PROJET JAVA

Montage de films de caractères

Sébastien **CUVELLIER**, Groupe 111
Fleur **LAURENS**, Groupe 104

Table des matières

Présentation de l'application

Introduction

Pour répondre au besoin de la société de production qui souhaite révolutionner le marché avec le format TXT, nous allons travailler pour proposer une librairie permettant de fournir les différentes options de montage demandé dans le cahier des charges. Qui seront ensuite utilisé par une autre équipe, qui elle s'occupera de l'interface graphique.

Problématique

Produire une librairie pour aider l'équipe de développement à la confection du nouveau logiciel destiné au montage des films respectant le format txt.

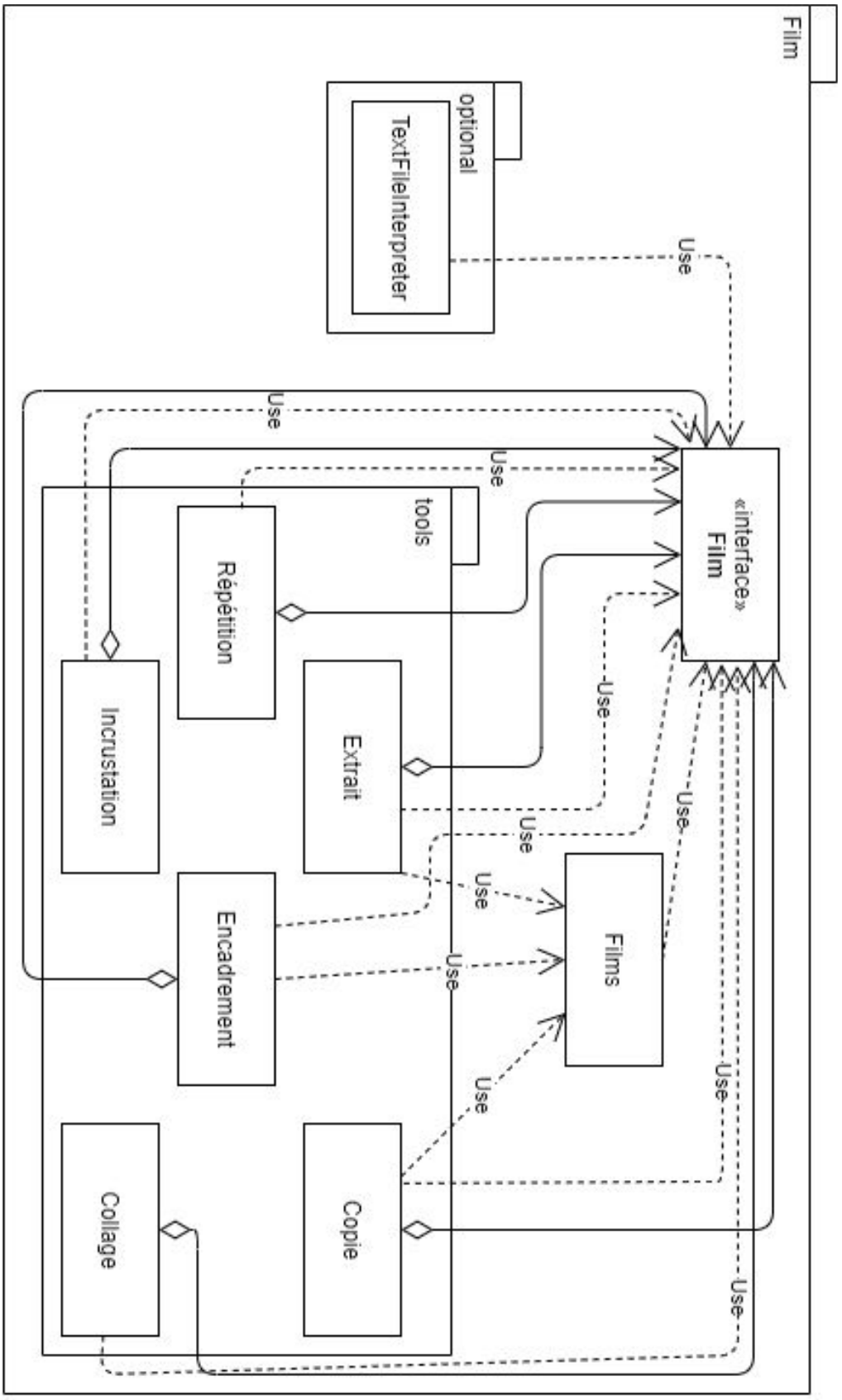
Rôle fonctionnel du projet

Le but de ce projet est de développer un ensemble d'opération entre un ou plusieurs films, tout cela en facilitant l'utilisation de cette librairie pour une supposé équipe chargé de la productions de l'interface graphique. Il faut donc produire un code simple et il a été ici choisie d'utiliser une interface qui donne une vue abstraite de ce qu'est un film au format txt. Composé de quatre méthodes simples accompagner des méthodes complète de la classe Films qui permettent au développeur de l'interface de gagner en compréhension et en efficacité, tout en proposant une cohérence.

