PROGRAMACIÓN
Trabajo 3: Saber y Ganar

Programación
CFGS DAW

Autores:

Lionel Tarazón lionel.tarazon@ceedcv.es

Encarni Serrano encarni.serrano@ceedcv.es

2019/2020

Saber y Ganar

El Videojuego

Fuente de la imagen: https://www.rtve.es

ÍNDICE

Lee atentamente todo el documento, varias veces si es necesario, **subraya** o **toma notas** de los aspectos más importantes, y tenlo a mano para consultarlo durante las fases de diseño y desarrollo.

1. OBJETIVO

Desarrollar un programa en Java que permita jugar a un juego de preguntas y respuestas inspirado en el conocido concurso de televisión Saber y ganar.

2. DESCRIPCIÓN GENERAL

El programa deberá permitir jugar a una versión simplificada de un concurso de preguntas y respuestas en el que varios jugadores responden preguntas por turnos, acumulan puntos y al final gana el que más puntos tiene. El programa también mantendrá un registro de los jugadores, el histórico de partidas jugadas y un ranking de los mejores jugadores. Toda la interacción con los usuarios será mediante entrada y salida estándar (teclado y pantalla). Al iniciar el programa se mostrará el siguiente menú principal:

1. **Jugar Partida** -> Permite jugar una nueva partida.
2. **Ranking** -> Muestra el ranking de los mejores jugadores.
3. **Histórico** -> Muestra el histórico de partidas.
4. **Jugadores** -> Permite acceder a un submenú de gestión de jugadores.
5. **Salir** -> Termina el programa.

3. JUGAR PARTIDA

Una partida se compone de jugadores y preguntas. Los jugadores podrán ser entre 1 y 6 (se permite un solo jugador para practicar o entrenar). El juego podrá tener un número variable de rondas: 3 (partida rápida), 5 (partida corta), 10 (partida normal) o 20 (partida larga). En cada ronda se le hace una pregunta a cada jugador y se obtienen puntos por contestar correctamente. Antes de jugar se pedirá por teclado la información necesaria para iniciar la partida.

3.1. Dinámica de la partida

Al principio de la partida los N jugadores se ordenan de forma aleatoria (1º, 2º, 3º, etc.) y su marcador se pone a cero puntos. Luego se juegan las R rondas de la partida. En cada ronda se le hace una pregunta distinta al azar a cada jugador y contestan. Es decir, se le pregunta al jugador 1, contesta, se le pregunta al jugador 2, contesta, etc. No hay rebotes de preguntas ni acertar permite contestar otra pregunta ni nada similar. Los jugadores contestan por turnos, sin más.

Cada pregunta acertada da un punto y los fallos dan cero puntos (no restan). Cuando se falla una pregunta debe mostrarse cual era la respuesta correcta. Cuando los N jugadores han contestado su pregunta entonces termina la ronda, se muestran los puntos de todos los jugadores y se pasa a la siguiente ronda.

Cuando se han jugado todas las rondas la partida termina, se muestran las puntuaciones finales, quién ha sido el ganador, se registra la partida en el histórico y se vuelve al menú principal.

3.2. Preguntas

Las preguntas de una partida pueden ser de tres tipos: Mates, Letras e Inglés.

- **Pregunta de Mates:** El jugador debe calcular el resultado de una expresión matemática generada al azar utilizando de 4 a 8 enteros, cada uno con valor entre 2 y 12, combinando sumas, restas y multiplicaciones (no hay divisiones). Por ejemplo: "10 + 7 * 2 + 4 * 5", la respuesta es 44.

  Te puede ser de utilidad la clase SriptEngineManager ya que permite evaluar expresiones matemáticas en formato String y obtener su resultado.

```
ScriptEngineManager manager = new ScriptEngineManager();
ScriptEngine engine = manager.getEngineByName("javascript");
Object result = engine.eval("10 + 7 * 2 + 4 * 5");
int valor = Integer.decode(result.toString());
System.out.println(valor); // muestra 44
```

- **Pregunta de Letras:** Dada una palabra elegida al azar del archivo *diccionario.txt*, se mostrará parcialmente dicha palabra (ocultando algunas letras) y el jugador deberá intentar adivinarla escribiendo la palabra completa correctamente. La palabra tendrá **L letras**, siendo obligatorio **L > 3** (más de 3 letras), y se ocultarán **L / 3** letras elegidas al azar (una letra por cada 3 letras que tenga la palabra). Por ejemplo:
  - Con "casa" se ocultaría 1 letras (4/3). Se podría mostrar "ca*a".
  - Con "sobredosis" se ocultarían 3 letras (10/3). Se podría mostrar "*ob*edo*is".

