# Conecta 4

PROP 2025-26 Q1

Mario Martin – Ignasi Gómez – Jordi Urmeneta – Bernat Orellana

# Tècniques d'intel·ligència Artificial

- Algoritmes per a jocs:
  - MIN-MAX
  - MIN-MAX amb poda alpha-beta

# Objectiu

- Guanyar una partida de conecta 4
- Adversaris:
  - Aficionat (Aleatori)
    - Guanyar
  - Professional (MIN-MAX)
    - Guanyar o empatar

# Programació bàsica

- Fitxer *Juga2.java*
  - *Constructor*

```
 */
public Juga2(Jugador p1, Jugador p2, boolean useAutoMode) {
    initComponents();

    jTextField1.setEnabled(false);
    jTextField2.setEnabled(false);
    jTextField3.setEnabled(false);
    player1 = p1;
    player2 = p2;
    this.autoMode = useAutoMode;

    init();

}

private void init() {
    t = new Tauler(8);

    currentPlayer = player1;
    otherPlayer = player2;
    currentColor = 1;
    otherColor = -1;

    jTextField1.setText(player1.nom());
    jTextField3.setText(player2.nom());


    Dimension mides = jLayeredPanel.getSize();
    Ymax = mides.getHeight();
    Xmax = mides.getWidth();
    Step = (int) Xmax / 8;
}
```

Creació del tauler

# Programació bàsica

- Fitxer *Juga2.java*
  - *main*

```
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
     * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
     */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex) {
        java.util.logging.Logger.getLogger(NewJFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (InstantiationException ex) {
        java.util.logging.Logger.getLogger(NewJFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {
        java.util.logging.Logger.getLogger(NewJFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {
        java.util.logging.Logger.getLogger(NewJFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    }
    //</editor-fold>


    // Definiu al vostre gust els jugadors a enfrontar.
    Jugador p1 = new Manual();
    //Jugador p1 = new Aleatori();

    Jugador p2 = new Profe(8,false);
    //Jugador p2 = new Manual();

    boolean autoMode = true;
    final Juga2 j = new Juga2(p1,p2, autoMode);

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        @Override
        public void run() {
            j.setVisible(true);
            j. mostraTornActual();
        }
    });
}
```

Creació dels jugadors, els paràmetres necessaris es passen pel constructor (*p.ex. el número de nivells que ha de baixar el minimax*)

# Programació bàsica

- Fitxer *Juga2.java*
  - *Click (tirada) i fil de backgrou*

```
private void jLayeredPane1MouseClicked(java.awt.event.MouseEvent evt) {//GEN
      // TODO add your handling code here:

    if(estaPensant) return;

    int X = evt.getX();
    int Y = evt.getY();


    int col = (int) X / Step;

    if ("Manual".equals(currentPlayer.nom())) {
        mouCurrentPlayer(col);
    } else {
        runAuto();
    }
    //mostraTornActual();

} //GEN-LAST:event_jLayeredPane1MouseClicked


private void mouCurrentPlayer(int colu) {

    t.afegeix(colu, currentColor);
    repaint();
    verificaSiHaAcabat(colu, currentColor);

}

private void runAuto() {
    estaPensant = true;
    // Executem l'automàtic en background
    String t =  (currentPlayer==player1?"P1":"P2")+ " ESTIC PENSANT.....";
    jTextField2.setText(t);
    jLayeredPane1.setBackground(new java.awt.Color(255, 255, 0));
    jLayeredPane1.setEnabled(false);
    (new Mover(currentColor, currentPlayer)).execute();
}
```

# Programació bàsica

- Fitxer *Juga2.java*
  - *Fil per "pensar" el moviment:*

```
class Mover extends SwingWorker<Integer, Object> {

    int color;
    Jugador jugador;

    Mover(int color, Jugador jugador) {
        this.color = color;
        this.jugador = jugador;
    }

    @Override
    public Integer doInBackground() {
        return jugador.moviment(t, color);
    }

    @Override
    protected void done() {
        try {

            jLayeredPanel.setBackground(new java.awt.Color(255, 255, 255));
            jLayeredPanel.setEnabled(true);

            mouCurrentPlayer(get());
            repaint();
            estaPensant = false;
        } catch (Exception ignore) {
        }
    }
}
```

