Práctica de curso 2025-26

El objetivo de la práctica es elaborar un programa Java que implemente el juego: **InforTactics UVa**

Descripción del Juego

InforTactics UVa es un juego inspirado en el famoso juego Clash Royale – Merge Tactics (tutorial rápido: https://www.youtube.com/shorts/otgWRBM_oOk). Clash Royale – Merge Tactics es un juego de defensa estratégica. En este juego, el jugador dispone de un tablero donde deberá colocar sus figuras para posteriormente combatir contra las figuras del enemigo. El objetivo principal es colocar las figuras de manera eficiente para eliminar todas las tropas enemigas antes de que nos eliminen las nuestras (ver Imagen 1). Cada figura tiene un porcentaje de ataque, un porcentaje de defensa, una característica que lo hace diferente del resto (p.ej., la arquera dispara a la figura más lejana) y un coste de elixir. Cuando el juego comienza, las figuras atacan de forma automática, por lo que la clave está en gestionar bien el espacio disponible y elegir adecuadamente tus tropas. Ganarás si consigues derrotar a todos las figuras del equipo rival.

Imagen 1. Descripción de las mecánicas importantes de Clash Royale – Merge Tactics.

InforTactics UVa implementará una versión más sencilla del juego, con una interfaz gráfica adaptada a la consola de comandos. Las mecánicas del juego implementadas son: (*) un tablero cuadrado de 6x6 en donde las 3 primeras filas estarán inicialmente ocupadas por las figuras del enemigo, y las 3 últimas, por nuestras figuras; (*) un elixir inicial de 8 puntos; y, (*) las diferentes figuras que podremos elegir (ver Imagen 3).

Requisitos de diseño del programa:

En esta práctica, se pide un programa que permita jugar a InforTactics UVa. Para ello, inicialmente el programa mostrará el menú del juego con diferentes opciones para los usuarios, tal y como se muestra en la Imagen 2. Entonces, se le pedirá al usuario que elija una opción. Si el usuario introduce una opción incorrecta, se le mostrará el mensaje "Opción no válida", se volverá a mostrar el menú, y se le pedirá de nuevo al usuario que elija una opción.

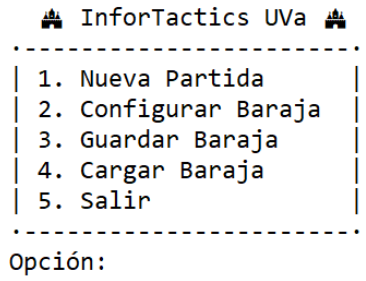


Imagen 2. Menú inicial del juego.

Si el usuario elige la **opción #5 (Salir)**, el programa mostrará por pantalla un mensaje de despedida y terminará.

Si el usuario elige la **opción #2 (Configurar Baraja),** el programa mostrará por pantalla: (1) el estado actual del tablero, (2) una tabla con los personajes a añadir y sus características, (3) el elixir restante, y (4) una línea para que el usuario elija su siguiente acción, tal y como se muestra en la Imagen 3.

