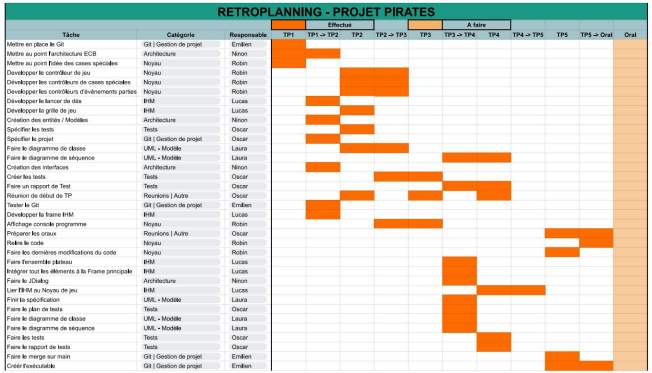
Projet Pirate

AUTEFAGE Ninon - Responsable Architecture
GUILLOT Laura - Responsable UML
SANCHEZ Emilien - Responsable Spécifications
PUJOL Robin - Responsable Développement noyau
QUANG Lucas - Responsable IHM
GLEDEL Oscar - Chef de projet

Rappel du sujet

- **Type** “Jeu de l’Oie”
- **30 cases** à parcourir
- **2 joueurs**
- Chaque joueur lance **2 dés**
- **5 PV** par joueur
- **Plus de PV** => Joueur perds
- Arrive à la **dernière case** => Joueur gagne
- **Thème** pirates



Réunions de début et de fin de projet

Temps de travail en groupe avant les séances de TP du lundi

Communication via Discord + Drive

Rétroplanning pour suivre les tâches

Présentation de l'interface

Présentation de l'interface

JPlateau

PanelCase

JPion

DiceCoursePanel

JDicePanel

JButton

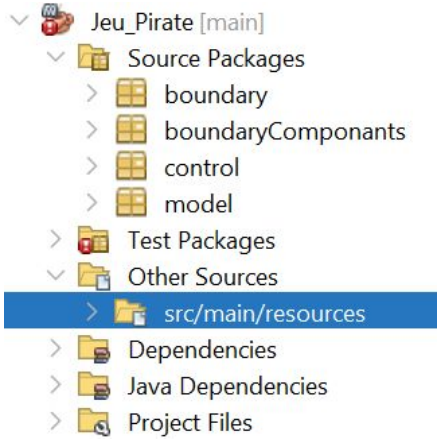
Présentation de l'interface

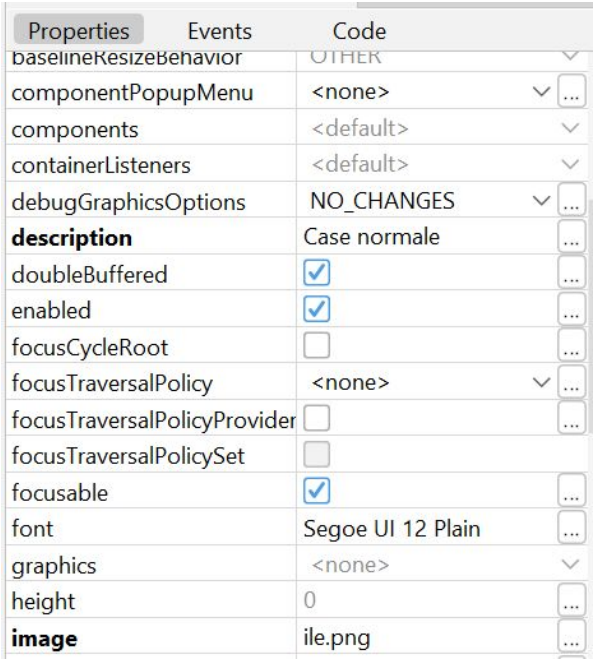
PanelInfosJoueur

- PanelTurn
- IconJoueur
- BarreDeVie
- JEffet

x2

Outils : FirePropertyChange & resources package






```java
public void setImage(String fichier){
    try {
        this.image = ImageIO.read(new File(getClass().getResource(fichier).toURI()));
    } catch (IOException | URISyntaxException ex) {
        Logger.getLogger(PanelImage.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

JInfoJoueurs : ImageDisplayer

```java
public abstract class ImageDisplayer extends javax.swing.JPanel {

    protected boolean x2Res = false;
    protected boolean alternative = false;

    public boolean isX2Res();
    public boolean isAlternative();
    public boolean setX2Res();
    public boolean setAlternative();