Cridem a la funció ***moviment*** de la interfície ***Jugador***, que és la que ha de fer la feina de decidir quin moviment fer (número de columna on tirar)

# Programació bàsica

- Fitxer *Juga2.java*
  - *Final del joc*

```
private void verificaSiHaAcabat(int colu, int color) {

    String text1 = "", text2 = "", text3 = "", dTitle = "";

    if (t.solucio(colu, color) || !t.espotmoure()) {

        if (t.solucio(colu, color)) {
            if (currentPlayer == player1) {
                text1 = "WINNER";
                text3 = "LOSER";
                text2 = "VERMELL AMB MOVIMENT A COLUMNA " + (colu + 1);
                dTitle = "GUANYA P1(" + currentPlayer.nom() + ")";
            } else {
                text3 = "WINNER";
                text1 = "LOSER";
                text2 = "BLAU AMB MOVIMENT A COLUMNA " + (colu + 1);
                dTitle = "GUANYA P2(" + currentPlayer.nom() + ")";
            }
        } else {
            // no es pot moure
            text1 = "NO PUC MOURE";
            text3 = "NO PUC MOURE";
            text2 = "TAULES";
            dTitle = "TAULES";
        }

        jTextField1.setText(text1);
        jTextField2.setText(text2);
        jTextField3.setText(text3);

        int n = JOptionPane.showConfirmDialog(
                this, dTitle,
                "Tornar a jugar",
                JOptionPane.YES_NO_OPTION);
        if (n == JOptionPane.YES_OPTION) {
            init();
        } else if (n == JOptionPane.NO_OPTION) {
            System.exit(0);
        }
    } else {
        canviTorn();
    }
    mostraTornActual();
}
```

# Programació bàsica

- Interfície del jugador

```java
/**
 * Interfície que han d'implementar tots els Jugadors (automàtics i manuals)
 * de la nostra aplicació.
 * @author Profe
 */
public interface Jugador {
    /**
     * Decideix el moviment del jugador donat un tauler i un color de peça que ha de posar.
     * @param t  Tauler actual de joc
     * @param color  Color de la peça que possarà
     * @return Columna on fer el moviment
     */
    public int moviment(Tauler t, int color);

    /**
     * Retorna el nom del jugador que s'utlilitza per visualització a la UI
     * @return Nom del jugador
     */
    public String nom();
}
```

# Programació bàsica

- Interfície IAuto
  - *És com una marca que simplement indica que el jugador és automàtic i per tant no li cal la intervenció de l'usuari humà.*

