TRIVIAL

Per: Jordi Ribellas i Xavier Martínez

Assignatura: Desenvolupament d'interfícies – M7

Índex

1. Objectius

El objetivo de esta práctica es crear un juego de preguntas para dos personas. La aplicación contendrá cuatro pantallas diferentes: La de inicio (introducción de datos), tablero principal, preguntas y final del juego.

La aplicación obtendrá las preguntas de un fichero XML, donde se seleccionará una pregunta al azar. Las preguntas serán tipo test donde solo hay una respuesta correcta. Este fichero XML se creará con las aportaciones de los compañeros de clase.

En el juego se alternarán los jugadores para responder las preguntas. En caso de respuesta acertada, el jugador ganará un punto y avanzará su ficha. En caso de respuesta incorrecta, no se puntuará ni se avanzará. Tras cada pregunta se pasará el turno al oponente.

Si el primer jugador llega el primero a la meta y al segundo jugador solo le falta una pregunta para ganar, se permitirá que el segundo jugador finalice su turno. Si acierta la pregunta, la partida acaba en empate. En caso de fallar, gana el primer jugador.

2. Introducció

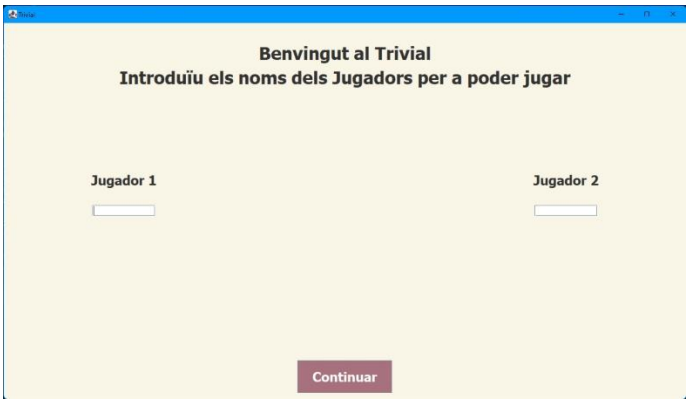
Benvingut/da a la documentació del projecte de desenvolupament del nostre Trivial programat en Java! En aquesta documentació, trobaràs informació detallada sobre com hem creat aquesta aplicació del joc clàssic de preguntes i respostes utilitzant el llenguatge de programació Java.

En aquest document, explicarem les diferents funcions i característiques del nostre Trivial, com ara el disseny de la interfície d'usuari, la lògica del joc, la gestió de preguntes i respostes, així com la seva implementació en Java.

En resum, aquesta documentació ofereix una visió detallada dels processos i tècniques que hem utilitzat per a desenvolupar una aplicació de joc lleuger, intuïtiva i entretinguda en Java.

3. Funcionament

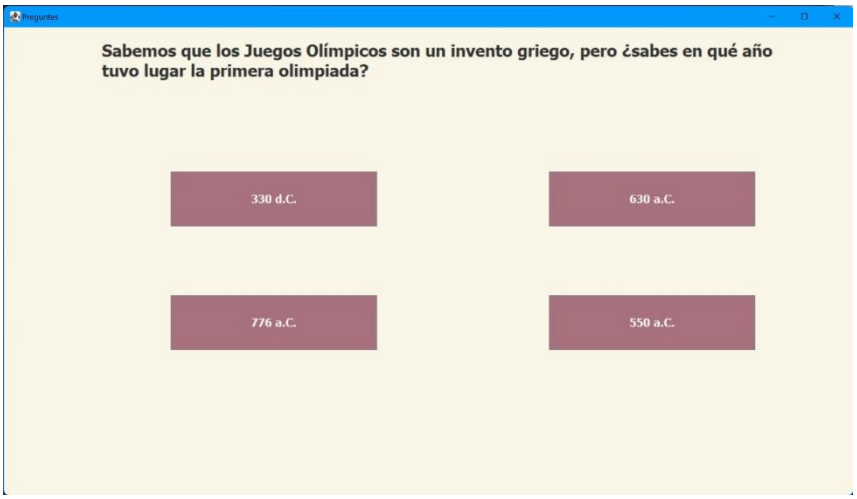
En aquest apartat descriurem com funciona l'aplicació.



Aquesta és la primera pantalla que es veu en executar el joc. Permet als jugadors introduir els seus noms. Aquests noms es mostraran en el tauler. Si es prem al botó de continuar amb els camps de text buits, s'assignen els noms "J1" i "J2" automàticament.

