del jugador; en el caso de que al lanzar los dados el resultado sea 12 (doble seis), la probabilidad sube a un 50%. Por otro lado, si un jugador cae en la casilla de salud, éste tendrá un 5% de probabilidad de activar la reliquia pero en este caso para conseguir una meditación extra a su colección.
- Entrega: Considere todo lo anterior e inclúyalo en su diagrama de clases considerando las relaciones de dependencias nuevas que encuentre. Debe subir a su repositorio de github una imagen de su diagrama de clases.

pauta:
https://docs.google.com/document/d/1kZkQICz5qoD8IAAKClf_8s0ub8MRjKJR2PYrObFNgzQ/edit#

Tarea 6. Fecha de entrega: 4/11/2018

En caso de que corresponda, agregue a su diagrama UML las relaciones de asociación que logre identificar.

- A partir de ahora el juego es custodiado por un Guardián (¡y el guardián custodia el juego!). El guardián consta de una salud equivalente a la suma de la salud de todos los jugadores, además, por cada turno, tendrá un 2% de probabilidad de lanzar su habilidad especial: furia, la cual le quita 1 punto de salud a todos los jugadores, además se cura 2 puntos de salud. Al lanzar los dados, los jugadores le quitan 1 punto de salud al guardián; en caso de que al lanzar los dados el resultado sea 12, se le quitan 3 puntos. Si al lanzar los dados un jugador acaba con la vida del guardián, éste ganará automáticamente el juego.
- Considere lo anterior para su diagrama de clases, teniendo en cuenta las nuevas relaciones de asociación y/o dependencia (si es que las hay) y suba a su repositorio de github una imagen del diagrama.

Pauta:
https://docs.google.com/document/d/1hPz_GUTI5_xB54MuG4OwOF15p65dgLJpbVCJFF4nqOM/edit#heading=h.ve2578ycopqq

Tarea 7. Fecha de entrega: 11/11/2018
Dado el avance en el modelo UML hasta ahora, considerando las relaciones de dependencia y asociación:

- Agregue al diagrama las relaciones de agregación y/o composición que logre identificar. Si cree que en alguna relación de asociación identificada en la tarea anterior tiene más sentido aplicar agregación/composición, haga los cambios pertinentes en el diagrama.
- Para la clase Casilla, aplique el concepto de Enumeration para el atributo tipoCasilla (salud, desafío, etc.). Agregue los cambios a su diagrama de clases.
- Genere código con la herramienta de Visual Paradigm a partir de su diagrama de clases y programe la lógica de las clases que aún no ha implementado.
- Suba a su repositorio de github el proyecto Java con las nuevas clases programadas y una imagen del diagrama recientemente modificado.

Pauta:
https://docs.google.com/document/d/1uBsRjDPjOwoxKPHD0OZjGPqCbbJOpmgr7ptcds7H_Bo/edit#heading=h.910egnq6ttv6

Tarea 8. Fecha de entrega: 25/11/2018
A partir de ahora existen dos clases de jugadores:

- **Guerrero**: su característica particular es que consta con 20 puntos de salud máxima. Por otra parte, cuenta con una habilidad especial llamada "**enfurecerse**", la cual consiste en elegir un a uno de los demás jugadores para inflingir un punto de daño físico; cada uso del ataque especial descuenta un punto de furia. La **furia** consiste

en la fuente de poder de este tipo de jugadores, originalmente cada jugador guerrero dispone de 5 puntos de furia.

- **Mago:** su característica particular es que consta con 7 meditaciones. Por otra parte, cuenta con una habilidad especial llamada "**concentración**", la cual consiste en inflingir dos puntos de daño mágico hacia el Guardián del juego; cada uso del ataque especial descuenta un punto de maná. El **maná** es una fuente de energía para este tipo de jugadores, originalmente cada jugador mago dispone de 4 puntos de maná.
- Por cada turno, los jugadores pueden elegir la opción de lanzar su habilidad especial.
- Suba al repositorio github una imagen del diagrama resultante y la implementación de este (archivos java). Recuerde en su github subir los resultados a un subdirectorio "tarea 8" o crear un nuevo repositorio para la tarea, para hacer más sencilla su revisión.

pauta:
https://docs.google.com/document/d/1NDU2EImhef3GWv8z6nYRTrsBj8UgXHY8i1-gstEOufE/edit#heading=h.ww3ogqseh7b7

Tarea 9. Fecha de entrega: 2/12/2018
Para comenzar a rearmar el proyecto con orientación al diseño de tres capas (vista o presentación, lógica o de negocios y acceso a datos), se pide comenzar con la capa de vista o GUI y parte de la capa de lógica. Para esto, debe incluir las siguientes características:

- Creación del juego: pantallas de diálogo (recomendación: hacerlas con JOptionPane) con el usuario para el ingreso de los datos iniciales del juego, tales como la cantidad de casillas del tablero y jugadores con sus características.
- Ventana (JFrame) que contenga la visualización del tablero generado (como un conjunto de botones, por ejemplo).
- Sistema básico de turnos: se debe mostrar al jugador actual jugando (basta con un JLabel), el cual al apretar un botón "hacer jugada", se despliegue un conjunto de opciones posibles (lanzar datos, meditar, etc); al elegir una opción y dar a "aceptar", se debe pasar al siguiente jugador (ojo, solamente se pide hacer el sistema de turnos, no implementar aún las funcionalidades de cada opción que elija el jugador).
- Entrega: imagen del diagrama UML, junto al proyecto Java en su repositorio github. Recuerde hacer un subdirectorio "tarea 9" o bien otro repositorio para subir la tarea.

Pauta:
https://docs.google.com/document/d/1uQXZ6T4qtdVgjirpl1wIa_VsXfCVfrC8bSHPUGpl9E4/edit#

Tarea 10. Fecha de entrega: 9/12/2018

Siguiendo con el objetivo de lograr hacer un programa separado por 3 capas, se pide en esta ocasión:

- Conectar la capa de vista (GUI) con la capa de modelo, de esta manera se podrá, por ejemplo, hacer un sistema de turnos funcional, el cual se diseñó en la tarea anterior y ahora debería (entre otras cosas) añadir/restar vida, meditaciones, etc. a

los jugadores según corresponda (recordar que por cada turno, el jugador tendrá la opción de lanzar dado, meditar, lanzar habilidad especial, etc. tal como se ha definido durante el proyecto).

- Capa de datos: cada vez que se termine un juego, se debe hacer un registro en un archivo de texto indicando la posición de los jugadores en el podio, además de los de detalles de la fecha y hora en que se terminó el juego.
- Entrega: suba el diagrama UML junto al proyecto Java a su repositorio Github. Recuerde que debe subir la tarea en un subdirectorio llamado (por ejemplo) Tarea 10 o en un nuevo repositorio, para que así sea más fácil la revisión.