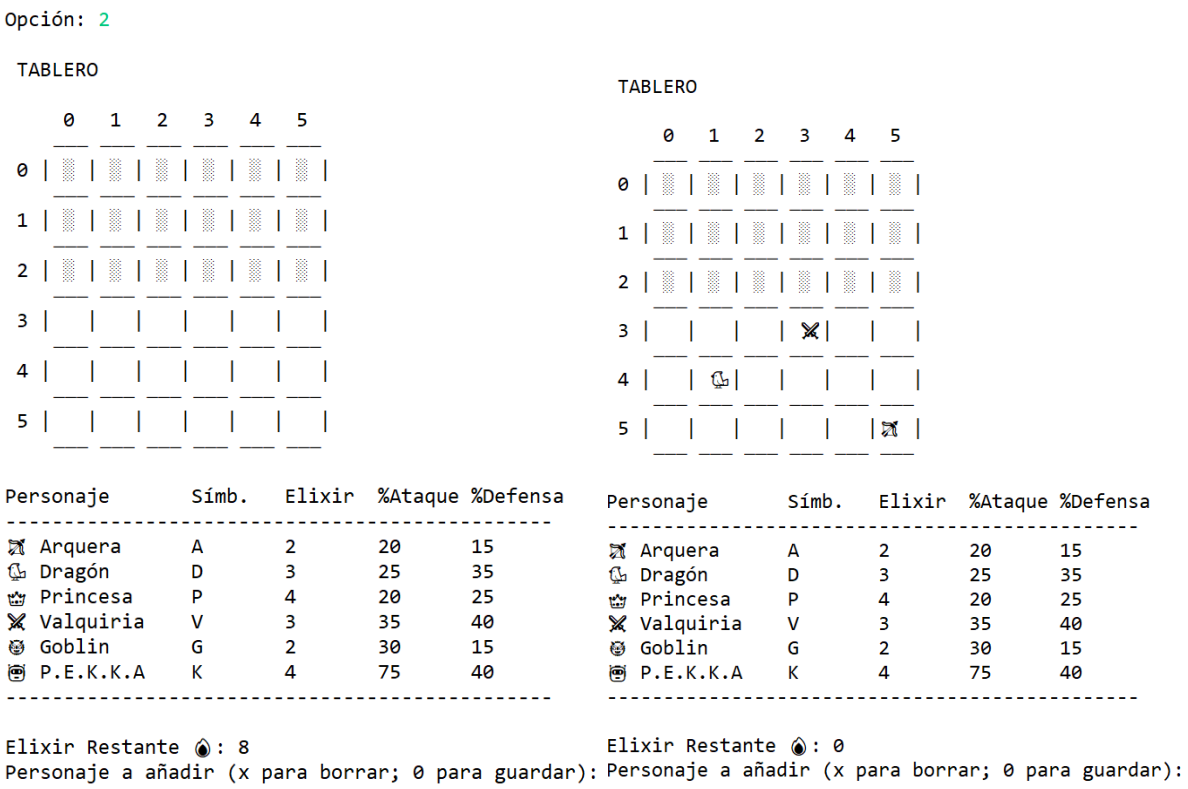


Imagen 3. Elementos a mostrar tras seleccionar la opción *"Configurar Baraja"*. Izq: interfaz antes de añadir ningún personaje. Der: interfaz después de añadir tres personajes.

Las características de los personajes pueden verse en la Imagen 3, teniendo en cuenta además que:

- Arquera (A): Dispara a la figura más alejada del tablero
- Dragón (D): Dispara a todas las figuras delante de él, si no hay nadie, avanza una posición
- Princesa (P): 2x Dispara a la figura más alejada del tablero
- Valquiria (V): Ataca a las tres posiciones delante de ella, si no hay nadie, avanza una posición
- Goblin (G): Ataca a las tres posiciones delante de él, si no hay nadie, avanza una posición
- P.E.K.K.A (K): 2x Ataca a las tres posiciones delante de él, si nadie, avanza una posición

A continuación, se introducirá el símbolo correspondiente al personaje que se quiere añadir. En caso de insertar un símbolo no correspondiente a ninguna figura, se le notificará al usuario y se volverá a mostrar toda la información de la opción #2. En caso de intentar añadir una figura con un coste superior al elixir restante, se le mostrará el mensaje "¡No tienes suficiente elixir!", y se volverá a mostrar toda la información de la opción #2. El carácter 'x' servirá para borrar una figura ya colocada en el tablero y el carácter '0', para guardar la configuración actual y regresar al menú inicial.