Entrées et sorties

Une bibliothèque ne contient pas de programme principal, les entrées et sorties seront donc gérées par le programmeur utilisateur qui va créer un programme principal à partir de notre bibliothèque de classes.

Diagramme de classes UML



Organisation des tests

Pour organiser les tests de l'application, nous avons, pour chaque classe, écrit des fichier de test JUnit. Nous avons utilisé les différents mots clés assertEquals , assertTrue, assertFalse ... disponible dans la librairie de JUnit. Cela nous a permis de vérifier le bon fonctionnement de tous les éléments de notre application.

Bilan

Tous nos tests s'exécutent avec succès avec un coverage du code assez correct.
Voir l'annexe 1 pour les codes des fichier de test JUnit.

Bilan sur le projet

Pendant ce projet, nous avons renforcé nos connaissances de la conception et de la programmation en paradigme objet. Nous avons ainsi pu expérimenter l'utilisation de nouveaux composants comme les interfaces.

Nous avons été nettement plus productifs que lors des projets précédents. En effet, nous continuons d'utiliser des outils de gestions de code très utiles que nous maîtrisons de plus en plus dont principalement GitHub qui nous permet de faire un versionning et le principe de branche qui nous est de plus en plus familier.

D'un point de vue plus général, ce projet a été une très bonne expérience ; elle nous a permis de confirmer les compétences déjà acquises en Java et en programmation objet ainsi que d'en acquérir de nouvelles dans de bonnes conditions. Le sujet, à la fois ludique et intéressant, nous a permis de ressentir l'esprit du développeur. De plus, la création d'une bibliothèque de classe était pour nous une expérience nouvelle, tout à fait enrichissante.

Pour ce qui est des difficultés rencontrées, nous avons parfois eu des problèmes surtout au début du projet. En effet nous n'avions pas assez bien compris le fonctionnement de la méthode suivante qui nous a donc poser pas mal de problème mais qui auront finalement été résolue par le temps en s'entraînent d'abord sur le développement de petit film comme l'exemple fournit (la diagonale du fou). Nous avons dû également proposer de faire une copie profonde d'un film car si nous mettions un film en collage avec lui même de nombreux problème apparaissait au niveaux des références. Nous avons donc pris le soin d'ajouter une classe Copie. Nous avons également eu des problèmes avec les fichiers fournis principalement avec les accents malgrés des essai de nombreux réglage de l'encodage, c'est pourquoi tout les méthodes ont été écrite ou réécrite sans accents.

Enfin, l'aspect collaboratif de ce projet fait partie de ses points forts. En effet, la motivation mutuelle est un moteur puissant qui permet de surmonter tout type de difficulté. De plus, la résonance de deux esprits focalisés sur le même objectif, communicant sur un problème, est une des formes les plus efficaces de réflexion. Chacun comblant les faiblesses de l'autre, nous avons pu nous tirer mutuellement vers le haut. Ayant déjà l'expérience du travail en commun, nous avons renforcé notre niveau de collaboration et cela nous a permis d'avancer encore plus, chacun connaissant les spécificités de l'autre.