Aquesta es la pantalla principal del joc: el tauler. El nostre tauler disposa de 2 files de 8 caselles cada fila (contant l'inici i la meta dins les 8). Els jugadors son representats amb la icona de la peça d'escacs i la meta es la bandera. La figura del jugador actual apareixerà i desapareixerà intermitentment per remarcar a qui li toca ara mateix.

En la pantalla també es mostra el nom del jugador actual, el numero de la ronda actual (una ronda passa quan els dos jugadors es mouen) i el botó per mostrar la pregunta i que el jugador actual la respongui

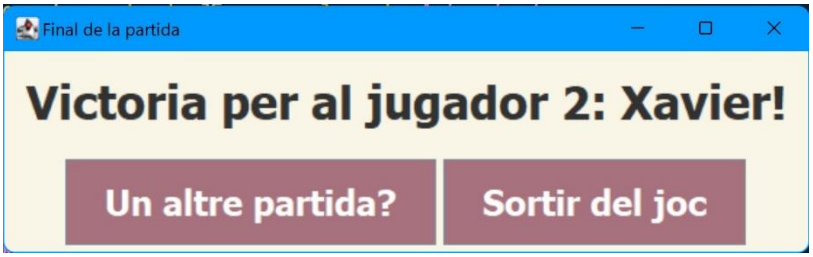


Aquesta es la pantalla de preguntes. Aquí es mostra l'enunciat i 4 botons amb una resposta cada botó. Per respondre a la pregunta, sols cal fer click en el botó de la resposta per veure si hem acertat o no. Si una resposta es molt llarga, el text queda retallat i es mostra amb punts suspensius al final. El text de l'enunciat fa noves línies perquè es mostri sempre sencer.

Pantalla d'una pregunta que hem contestat correctament. Aquí hem fet click a "Milú" i com que ha estat la resposta correcta, ha pintat el botó de color verd. Ara si fem click un altre vegada a qualsevol dels botons, tancarem la pregunta i tornarem al tauler amb el jugador actual en la nova posició (si es que ha encertat)

En aquesta captura, hem fet click a una resposta incorrecta. Hem fet click a "Rafa Nadal", quan la resposta correcta era "Jimmy Connors". Quan ens equivoquem, pinta la resposta que hem escollit nosaltres en vermell i la correcta en verd; per tal de dir al jugador quina es la correcta. Al igual que abans, si ara fem click a qualsevol botó un altre vegada, tancarem la pregunta i tornarem al tauler

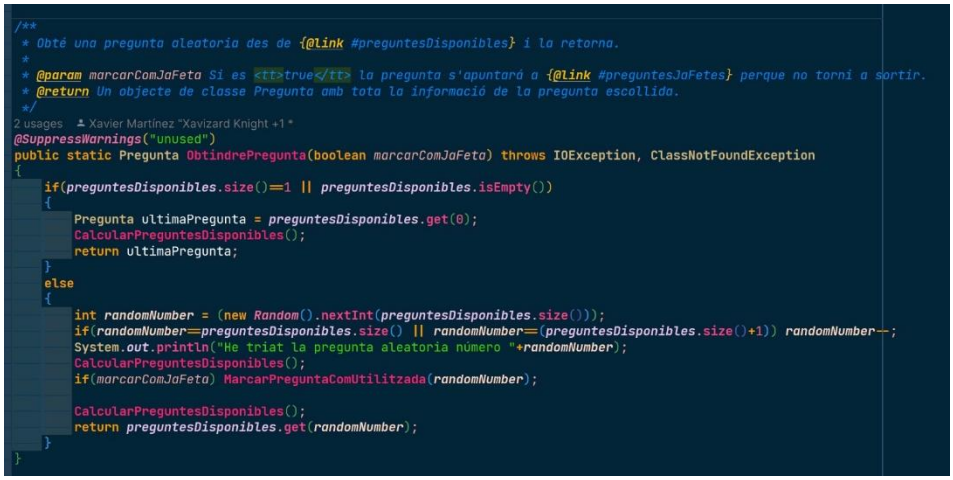
Captura del tauler en una partida mes avançada. Aquí el jugador de dalt (Xavier) esta perdent en contra del jugador de baix (en Jordi).



Aquesta es la petita pantalla final del joc. Quan algun jugador guanya, el tauler es tanca i mostra aquesta pantalla. Aquí diu qui numero te el jugador que ha guanyat, el seu nom i te dos botons: fer un altre partida i tancar el joc. En fer un altre partida, s'obre la pantalla de posar els noms; però amb els noms dels jugadors actuals ja omplerts, per així començar una nova partida amb la mateixa gent en sols 2 clicks o poder canviar els noms abans de començar un altre partida.