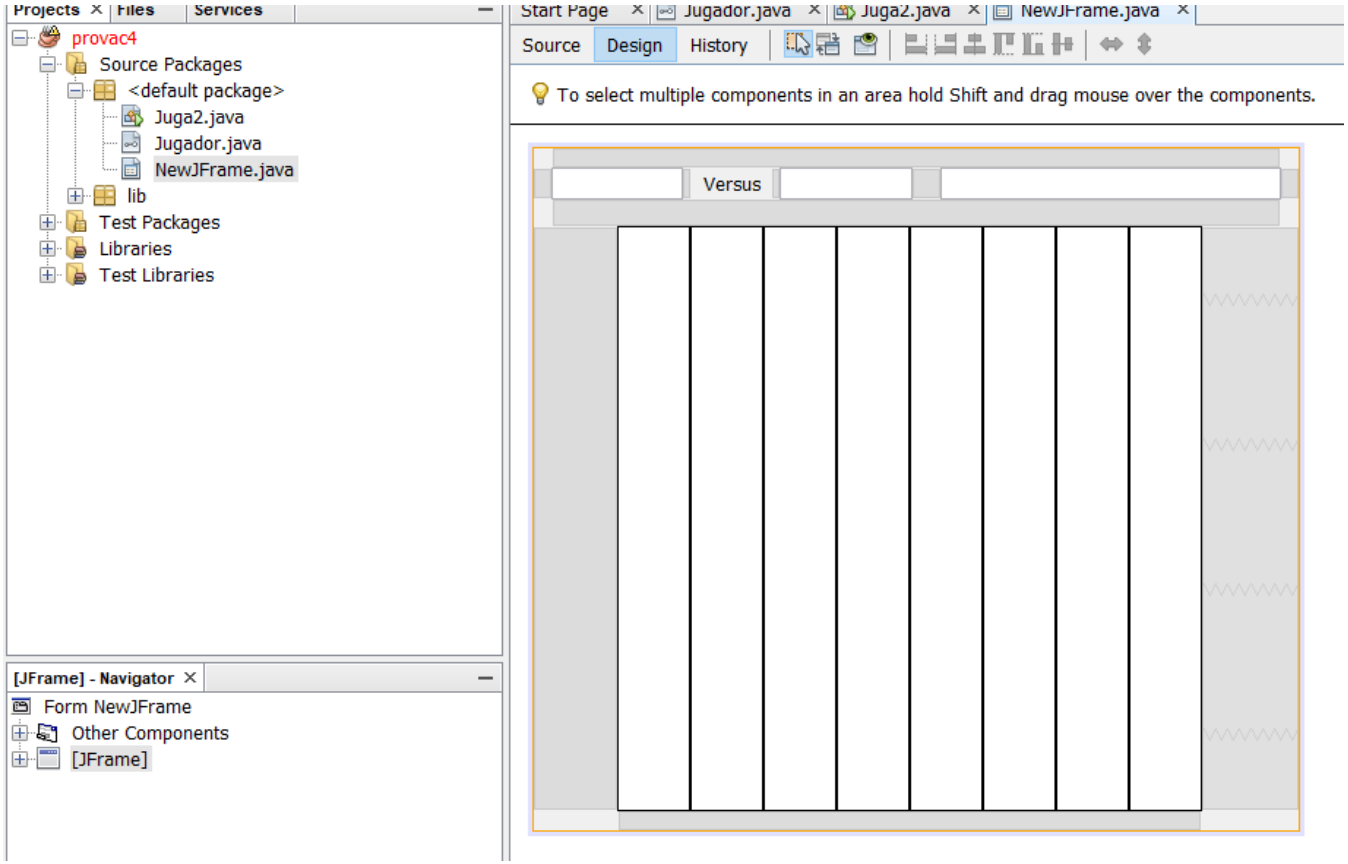
# Programació bàsica

- Ex. Jugador aleatori

```
/**
 * Jugador aleatori
 * "Alea jacta est"
 * @author Profe
 */
public class Aleatori
  implements Jugador, IAuto
{
  private String nom;

  public Aleatori()
  {
    nom = "RandomBanzai";
  }

  public int moviment(Tauler t, int color)
  {
    int col = (int)(8.0D * Math.random());
    while (!t.movpossible(col)) {
      col = (int)(8.0D * Math.random());
    }
    return col;
  }

  public String nom()
  {
    return nom;
  }
}
```

# Programació bàsica

- Interfície gràfica (Jframe)



# API Tauler

- Documentació
  - *Fitxer : javadoc_libc4/index.html*

# API Tauler

**Constructors**

| Constructor and Description |
| --- |
| `Tauler(int mida)`<br>Constructor del tauler |
| `Tauler(Tauler t)`<br>Constructor còpia del tauler |

***Method Summary***

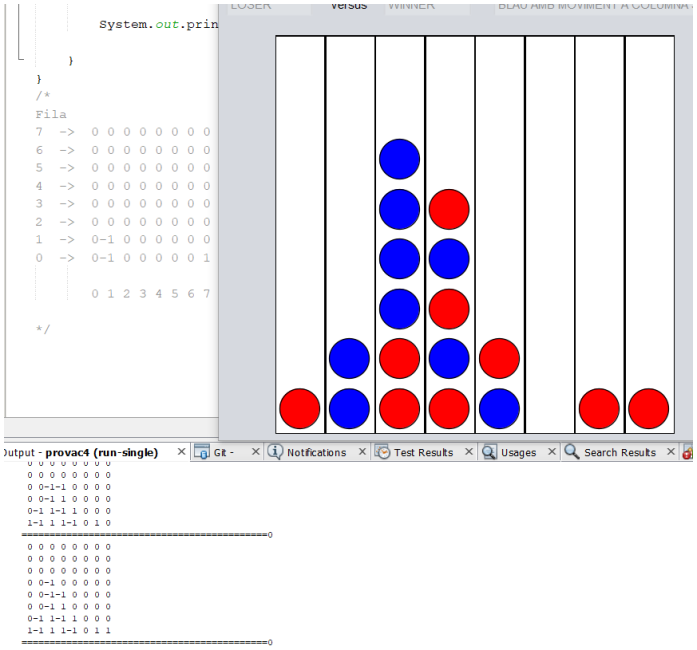
**All Methods** | **Instance Methods** | **Concrete Methods**