En caso de introducir un símbolo de personaje y disponer de suficiente elixir, se pedirá al usuario que inserte una posición numérica XY (siendo X el número de fila de acuerdo con la información mostrada en el tablero, e Y, el número de columna) donde se quiere insertar dicha figura (ver Imagen 4). En caso de que ya exista una figura en esa posición, se deberá notificar al usuario y no se añadirá ninguna figura. En caso de que la posición sea incorrecta, se le notificará al usuario y no se añadirá ninguna figura. Si la posición es correcta y está vacía, entonces se añadirá la figura al tablero. Para cualesquiera de las opciones anteriores, se volverá a mostrar toda la información de la opción #2.

```
Elixir Restante 💧: 5
Personaje a añadir (x para borrar; 0 para guardar): V
Introduce posición (p.ej. 11): 33
```

Imagen 4. Procedimiento para añadir nuevas figuras al tablero.

Si el usuario elige la **opción #3 (Guardar Baraja)**, el programa guardará en un fichero llamado "BarajaGuardada.txt" las figuras y su posición, como están actualmente en el juego. El fichero "BarajaGuardada.txt" deberá incluirse en una carpeta "Barajas" previamente creada en la carpeta del proyecto java. El formato de guardado de la baraja consistirá en una sucesión de caracteres separados por espacios, de la forma AXY, siendo A el símbolo del personaje, X su posición fila, e Y su posición columna. Por ejemplo, para la baraja de la Imagen 3izq, se podría guardar la siguiente secuencia: V33 D41 A55. Si al guardar, ya existía una baraja previamente guardada, ésta se sobrescribirá. Si el proceso de guardado ha sido satisfactorio, se mostrará un mensaje confirmando que se ha guardado correctamente y se volverá a mostrar el menú inicial. Si, por el contrario, ha habido algún problema al guardar, se notificará al usuario y se volverá al menú inicial.

Si el usuario elige la **opción #4 (Cargar Baraja)**, el programa cargará la baraja guardada en el fichero "BarajaGuardada.txt" situado en la carpeta "Barajas" del proyecto java. Si el fichero no existiera, estuviera vacío o hubiera algún tipo de problema al cargar la baraja, el programa lo notificará al usuario y volverá a mostrar el menú inicial. Si la carga se ha completado satisfactoriamente, se notificará al usuario y se volverá al menú inicial. Tened en cuenta, que además de cargar los personajes y su posición, también hay que calcular el elixir por si acaso, después de cargar, se quieren hacer cambios.

Si el usuario elige la **opción #1 (Nueva Partida)**, el programa comprobará si la baraja actual tiene al menos un personaje, de lo contrario, el programa mostrará el mensaje "¡Tienes que configurar tu baraja antes!" y volverá a mostrar el menú inicial. Si la baraja actual tiene al menos un personaje, entonces el programa cargará una baraja enemiga de forma aleatoria del fichero

“BarajasEnemigas.txt” contenida en la carpeta “Barajas”, mostrará el contenido de la baraja por pantalla (p.ej., ver Imagen 5) y se llamará al método startGame() proporcionado en la práctica. Este método permite que como máximo, durante 10 turnos, los personajes realicen sus acciones correspondientes por turno y se conozca quién es el ganador o perdedor de la partida (parte del trabajo de un programador es saber integrar métodos desarrollados por otras personas).

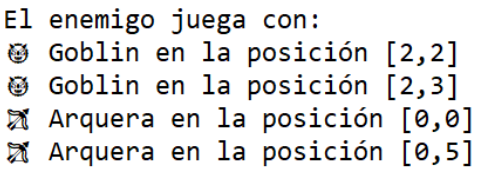


Imagen 5. Ejemplo de información a mostrar tras carga de baraja enemiga.

A continuación, el programa se irá avanzando con “enters” para ir viendo lo que sucede en cada turno como se muestra en la Imagen 6.