    /**
     * Mets à jour les fichiers images. Spécifique aux classes enfant.
     */
    protected abstract void updateImagesFiles();
```

```java
    /**
     * Renvoi une extension pour les images à utiliser en fonction des variables
     * x2res et alternative
     * @return extension du nom de fichier
     */
    protected String getImageExtension() {
        String temp = "";

        if(x2Res)
            temp += "_2x";

        if(alternative)
            temp += "_alt";

        return temp + ".png";
    }
```

Généricité de fonctionnement | **Noms unifiés** d'assets | **Gestion des variantes** d'assets

JInfoJoueurs : JEffet

```java
private void updateImageFile() {
        switch(etat) {
            case PASSETOUR:
                image = "avalanche";
                break;
            case ESTPRISON:
                image = "lianes";
                break;
            case ESTPOURSUIVI:
                image = "monstre";
                break;
            default:
                if (changement<0){
                    image = "crane";
                }else if (changement>0){
                    image = "nourriture";
                }else{
                    image = null;
                }
                break;
        }
        if(image == null)
            return;

        try {
            effetImage = ImageIO.read(
                    getClass().getResource("/" + image + ".png"));

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

```java
private void updateLabels() {
        String nom, description;
        switch(etat) {
            case PASSETOUR:
                nom = "Bloqué";
                description = "Passe un tour";
                break;
            case ESTPRISON:
                nom = "Emprisonné";
                description = "Jusqu'à faire un 10";
                break;
            case ESTPOURSUIVI:
                nom = "Poursuivi";
                description = "Doit faire un 8";
                break;
            default:
                if (changement > 0){
                    nom = "Boost";
                    description = "+" + changement + " au prochain lancer";
                }else if (changement < 0){
                    nom = "Malus";
                    description = changement + " au prochain lancer";
                }else{
                    nom = "";
                    description = "";
                }
        }
        labelNomEffet.setText(nom);
        labelDescriptionEffet.setText(description);
    }
```

JInfoJoueurs : JBarreDeVie

```
protected void updateImagesFiles() {
        try {
            coeurPlein = ImageIO.read(
                    getClass().getResource("/" + imagePlein + getImageExtension()));
            coeurVide = ImageIO.read(
                    getClass().getResource("/" + imageVide + getImageExtension()));

            this.imWidth = coeurPlein.getWidth();

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

```
public void setNombrePV(int value) {
        // Pour l'édition dans le GUI
        firePropertyChange("Nombre PV", null, nombrePV);

        // Pour éviter les dépassements
        if(value > maxPV) {nombrePV = maxPV;}
        else if(value < 0) {nombrePV = 0;}
        else {nombrePV = value;}

        // On actualise l'affichage
        repaint();
    }
public void paintComponent(Graphics g) {
        super.paintComponent(g);

        // Afficher les points de vie
        for(int i = 0; i < nombrePV; i++) {
            g.drawImage(coeurPlein, i * imWidth, 0, null);
        }
        for(int i = nombrePV; i < maxPV; i++) {
            g.drawImage(coeurVide, i * imWidth, 0, null);
        }
    }
```

JInfoJoueurs : JPanelTurn

```
protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        try {
            image = ImageIO.read(getClass().getResource("/" + pathTurn));
            g.drawImage(image, 0, 0, this);
        } catch (IOException ex) {
            g.drawOval(0, 0, g.getClipBounds().width, g.getClipBounds().height);
            Logger.getLogger(JPion.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    public void setPathTurn(String pathTurn) {
        this.pathTurn = pathTurn;
        firePropertyChange("Turn path", null, pathTurn);
    }
```

JPlateau : JCase

```java
@Override
protected void paintComponent(Graphics g) {
    super.paintComponent(g); // Generated from
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/OverriddenMethodBody
    if (image != null){
        Dimension panelDim = this.getSize();
        int xStart = (panelDim.width - imSize) / 2;
        int yStart = (panelDim.height - imSize) / 2;
        g.drawImage(image, xStart, yStart, imSize, imSize, this);
    }
    drawNumeroCase(g);
    drawBorder(g);
    if (sombre || active){
        drawOpacite(g);
    }
}
```

```java
private void drawNumeroCase(Graphics g){
    g.setColor(Color.BLACK);
    g.setFont(new java.awt.Font("Segoe UI", 1, 12));
    g.drawString(String.valueOf(numero), 5, 15);
}

private void drawBorder(Graphics g){
    Graphics2D g2D = (Graphics2D) g;
    g2D.setStroke(new BasicStroke(5));
    g2D.setColor(couleur);
}

private void drawOpacite(Graphics g){
    g.setColor(couleur);
    g.fillRect(0, 0, getWidth(), getHeight());
}
```

JPlateau : JCase