| Modifier and Type | Method and Description |
| --- | --- |
| `void` | `afegeix(int columna, int color)`<br>Afegeix una peca d'un color a una columna. |
| `boolean` | `espotmoure()`<br>Ens diu si el tauler encara hi ha moviments possibles (hi ha columnes lliures), o si per contra està totalment ple. |
| `int` | `getColor(int f, int c)`<br>Retorna el color d'una fitxa a una fila i columna del tauler |
| `int` | `getMida()`<br>Retorna la mida del costat del tauler ( és quadrat! |
| `boolean` | `movpossible(int col)`<br>Indica si podem posar una fitxa a una columna concreta ( si hi cap ) |
| `void` | `pintaTaulerALaConsola()`<br>Mostra el tauler a la consola ( for debbugging purposes ) |
| `boolean` | `solucio(int c, int clr)`<br>Mira si la última jugada (ja realitzada amb afegeix() ) a la columna c pel jugador de color clr és guanyadora. |

!!!

# API Tauler

- pintaTaulerAConsola():



# Instal·lació

- Obrir projecte
  - Obrir directori **c4_the_game** com a projecte Netbeans
  - Executar **Juga2** per provar
    - Es poden canviar paràmetres dels participants a ***main***.
    - Podeu activar o desactivar que els jugadors ***IAuto*** juguin automàticament (***autoMode***=true) o esperin a un clic (***autoMode***=false)

```
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
     Look and feel setting code (optional)

    // Definiu al vostre gust els jugadors a enfrontar.
    //Jugador p1 = new Manual();
    Jugador p1 = new Aleatori();

    Jugador p2 = new Profe(4,false);//
    //Jugador p2 = new Manual();


    boolean autoMode = false;
    final Juga2 j = new Juga2(p1, p2, autoMode);
```

# Activitat

- Entrega a Atenea segons data de lliurament programada.
- Grups de 2 persones. **Recordeu que els grups han de ser diferents a l'activitat anterior!!!**
- Zip amb
  - Codi (projecte NetBeans sense *build* ni *dist*)
  - Documentació en format PDF
  - Fitxer de text amb noms i NIFs dels alumnes
- El nom del zip: **NIF1_NIF2.zip**
- Només lliura una persona de la parella.

# Activitat

- 10% Nota total de curs
- Nota
  - 75% Codi i funcionalitat
    - 20% El jugador funciona
      - No triga més de 20s en decidir cada moviment ( a profunditat 8)
      - S'ha parametritzat el codi amb profunditat màxima (via constructor)
      - Es mostra per consola el nombre de jugades finals explorades (nodes on s'ha calculat l'heurística.
    - 20% Guanya Aleatori
    - 20% Guanya o empata amb "Profe"
      - Profunditat 8. Assegureu que la profunditat és la que dieu que és !

# Activitat

- 25% Documentació
  - 5% : Documentació del codi (Javadoc) de la vostra classe escrita i generada en una carpeta ./Javadoc dins del mateix projecte.
  - 10%: Explicació de l'heurística dissenyada pel jugador i detalls d'implementació que considereu rellevants.
  - 5%: Estudi de la incidència de la poda alfa-beta en el número de nodes explorats. Avaluació de la possible incidència de l'ordre d'exploració dels fills en la poda.
  - 5%: Avaluació de la incidència de l'ordre d'exploració dels fills en la poda alfa-beta.

# Activitat

- Condicions
  - El jugador ha de ser un treball original vostre
  - El codi del jugador ha de ser correcte
    - Netbeans no reporta errors
    - El projecte compila
    - El jugador funciona i no dona excepcions durant l'execució de tota la partida.