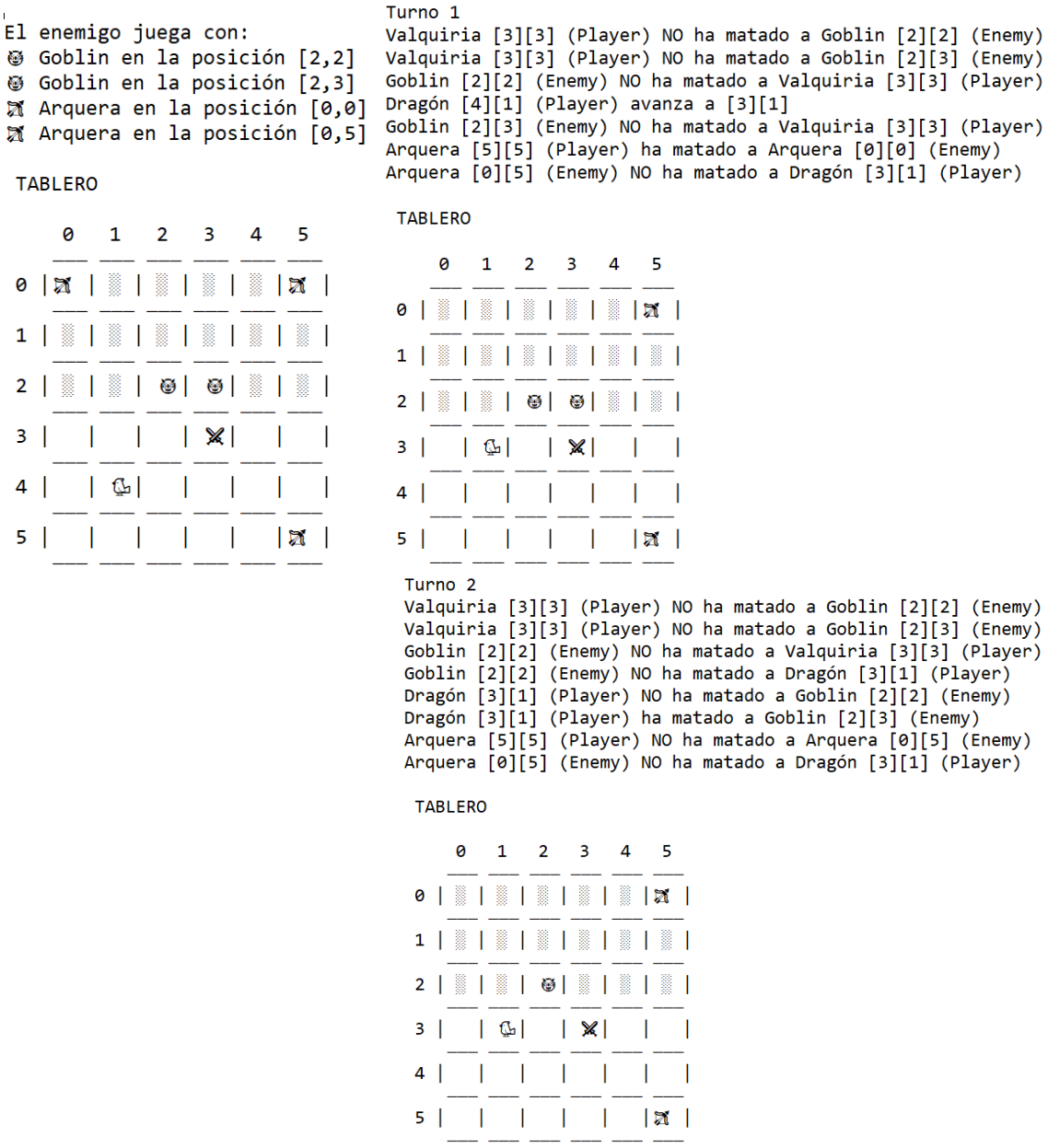
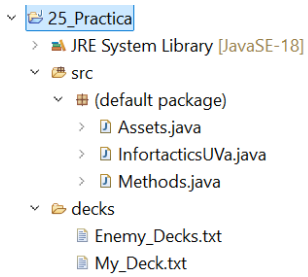


Imagen 6. Ejemplo de flujo de juego.