  Hay que tener en cuenta que la solución es única. No se aceptarán otras palabras válidas que pertenezcan al diccionario. Por ejemplo para "ca*a" no se aceptarán otras palabras del diccionario como "caza" o "cata". Solo se aceptará la palabra elegida "casa". Esto añade mayor dificultad debido al factor suerte.

- **Pregunta de Inglés:** Se mostrará una pregunta aleatoria del archivo *ingles.txt*, se mostrarán las cuatro posibles respuestas en orden aleatorio (A, B, C o D) y el jugador elegirá cuál cree que es la correcta. La estructura del archivo *ingles.txt* es muy sencilla. Cada pregunta ocupa 5 líneas: La primera contiene la pregunta, la segunda la respuesta correcta y las otras 3 líneas contienen 3 opciones incorrectas. Por ejemplo:

  Which programming language shares its name with an island in Indonesia?
  Java
  Python
  C
  Jakarta

3.3. Jugadores

Los jugadores podrán ser 'Humanos' (jugadores registrados en el sistema) o 'CPU' (jugará el ordenador, no se considera un usuario del sistema). En cada partida podrá jugar cualquier combinación de jugadores humanos y/o CPU. Todo jugador tiene un nombre identificativo único. En el caso de jugadores humanos será el nombre que tenga registrado en el sistema (no se permiten espacios). En el caso de jugadores CPU su nombre será "AI1", "AI2", etc.

Los jugadores 'Humanos' jugarán, como es lógico, introduciendo la respuesta por teclado.

Lo jugadores 'CPU' responderán automáticamente de forma distinta según el tipo de pregunta:

- Preguntas de Mates: Una CPU es experta en cálculos matemáticas. Siempre acierta.
- Preguntas de Letras: Las letras no son su punto fuerte. Siempre falla.
- Preguntas de Inglés: La CPU no sabe inglés pero confía en la suerte. Elegirá al azar una de las 4 posibles respuestas (A, B, C o D).

4. OTRAS CARACTERÍSTICAS

4.1. Ranking

La opción 'Ranking' del menú principal mostrará una lista de todos los jugadores humanos del sistema y el número total de preguntas correctas que ha contestado cada uno (en orden decreciente, es decir, primero los que más preguntas han contestado). Por ejemplo:

Ana 27
Pepito 22
María 12
Jaimito 0

El ranking de jugadores puede calcularse automáticamente a partir del histórico de partidas. Otra opción es mantenerlo en un archivo *ranking*.txt que deberá actualizarse tras cada partida.

4.2. Histórico

La opción 'Histórico' del menú principal mostrará una lista con todas las partidas que se han jugado (en orden cronológico, primero las partidas más antiguas). Se mostrará una partida por línea con los nombres de los jugadores y sus puntuaciones, por ejemplo así:

Pepito 4 María 3 Ana 5
María 9 Pepito 6 AI1 3
Ana 10 Pepito 12 AI1 5 AI2 7
AI1 5 AI2 7 AI3 3 AI4 8
Ana 12

El histórico puede mantenerse en un archivo *historico.txt* en el que habrá que registrar los resultados de cada partida jugada.

4.3. Jugadores

La opción 'Jugadores' del menú principal permitirá acceder a un submenú con unas sencillas opciones de gestión de jugadores registrados:

1. **Ver jugadores** -> Muestra la lista de jugadores registrados
2. **Añadir jugador** -> Permite añadir al sistema un nuevo jugador
3. **Eliminar jugador** -> Permite eliminar del sistema un jugador registrado
4. **Volver** -> Vuelve al menú principal

Algunas consideraciones:

- Los jugadores registrados son solo humanos. No hay que registrar jugadores CPU.
- Los nombres de jugadores no pueden contener espacios.
- No pueden existir dos jugadores con el mismo nombre.
- Cuando se elimina un jugador éste debe eliminarse del ranking pero el histórico de partidas se mantiene intacto.