Annexe 1, Tests Unitaires

CollageTest:

```
package fr.cuvellier_laurens.film_maker.test.tools;

import fr.cuvellier_laurens.film_maker.film.Film;
import fr.cuvellier_laurens.film_maker.film.Films;
import
fr.cuvellier_laurens.film_maker.test.necessary_for_testing.LaDiagonaleD
uFou;
import
fr.cuvellier_laurens.film_maker.test.necessary_for_testing.LaLigneDuFou
;
import fr.cuvellier_laurens.film_maker.film.tools.Collage;
import org.junit.Test;

import java.util.ArrayList;

import static org.junit.Assert.assertEquals;

class CollageTest {

    @Test
    void hauteur() {
        Film film = new Collage(new LaLigneDuFou(), new
LaDiagonaleDuFou());
        assertEquals(4, film.hauteur());
    }

    @Test
    void largeur() {
        Film film = new Collage(new LaLigneDuFou(), new
LaDiagonaleDuFou());
        assertEquals(4, film.largeur());
    }

    @Test
    void suivante() {
```

```
        Film f2 = new Collage(new LaDiagonaleDuFou(), new
LaLigneDuFou());
        char[][] ecran = new char[f2.hauteur()][f2.largeur()];
        ArrayList<char[][]> ordre = new ArrayList<>();

        ordre.add(new char[f2.hauteur()][f2.largeur()]);
        Films.effacer(ordre.get(ordre.size() - 1));
        ordre.get(ordre.size() - 1)[0][0] = 'a';

        ordre.add(new char[f2.hauteur()][f2.largeur()]);
        Films.effacer(ordre.get(ordre.size() - 1));
        ordre.get(ordre.size() - 1)[1][1] = 'a';

        ordre.add(new char[f2.hauteur()][f2.largeur()]);
        Films.effacer(ordre.get(ordre.size() - 1));
        ordre.get(ordre.size() - 1)[2][2] = 'a';

        ordre.add(new char[f2.hauteur()][f2.largeur()]);
        Films.effacer(ordre.get(ordre.size() - 1));
        ordre.get(ordre.size() - 1)[3][3] = 'a';

        ordre.add(new char[f2.hauteur()][f2.largeur()]);
        Films.effacer(ordre.get(ordre.size() - 1));
        ordre.get(ordre.size() - 1)[0][0] = 'a';

        ordre.add(new char[f2.hauteur()][f2.largeur()]);
        Films.effacer(ordre.get(ordre.size() - 1));
        ordre.get(ordre.size() - 1)[0][1] = 'a';

        for (char[][] chars : ordre) {
            Films.effacer(ecran);
            f2.suivante(ecran);
            for (int i = 0; i < chars.length; ++i)
                for (int j = 0; j < chars[i].length; ++j)
                    assertEquals(chars[i][j], ecran[i][j]);
        }
    }

    @Test
    void rembobiner() {
```

```
        Film film = new Collage(new LaLigneDuFou(), new
LaDiagonaleDuFou());
        char[][] ecran1 = new char[film.hauteur()][film.largeur()];
        ArrayList<char[][]> ordre1 = new ArrayList<>();
        while (film.suivante(ecran1)) {
            ordre1.add(ecran1.clone());
        }
        film.rembobiner();
        for (char[][] chars : ordre1) {
            film.suivante(ecran1);
            for (int i = 0; i < chars.length; ++i)
                for (int j = 0; j < chars[i].length; ++j)
                    assertEquals(chars[i][j], ecran1[i][j]);
        }
    }

    @Test
    public void test2() {
        Film f = new LaDiagonaleDuFou();
        Film film = new Collage(f, f);

        char[][] ecran = Films.getEcran(film);
        int nb = 0;
        while (film.suivante(ecran))
            ++nb;
        assertEquals(8, nb);
    }
}
```

CopieTest:

```
package fr.cuvellier_laurens.film_maker.test.tools;

import fr.cuvellier_laurens.film_maker.film.Film;
import fr.cuvellier_laurens.film_maker.film.Films;
import fr.cuvellier_laurens.film_maker.film.tools.Copie;
import fr.cuvellier_laurens.film_maker.film.tools.Extrait;
```

```
import
fr.cuvellier_laurens.film_maker.test.necessary_for_testing.LaDiagonaleD
uFou;
import
fr.cuvellier_laurens.film_maker.test.necessary_for_testing.LaLigneDuFou
;
import org.junit.Test;

import java.util.ArrayList;

import static org.junit.Assert.assertEquals;

class CopieTest {

    @Test
    void hauteur() {
        Film film = new Copie(new LaLigneDuFou());
        assertEquals(2, film.hauteur());
    }

    @Test
    void largeur() {
        Film film = new Copie(new LaLigneDuFou());
        assertEquals(2, film.largeur());
    }


    @Test
    void suivante() {
        Film f2 = new Copie(new LaDiagonaleDuFou());
        char[][] ecran = new char[f2.hauteur()][f2.largeur()];
        ArrayList<char[][]> ordre = new ArrayList<>();

        ordre.add(new char[f2.hauteur()][f2.largeur()]);
        Films.effacer(ordre.get(ordre.size() - 1));
        ordre.get(ordre.size() - 1)[0][0] = 'a';

        ordre.add(new char[f2.hauteur()][f2.largeur()]);
        Films.effacer(ordre.get(ordre.size() - 1));
        ordre.get(ordre.size() - 1)[1][1] = 'a';
```

```
        ordre.add(new char[f2.hauteur()][f2.largeur()]);
        Films.effacer(ordre.get(ordre.size() - 1));
        ordre.get(ordre.size() - 1)[2][2] = 'a';

        ordre.add(new char[f2.hauteur()][f2.largeur()]);
        Films.effacer(ordre.get(ordre.size() - 1));
        ordre.get(ordre.size() - 1)[3][3] = 'a';

        for (char[][] chars : ordre) {
            Films.effacer(ecran);
            f2.suivante(ecran);
            for (int i = 0; i < chars.length; ++i)
                for (int j = 0; j < chars[i].length; ++j)
                    assertEquals(chars[i][j], ecran[i][j]);
        }
    }

    @Test
    void rembobiner() {
        Film film = new Extrait(new LaLigneDuFou(),2,4);
        char[][] ecran1 = new char[film.hauteur()][film.largeur()];
        ArrayList<char[][]> ordre1 = new ArrayList<>();
        while (film.suivante(ecran1)) {
            ordre1.add(ecran1.clone());
        }
        film.rembobiner();
        for (char[][] chars : ordre1) {
            film.suivante(ecran1);
            for(int i=0;i<chars.length;++i)
                for(int j = 0; j<chars[i].length;++j)
                    assertEquals(chars[i][j], ecran1[i][j]);
        }
    }
}
```