```
public void caseSombre(){
        if (!active){
            this.couleur = NOIR;
        }
        this.sombre = true;
        repaint();
    }

    public void caseClaire(){
        this.sombre = false;
        repaint();
    }
```


```
public void caseValide(){
        this.couleur = VERT;
        this.active = true;
        timer.start();
        repaint();
    }

    public void caseInvalide(){
        this.couleur = ROUGE;
        this.active = true;
        timer.start();
        repaint();
    }
```


```
private void timerEventHandler(ActionEvent e){
        timer.stop();
        this.active = false;
        if (sombre){
            this.couleur = NOIR;
        }
        repaint();
    }
```

JPlateau : JPion

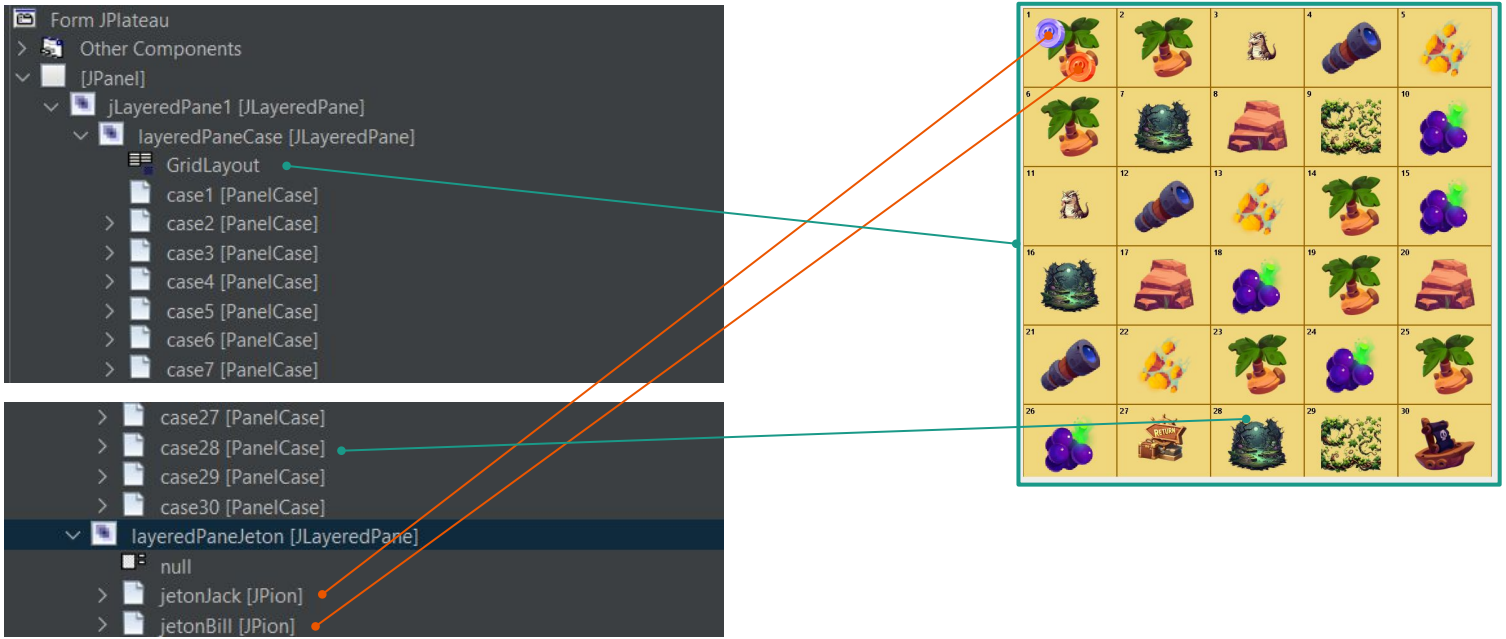
```java
private String pathProperty;
private BufferedImage image;
private boolean movable;
```

```java
public boolean isMovable() {
    return movable;
}

public void setMovable(boolean movable) {
    this.movable = movable;
    firePropertyChange("movable", null, movable);
}
```

JPlateau : JPion