5. ELABORACIÓN DEL TRABAJO

5.1. Fase 1 - Diseño (en equipo)

Esta fase podrá realizarse en **equipos de 2 o 3 personas**. Podéis utilizar el foro del aula virtual o cualquier otro medio de comunicación telemático para poneros en contacto con otros compañeros, organizar vosotros los equipos y realizar esta fase. Deberéis comunicarnos a través del foro cual es el nombre del equipo (por ejemplo, *Los Meme Developers*) y los miembros que lo componen (nombres y apellidos). En casos excepcionales se permitirá hacerlo individualmente, previa autorización de los profesores.

Tras leer y analizar el enunciado del trabajo y los requisitos en profundidad, deberéis realizar el **diseño del diagrama de clases UML** (clases con sus relaciones, atributos y métodos). **Esta fase del trabajo es fundamental ya que un correcto diseño facilitará mucho el desarrollo**. Dicho de otro modo, un mal diseño dificultará mucho la programación del software. Podéis realizar el diagrama de clases UML con la herramienta de vuestra elección, por ejemplo con draw.io.

Deberéis **enviar por correo electrónico privado a ambos profesores el diagrama de clases** para que lo revisen y os den indicaciones o sugerencias de mejora. Se podrán plantear dudas o preguntas puntuales sobre el diseño, pero no se garantiza que se proporcionen todas las respuestas. En el momento que consideréis oportuno podréis pasar a la fase 2.

5.2. Fase 2 - Desarrollo (Individual)

El desarrollo (programación) deberéis hacerlo de forma individual y deberá estar basada en el diseño inicial. Podréis hacer pequeños cambios sobre el diseño inicial si no hay más remedio.

El programa se desarrollará en **JAVA** mediante **POO**. Deberá **funcionar correctamente** siguiendo las especificaciones dadas, manejar las posibles **excepciones** e incluir **breves comentarios explicativos** (al menos uno por clase y método).

6. CRITERIOS DE CALIFICACIÓN

El trabajo se calificará según el diagrama UML y las funcionalidades que se hayan desarrollado, su correcto **funcionamiento**, un correcto diseño y aplicación de la **Programación Orientada a Objetos**, el manejo de **excepciones**, la manipulación de **ficheros**, el uso de **estructuras de datos** (arrays y/o ArrayLists), la correcta **estructuración y legibilidad del código** así como el uso de **comentarios explicativos** (es suficiente con un comentario por clase y método).

- **Diagrama de clases UML (1 punto)**.
- **Jugar partida (6 puntos)**: Se pueden jugar partidas correctamente entre varios jugadores y con distintos tipos de preguntas tal y como se especifica en los requisitos.
- **Jugadores/Usuarios (1 punto)**: El programa incorpora jugadores (usuarios registrados) y permite listarlos, añadirlos y eliminarlos.
- **Histórico (1 punto)**: El programa registra el histórico de partidas y permite visualizarlo.
- **Ranking (1 punto)**: El programa permite obtener y visualizar el ranking de usuarios.

Como puede observarse, **la funcionalidad fundamental es la de jugar partidas** que junto con el diagrama de clases UML permite aprobar el trabajo con una nota máxima de **7 puntos** (sin usuarios registrados ni histórico ni ranking).

Añadiendo las funcionalidades de **usuarios registrados, histórico y ranking es posible obtener hasta 3 puntos más** (uno por funcionalidad), que sumado a los 7 puntos anteriores se puede obtener una nota máxima de 10.

7. ENTREGA

Deberás entregar por el aula virtual:

- **Diagrama de clases UML** actualizado con los cambios realizados al programar.
- **Proyecto NetBeans** con la carpeta del proyecto comprimida en un **zip**.

La fecha límite de entrega es el **lunes 20 de abril a las 23:55h**.

Licencia