4. Parts importants del codi

En aquest apartat posarem les parts del codi que creiem que son les més complexes, posarem una imatge i una breu descripció.



```java
/**
 * Obté una pregunta aleatoria des de {@link #preguntesDisponibles} i la retorna.
 *
 * @param marcarComJaFeta Si es <tt>true</tt> la pregunta s'apuntarà a {@link #preguntesJaFetes} perque no torni a sortir.
 * @return Un objecte de classe Pregunta amb tota la informació de la pregunta escollida.
 */
2 usages  Xavier Martínez "Xavizard Knight +1 *
@SuppressWarnings("unused")
public static Pregunta ObtindrePregunta(boolean marcarComJaFeta) throws IOException, ClassNotFoundException
{
    if(preguntesDisponibles.size()==1 || preguntesDisponibles.isEmpty())
    {
        Pregunta ultimaPregunta = preguntesDisponibles.get(0);
        CalcularPreguntesDisponibles();
        return ultimaPregunta;
    }
    else
    {
        int randomNumber = (new Random().nextInt(preguntesDisponibles.size()));
        if(randomNumber==preguntesDisponibles.size() || randomNumber==(preguntesDisponibles.size()+1)) randomNumber--;
        System.out.println("He triat la pregunta aleatoria número "+randomNumber);
        CalcularPreguntesDisponibles();
        if(marcarComJaFeta) MarcarPreguntaComUtilitzada(randomNumber);

        CalcularPreguntesDisponibles();
        return preguntesDisponibles.get(randomNumber);
    }
}
```

Aquesta es la funció per que es rebi com a return value un objecte de Pregunta. Si sols queda una pregunta aleshores retornem la única pregunta que queda; sinó seleccionem un numero random i retornem una pregunta aleatòria.

```java
@Override
public void actionPerformed(ActionEvent e)
{
    System.out.println("Preparant Tauler.java...");
    ZonedDateTime now = ZonedDateTime.now();

    this.dispose();

    if(j1JTF.getText().isEmpty()) j1JTF.setText("J1");
    if(j2JTF.getText().isEmpty()) j2JTF.setText("J2");

    try
    {
        System.out.println("Preparant Tauler.java... OK en "+now.until(ZonedDateTime.now(),ChronoUnit.MILLIS)+"ms");
        if(MetaController.isThisDebugMode)
        {
            mainTesting.tauler = new Tauler(j1JTF.getText(), j2JTF.getText(), debugMode: true);
            mainTesting.tauler.setVisible(true);
        }
        else
        {
            mainTrivial.tauler = new Tauler(j1JTF.getText(), j2JTF.getText(), debugMode: false);
            mainTrivial.tauler.setVisible(true);
        }
    }
    catch(IOException ex)
    {
        throw new RuntimeException(ex);
    }
}
```

Aquesta es la captura del botó de continuar en la pantalla de posar els noms. Aquí es confirma si els camps estan buits o no (per si cal posar o no els noms per defecte) i carreguem la pantalla del tauler. Tenim dos mains (un normal i un altre de proves); i guardem la pantalla del tauler com una variable en el mainTesting o mainTauler; per això accedim als mains per tal d'accedir al tauler

```
19 usages  Xavier Martínez "Xavizard Knight +1
public class Utils
{
    /**
     * Retorna un <tt>BufferedImage</tt> ja redimensionat segons s'especifiqui.
     *
     * @param bufferedImage L'imatge a modificar
     * @param newWidth L'amplada que volem que tingui l'imatge.
     * @param newHeight L'alçada que volem que tingui l'imatge.
     * @return Un BufferedImage amb totes les especificacions especificades.
     */
    19 usages  Xavier Martínez "Xavizard Knight
    public static BufferedImage resize(BufferedImage bufferedImage, int newWidth, int newHeight)
    {
        Image image = bufferedImage.getScaledInstance(newWidth, newHeight, Image.SCALE_SMOOTH);
        BufferedImage newBufferedImage = new BufferedImage(newWidth, newHeight, BufferedImage.TYPE_INT_ARGB);

        Graphics2D graphics2D = newBufferedImage.createGraphics();
        graphics2D.drawImage(image,  x: 0,  y: 0,  observer: null);
        graphics2D.dispose();

        return newBufferedImage;
    }
}
```