```java
@Override
protected void paintComponent(Graphics g) {
    if (pathProperty == null)
        g.drawOval(0, 0, g.getClipBounds().width, g.getClipBounds().height);
    else {
        try {
            image = ImageIO.read(getClass().getResource("/" + pathProperty));
            g.drawImage(image, 0, 0, this);
        } catch (IOException ex) {
            g.drawOval(0, 0, g.getClipBounds().width, g.getClipBounds().height);
            Logger.getLogger(JPion.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}

public void setPathProperty(String path) {
    this.pathProperty = path;
    firePropertyChange("Image path", null, pathProperty);
}
```

JPlateau


Form JPlateau
Other Components
[JPanel]
jLayeredPane1 [JLayeredPane]
layeredPaneCase [JLayeredPane]
GridLayout
case1 [PanelCase]
case2 [PanelCase]
case3 [PanelCase]
case4 [PanelCase]
case5 [PanelCase]
case6 [PanelCase]
case7 [PanelCase]
case27 [PanelCase]
case28 [PanelCase]
case29 [PanelCase]
case30 [PanelCase]
layeredPaneJeton [JLayeredPane]
null
jetonJack [JPion]
jetonBill [JPion]

JPlateau : Ensemble

```java
private List<PanelCase> cases ;
private Dialog dialog;
private Point positionBill;
private Point positionJack;
```

```java
public void setImage(String[] images) {
    for (int i = 0; i < cases.size(); i++) {
        PanelCase caseAct = cases.get(i);
        caseAct.setImage(images[i]);
    }
}

public void setDescriptions(String[] descriptions) {
    for (int i = 0; i < cases.size(); i++) {
        PanelCase caseAct = cases.get(i);
        caseAct.setDescription(descriptions[i]);
    }
}
```

```java
public void plateauModeDeplacement(int num){
    IntStream.range(0, cases.size())
            .filter(i -> i != num)
            .forEach(i -> cases.get(i).caseSombre());
}

public void plateauModeNormal(){
    IntStream.range(0, cases.size())
            .forEach(i -> cases.get(i).caseClaire());
}
```

JPlateau : Ensemble

```
private void initCase(){
    cases = new ArrayList<>();
    Component[] composants = layeredPaneCase.getComponents();
    for (Component composant : composants) {
        if (composant instanceof PanelCase panelCase) {
            cases.add(panelCase);
        }
    }
    Collections.sort(cases , (PanelCase case1, PanelCase case2) -> {
    int posCase1 = layeredPaneCase.getPosition(case1);
    int posCase2 = layeredPaneCase.getPosition(case2);
    return Integer.compare(posCase1, posCase2);
    });
    for (int i = 0; i < cases.size(); i++) {
        PanelCase caseAct = cases.get(i);
        caseAct.setNumero(i + 1);
        caseAct.setImage("ile.png"); // pour les images personaliser
    }
}
```

JPlateau : Ensemble

```
private void jetonJackMouseDragged(java.awt.event.MouseEvent evt) {
    if(jetonJack.isMovable()) {
        jetonDeplacement(jetonJack);
    }
}

private void jetonJackMouseReleased(java.awt.event.MouseEvent evt) {
    if(jetonJack.isMovable()) {
        int positionCase = getCaseOnJeton(jetonJack);
        // TODO : informer le JDialog que le pion est a cette case
        dialog.eventDeplacement(positionCase);
    }
}
```

```
private void jetonDeplacement(JPion jeton){
    int newX = (int) this.getMousePosition().getX()- jeton.getWidth() / 2;
    int newY = (int) this.getMousePosition().getY() - jeton.getHeight() / 2;
    jeton.setLocation(newX, newY);
}

public void replacerJeton(JPion jeton, int numCase){
    Point cornerCase = getCase(numCase).getBounds().getLocation();
    Point pawnLocation = jeton.getPathProperty().equals("pion_bill.png")
      ? cornerCase : new Point(cornerCase.x + jeton.getWidth(), cornerCase.y);
    jeton.setLocation(pawnLocation);
}

public int getCaseOnJeton(JPion jeton){
    int res = 0;
    for (int i = 0; i < cases.size(); i++) {
        Rectangle bounds = cases.get(i).getBounds();
        if(bounds.contains(jeton.getX(), jeton.getY())){
            res = i;
            break;
        }
    }
    return res;
}
```

JPlateau : Ensemble

Dés : JDicePanel

```
public class JDicePanel extends javax.swing.JPanel {


    private BufferedImage image;
    private int number;


    /**
     * Creates new form mon_panel
     */
    public JDicePanel() {
        initComponents();
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
        this.setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 206, Short.MAX_VALUE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 181, Short.MAX_VALUE)
        );
    }// </editor-fold>
```

```
    public void changeImage (BufferedImage im){
        this.image = im;
        repaint();
    }

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        Graphics2D ds = (Graphics2D)g;
        ds.drawImage(image, 0, 0,getWidth(), getHeight(), this);

