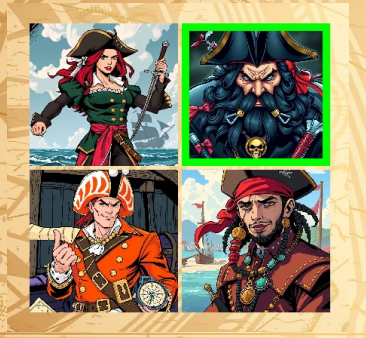
Présentation Partie IHM

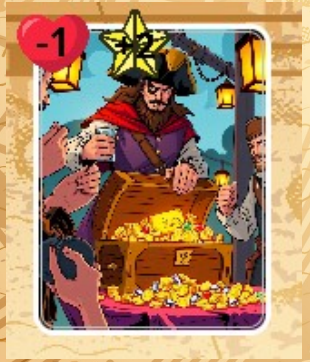
Démonstration du jeu

Répartition du code

Composants personnalisés



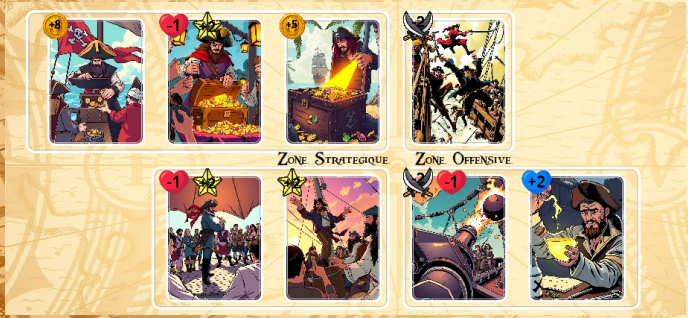
- PanelPirate



- PanelCarte

Composants personnalisés

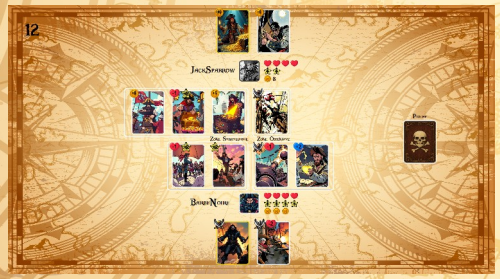
- PanelInformations



- PanelZoneCarte

Composants personnalisés

- PanelImage



- PanelPlateau

Séparation des parties (architecture de Seeheim)

```java
public class PanelCarte extends JPanel {
    // === MODÈLE (Données) ===
    private final Carte carte;
    private boolean bloque = false;

    // === PRÉSENTATION (UI) ===
    private Image image;
    private final Point pointDessin = new Point(0, 0);
    private boolean etaitDansZoneStrategique = false;
    private boolean etaitDansZoneOffensive = false;

    // === DIALOGUE (Contrôle) ===
    private final JPanel zoneStrategique, zoneOffensive, zoneMain;
    private final JLayeredPane zoneJeu;
    private Point decalageSouris;
    private boolean enDeplacement = false;
    private Rectangle rectangleStrategique, rectangleOffensive;
```

```java
// === PRÉSENTATION ===
private void initUI() {
    setOpaque(false);
    setPreferredSize(new Dimension(149, 190));
    setToolTipText("<html>" + carte.toString().replace("\n", "<br>") + "</html>");
}

private void chargerImage() {
    try {
        String cheminImage = System.getProperty("user.dir") + "/" + carte.getCheminImage();
        image = ImageIO.read(new File(cheminImage));
    } catch (IOException e) {
        System.out.println("Erreur lors du chargement de l'image: " + e.getMessage());
    }
}

@Override
protected void paintComponent(Graphics g) {
    super.paintComponent(g);
    Graphics2D g2 = (Graphics2D) g.create();
    g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);
```

```java
// === DIALOGUE ===
private void initZones() {
    try {
        Point posStrategique = zoneStrategique.getLocationOnScreen();
        Point posOffensive = zoneOffensive.getLocationOnScreen();
        rectangleStrategique = new Rectangle(posStrategique.x, posStrategique.y, zoneStrategique.getWidth(), zoneStrategique.getHeig
        rectangleOffensive = new Rectangle(posOffensive.x, posOffensive.y, zoneOffensive.getWidth(), zoneOffensive.getHeight());
    } catch (IllegalComponentStateException ignored) {}
}

private void initInteractions() {
    addMouseListener(new MouseAdapter() {
        @Override
        public void mousePressed(MouseEvent e) {
            mousePressedActionPerformed(e);
        }

        @Override
        public void mouseReleased(MouseEvent e) {
            mouseReleasedActionPerformed(e);
        }
    });

    addMouseMotionListener(new MouseMotionAdapter() {
        @Override
```

```java
// === GETTERS/SETTERS ===
public Carte getCarte() {
    return carte;
}

public void setBloque(boolean bloque) {
    this.bloque = bloque;
}
```

Timers


```
panelJeu.add(labelTempsTour);
timerTour = new Timer(1000, (e) -> updateTempsTour(e));
//timerTour.addActionListener(this::updateTempsTour);
timerTour.start();
```

```
private void updateTempsTour(ActionEvent e) {
    tempsRestantTour--;
    if (tempsRestantTour <= 0) {
        changerTour();
    } else {
        labelTempsTour.setText("" + tempsRestantTour);
    }
}
```

```
private void changerTour() {
    tempsRestantTour = 30;
    labelTempsTour.setText("" + tempsRestantTour);
    timerTour.restart();

    controlJeu.passerAuJoueurSuivant();
    if(controlJeu.getJoueurActif() == 0){
        debloquerCartes(panelMainJoueur1);
        bloquerCartes(panelMainJoueur2);
        ((PanelImage) panelImagePirate1).setGrise(false);
        ((PanelImage) panelImagePirate2).setGrise(true);
    }else{
        debloquerCartes(panelMainJoueur2);
        bloquerCartes(panelMainJoueur1);
        ((PanelImage) panelImagePirate1).setGrise(true);
        ((PanelImage) panelImagePirate2).setGrise(false);
    }
    panelInformationsJoueur1.repaint();
    panelInformationsJoueur2.repaint();
```

Conclusion