Guías para la programación y refinamiento:

- En esta práctica os proporcionamos:
  - El fichero “BarajasEnemigas.txt” que contiene una lista de barajas enemigas a razón de una por línea, que podrán ser elegidas de forma aleatoria cuando se inicia una partida. Este fichero debe colocarse dentro de la carpeta “Barajas” situada en la raíz del proyecto java. Este fichero no debe modificarse excepto si se realizan mejoras del juego.
  - El fichero “Assets.java” que contiene todas las constantes relativas al juego y a sus personajes. Para poder utilizar dichas constantes, deberéis incluir el fichero en el mismo paquete que vuestro fichero .java y llamarlas de la forma Clase.CONSTANTE (p.ej., Assets.INITIAL_ELIXIR). Este fichero no debe modificarse excepto si se realizan mejoras del juego.
  - El fichero “Methods.java” que contiene el procedimiento startGame() y otros métodos para que lo llaméis de acuerdo con las instrucciones de la opción #1. Para llamar al procedimiento y/o los métodos debéis hacer lo mismo que en el punto anterior. Este fichero no debe modificarse excepto si se realizan mejoras del juego.

- La estructura de ficheros del proyecto debería ser algo parecido a la siguiente imagen (los nombres de las carpetas y ficheros podrían ser diferentes).



- Para que los iconos se representen adecuadamente en la consola hay que establecer UTF-8 como método de codificación del proyecto. Para ello, en Eclipse, iremos a *Project* > *Properties* > *Resource* > *Text file encoding* > *Other: UTF-8*. Todos los iconos deben llamarse a través de las constantes proporcionadas en el fichero Assets.java

- Como en la mayoría de juegos, tenemos que distinguir qué es información del juego, y la forma en que muestra. En este juego, la información de juego consiste en las posiciones de los personajes (tanto los propios, como los enemigos). Es por eso, que tendremos que definir un vector de *Strings* para almacenar los personajes de nuestra baraja, y un vector de *Strings* para almacenar los del enemigo. En cada posición del vector tendremos un personaje y su posición, de acuerdo a lo explicado en la opción #3 (p.ej., “V44”, “G33”). Para aquellas posiciones vacías, utilizaremos una cadena vacía (“”).

- Para representar el tablero por pantalla deberemos definir un procedimiento con el nombre printBoard() que tendrá como parámetro de entrada una referencia a un vector con los personajes que se van a mostrar en el tablero y su posición (ver punto anterior). La salida de este procedimiento deberá ser similar a lo mostrado en la parte superior de la Imagen 3izq.

- Todas las entradas del usuario en el juego deben validarse, cuanto más robusto y fiable sea el programa, más puntuación

.

- Debéis documentar los métodos de acuerdo a los criterios de la asignatura. Para ello, podéis hacer uso de javadoc (aunque no es obligatorio).

Se recomienda también que, para hacer las pruebas de los diferentes métodos que se irán programando, no se empiece el programa desde el principio, sino que directamente se prueben los métodos (o la parte del código correspondiente) para unos valores dados (independientemente del resto del programa).

Como la mayoría del software, antes de hacer la entrega, se recomienda la realización de pruebas con potenciales usuarios del juego. De esta manera, los programadores podremos detectar si los usuarios entienden las instrucciones del programa y mejorar así su usabilidad. En próximas asignaturas veréis cómo hacerlo de una forma metodológica, pero debéis probar el programa desarrollado con amigos y familiares para intentar refinarlo antes del envío final, y documentarlo en la memoria: cuántas personas, cuál ha sido el criterio de selección, qué han tenido que hacer, qué resultados habéis obtenido y cómo habéis refinado el código.

Intentad mantener un registro y copia de las versiones que vais desarrollando. Más adelante veréis lo que son las plataformas de control de versiones, especialmente útiles para proyectos colaborativos en la que varias personas trabajan sobre el mismo software. Podéis investigar un poco más sobre ello, o mantener manualmente un buen registro de vuestras versiones.