    }

    /**
     * @return the number
     */
    public int getNumber() {
        return number;
    }

    /**
     * @param number the number to set
     */
    public void setNumber(int number) {
        this.number = number;
    }

    // Variables declaration - do not modify
    // End of variables declaration
}
```

Dés : JDiceCoursePanel

1. Attributs

```
private List <BufferedImage> buffer;
private JDicePanel diceArray[];
private Timer timerArray[];

private Random seed;
```

```
private Dialog dialog;
// Variables declaration - do not modify
private boundaryComponants.JDicePanel dicePanel1;
private boundaryComponants.JDicePanel dicePanel2;
private javax.swing.JButton jButtonThrow;
// End of variables declaration
```

Dés : JDiceCoursePanel

2. Constructeur

```
public DiceCoursePanel() {
    this.buffer = new ArrayList<>();

    this.seed = new Random();

    this.diceArray = new JDicePanel[2];
    this.timerArray = new Timer[3];

    try {
        buffer.add(ImageIO.read(getClass().getResource("/de1.png")));
        buffer.add(ImageIO.read(getClass().getResource("/de2.png")));
        buffer.add(ImageIO.read(getClass().getResource("/de3.png")));
        buffer.add(ImageIO.read(getClass().getResource("/de4.png")));
        buffer.add(ImageIO.read(getClass().getResource("/de5.png")));
        buffer.add(ImageIO.read(getClass().getResource("/de6.png")));
    }catch (IOException e) {
        e.printStackTrace();
    }

    initComponents();

    diceArray[0] = dicePanel1;
    diceArray[1] = dicePanel2;
```

```
    timerArray[0] = new Timer(100, e -> {
        // Changer l'image du dé aléatoirement
        int index = seed.nextInt(6);
        diceArray[0].changeImage(getBuffer().get(index));
    });
    timerArray[1] = new Timer(100, e -> {
        // Changer l'image du dé aléatoirement
        int index = seed.nextInt(6);
        diceArray[1].changeImage(getBuffer().get(index));
    });
    timerArray[2] = new Timer(2000, e -> {
        timerArray[0].stop();
        timerArray[1].stop();
        timerArray[2].stop();
        endAnimation();
    });

    diceArray[0].changeImage(getBuffer().get(0));
    diceArray[1].changeImage(getBuffer().get(5));
}
```

Dés : JDiceCoursePanel

3. Animation et lancer des dés

```
private void endAnimation(){
    dialog.eventLancerDe();
}

/**
 * @return the buffer
 */
public List <BufferedImage> getBuffer() {
    return buffer;
}

public void setDialog(Dialog dialog){
    this.dialog = dialog;
}

/**
 * @return the seed
 */
public Random getSeed() {
    return seed;
}
```

```
public void diceAnimation(){
    timerArray[0].start();
    timerArray[1].start();
    timerArray[2].start();
}

public void enablejButtonThrow(boolean enabled) {
    jButtonThrow.setEnabled(enabled);
}

public void setValuesDice(int[] values){
    dicePanel1.changeImage(getBuffer().get(values[0] - 1));
    dicePanel2.changeImage(getBuffer().get(values[1] - 1));
}
```

Dialog

Dialog

Adaptateur

```
(int[])adaptateur.getResultatsDes();
(int)adaptateur.getPirateCourant();
(int)adaptateur.getProchaineCase();
(int)adaptateur.getChangement();
(int)adaptateur.getPointsVie();
(int)adaptateur.getResultat();
(Etat)adaptateur.getEtat();
```

Dialog

Dialog

```
changement = adaptateur.getChangement();
frame.getComposant().setChangement(changement);
```

Dialog

Fin dialog

```
adaptateur.confirmationChangement();
```

Démonstration

Partie très rapide

Démonstration

Partie rapide

Démonstration

Partie très lente

Conclusion : retour d'expérience

Liaison entre IHM et noyau fonctionnel

Gestion des images au travers du getResources

Dynamisme des composants en fonction de la taille de la fenêtre

Mise en pratique de compétences vues en cours et en TP d'IHM

Création de composants personnalisés complexes (assemblages de composants)

Développement d'un projet complet de A à Z avec une sortie graphique fonctionnelle

Merci pour votre écoute

Liens utiles & Annexes

GitHub - https://github.com/emilien3/Jeu_Pirate
Moodle ILU4 - https://moodle.univ-tlse3.fr/course/view.php?id=6114

Assets Graphiques par KlyakSun sur Canva
Assets des dés et des cases spéciales générer par DALL-E puis retouchés