Aquesta es la única funció d'una classe de mètodes random per a coses varies; pensàvem que tindríem mes mètodes però al final aquesta classe s'ha quedat amb una sola funció. El que fa aquesta funció es redimensionar una imatge. En el tauler les imatges de les caselles buides, els jugadors i les banderes han de ser de exactament 175x175 píxels, però les imatges reals son algú mes grans; en cridar aquest mètode en assignar les imatges al Imagino, la redimensionem al tampany que toca perquè no hi hagi cap problema

```
/**
 * Pinta tot el tauler per primera vegada.
 *
 * <p>Es important que aquest metode s'executi abans de la primera crida a {@link #paintColoursTiles()}.
 * Perquè el que fa {@link #paintColoursTiles()} és actualitzar la imatge d'ImageIcon(s) ja existents;
 * aquest metode els crea per primera vegada.
 *
 * @throws IOException
 */
/javadoc/
public void firstPaintTiles() throws IOException
{
    for(int i=0; i<16; i++)
    {
        BufferedImage bufferedImage;
        bufferedImage = i % 2==0
                ? ImageIO.read(new File(pathCollection.get("taulerClar")))
                : ImageIO.read(new File(pathCollection.get("taulerFosc")));
        images[i] = new JLabel(new ImageIcon(Utils.resize(bufferedImage,  newWidth: 175,  newHeight: 175)));
    }
}
```

Ara anem a parlar de com assignem les imatges a les caselles del tauler. El tauler es una GridLayout de 2 files i 8 columnes. A cada casella hi ha una imatge que la posem via ImageIcon d'un JLabel. Aquest mètode sols s'executa una vegada i pinta totes les caselles buides via construint els ImageIcons.

```
 */
/javadoc/
public void paintColoursTiles() throws IOException
{
    for(int i=0; i<16; i++)
    {
        BufferedImage bufferedImage;
        switch(i)
        {
            case 7: case 15:
                bufferedImage = ImageIO.read(new File(pathCollection.get("taulerFosc-bandera50")));
                break;
            default:
                bufferedImage = i % 2==0
                        ? ImageIO.read(new File(pathCollection.get("taulerClar")))
                        : ImageIO.read(new File(pathCollection.get("taulerFosc")));
                break;
        }
        images[i].setIcon(new ImageIcon(Utils.resize(bufferedImage, newWidth: 175, newHeight: 175)));
    }
}
```

Aquesta funció es similar a l'anterior però aquí assignem noves imatges a ImageIcons ja existents (els hem construït abans, tal com parlava en la captura anterior). Aquí a demes de pintar el color de fons clar o fosc segons si la columna es parell o senar, també posem les banderes al final (caselles 7 i 15).

```
/**
 * Aquest metode dibuixa als jugadors a allà on els toqui. Basicament canvia la imatge ja existent per la del
 * jugador.
 *
 * <p>El <tt>bufferedImage1</tt> es refereix al primer jugador (el de la fila de dalt), i el <tt>bufferedImage2</tt>
 * al segon jugador (fila de baix).
 *
 * @throws IOException
 */
/javadoc/
public void paintPlayerPositions() throws IOException
{
    BufferedImage bufferedImage1;
    bufferedImage1 = score[0] % 2==0
            ? ImageIO.read(new File(pathCollection.get("taulerClar-jugador95")))
            : ImageIO.read(new File(pathCollection.get("taulerFosc-jugador95")));
    images[score[0]].setIcon(new ImageIcon(Utils.resize(bufferedImage1, newWidth: 175, newHeight: 175)));

    BufferedImage bufferedImage2;
    bufferedImage2 = score[1] % 2==0
            ? ImageIO.read(new File(pathCollection.get("taulerClar-jugador95")))
            : ImageIO.read(new File(pathCollection.get("taulerFosc-jugador95")));
    images[score[1]+8].setIcon(new ImageIcon(Utils.resize(bufferedImage2, newWidth: 175, newHeight: 175)));
}
```

Aquesta funció pinta sols els jugadors a la posició on els hi toqui, canvia la imatge que ja hi havia per la nova icona del jugador.

5. Conclusió

En conclusió podem treure d'aquest projecte que ha sigut un repte ja que ha sigut bastant complicat i tediós de programar i hem aconseguit crear una aplicació de joc fiable i eficient, amb una interfície d'usuari atractiva i intuïtiva que permet als usuaris gaudir d'una experiència de joc entretinguda i amb musica de fons en certes pantalles.

En general, estem molt orgullosos del nostre treball en aquest projecte i esperem que, si aquesta aplicació es llancés al marcat, els usuaris finals puguin gaudir d'una experiència de joc emocionant i divertida amb el nostre Trivial programat en Java.