En esta práctica se permite el uso de herramientas de inteligencia artificial generativa sin repercusión en la nota de la práctica. Sin embargo, será obligatorio indicar en la memoria del proyecto qué usos se le ha dado y añadir su correspondiente referencia. Nota: sólo se permiten utilizar en el código los métodos vistos en la asignatura.

Versiones de la Práctica:

Versión básica (máximo 7 puntos de 10): Las opciones #2 y #5 antes descritas (sin lectura ni escritura de ficheros).

Versión completa (máximo 12 puntos de 10): El juego tal y como se ha descrito. Además, cualquier otra mejora o ampliación del juego deberá ser comentada con el profesor de laboratorio, que podrá aumentar hasta en 2 puntos la nota de la práctica.

Por ejemplo:

- Añadir colores para distinguir figuras y/o equipos: hasta 0.25 pt
- Tener un registro de estadísticas: hasta 0.5 pt
- Poder guardar varias barajas, y poder elegir cuál utilizar: hasta 1 pt
- Jugar con evoluciones: hasta 2pt
- Poder elegir diferentes tableros: hasta 2pt
- Jugar humano vs human: hasta 2pt

Notas:

- Las mejoras sólo podrán aplicarse sobre la versión completa.
- La nota final de la práctica será calificada sobre 10 puntos.

Entrega de la Práctica:

La práctica debe realizarse por parejas, idealmente del mismo grupo de laboratorio o de grupos que comparten un mismo profesor. Cualquier otra situación deberá ser acordada con los profesores de prácticas.

Será calificada con arreglo a los criterios de calidad explicados en la asignatura (p.ej., robustez, eficiencia, mantenibilidad), el documento de "Malas prácticas en programación", la documentación de teoría y prácticas, y el acto de defensa.

Solo un alumno de cada pareja debe entregar la práctica. En todos los documentos entregados deben aparecer los nombres de los alumnos que la han realizado, junto con sus grupos de teoría y laboratorio. Ambos alumnos se hacen responsables de la entrega.

Se debe entregar, en el campus virtual:

- El **código** del programa en un único archivo .java de nombre InfortacticsUVa.
    - Debe ser compilable desde la línea de órdenes con la orden "javac *InfortacticsUVa*.java" y poder ser ejecutado con la orden "java *InfortacticsUVa*".
- La **documentación** del programa en un archivo pdf, de no más de 6 páginas con un tamaño de letra de 12pt y un interlineado de 1.15, en la que debe describirse:
    - Una pequeña introducción sobre el programa que se ha desarrollado, incluyendo las características del juego, la versión realizada, la estrategia general de resolución, las dificultades encontradas, etc.
    - La cabecera de los métodos desarrollados y su funcionalidad, incluyendo la especificación de cada uno de ellos (objetivo, precondiciones, entradas, salidas, etc.).
    - Las evaluaciones realizadas.

Esta documentación no debe contener el código del programa a no ser que se trate de ejemplos para mejorar la comprensión de su información.

La entrega debe realizarse antes del **14 de diciembre de 2025 a las 23:59 en la tarea habilitada en el Campus Virtual para tal fin**.

Defensa de la Práctica:

La defensa de la práctica será obligatoria para todos los grupos. El profesor de prácticas convocará a los alumnos un acto de "defensa de la práctica" que podrá ser durante el último laboratorio de prácticas de la asignatura (última semana lectiva de diciembre), en horario de tutorías, o en un horario acordado con el profesor de laboratorio. En ella, los alumnos expondrán de forma breve los métodos y las estructuras utilizadas, el profesor hará preguntas a los alumnos, relacionadas con la práctica entregada y su elaboración, y pedirá la realización de modificaciones en directo, para que demostréis que sois conocedores de cómo funciona vuestro programa y de su código. **No saber describir el software desarrollado por vosotros mismos, o no saber hacer los cambios solicitados supondrá la suspensión de la práctica.**