EncadrementTest:

```
package fr.cuvellier_laurens.film_maker.test.tools;

import fr.cuvellier_laurens.film_maker.film.Film;
import fr.cuvellier_laurens.film_maker.film.Films;
import fr.cuvellier_laurens.film_maker.film.tools.Encadrement;
import
fr.cuvellier_laurens.film_maker.test.necessary_for_testing.LaLigneDuFou
;
import org.junit.Test;

import java.util.ArrayList;

import static org.junit.Assert.assertEquals;

class EncadrementTest {

    @Test
    void hauteur() {
        Film film = new Encadrement(new LaLigneDuFou());
        assertEquals(4, film.hauteur());
    }

    @Test
    void largeur() {
        Film film = new Encadrement(new LaLigneDuFou());
        assertEquals(4, film.largeur());
    }

    @Test
    void suivante() {
        Film f2 = new Encadrement(new LaLigneDuFou());

        ArrayList<char[][]> ordre = new ArrayList<>();

        //Ligne
        ordre.add(new char[f2.hauteur()][f2.largeur()]);
        Films.effacer(ordre.get(ordre.size() - 1));
```

```
ordre.get(ordre.size() - 1)[1][1] = 'a';
//bodure
ordre.get(ordre.size() - 1)[0][0] = '*';
ordre.get(ordre.size() - 1)[1][0] = '*';
ordre.get(ordre.size() - 1)[2][0] = '*';
ordre.get(ordre.size() - 1)[3][0] = '*';
ordre.get(ordre.size() - 1)[0][1] = '*';
ordre.get(ordre.size() - 1)[0][2] = '*';
ordre.get(ordre.size() - 1)[0][3] = '*';
ordre.get(ordre.size() - 1)[3][1] = '*';
ordre.get(ordre.size() - 1)[3][2] = '*';
ordre.get(ordre.size() - 1)[3][3] = '*';
ordre.get(ordre.size() - 1)[1][3] = '*';
ordre.get(ordre.size() - 1)[2][3] = '*';
//Ligne
ordre.add(new char[f2.hauteur()][f2.largeur()]);
Films.effacer(ordre.get(ordre.size() - 1));
ordre.get(ordre.size() - 1)[1][2] = 'a';
//bodure
ordre.get(ordre.size() - 1)[0][0] = '*';
ordre.get(ordre.size() - 1)[1][0] = '*';
ordre.get(ordre.size() - 1)[2][0] = '*';
ordre.get(ordre.size() - 1)[3][0] = '*';
ordre.get(ordre.size() - 1)[0][1] = '*';
ordre.get(ordre.size() - 1)[0][2] = '*';
ordre.get(ordre.size() - 1)[0][3] = '*';
ordre.get(ordre.size() - 1)[3][1] = '*';
ordre.get(ordre.size() - 1)[3][2] = '*';
ordre.get(ordre.size() - 1)[3][3] = '*';
ordre.get(ordre.size() - 1)[1][3] = '*';
ordre.get(ordre.size() - 1)[2][3] = '*';

char[][] ecran = new char[f2.hauteur()][f2.largeur()];
for (char[][] chars : ordre) {
    Films.effacer(ecran);
    f2.suivante(ecran);
    for(int i=0;i<chars.length;++i)
        for(int j = 0; j<chars[i].length;++j){
            assertEquals(chars[i][j], ecran[i][j]);
```

```
                }
            }
        }

        @Test
        void rembobiner() {
            Film film = new Encadrement(new LaLigneDuFou());
            char[][] ecran1 = new char[film.hauteur()][film.largeur()];
            ArrayList<char[][]> ordre1 = new ArrayList<>();
            while (film.suivante(ecran1)) {
                ordre1.add(ecran1.clone());
            }
            film.rembobiner();
            for (char[][] chars : ordre1) {
                film.suivante(ecran1);
                for(int i=0;i<chars.length;++i)
                    for(int j = 0; j<chars[i].length;++j)
                        assertEquals(chars[i][j], ecran1[i][j]);
            }
        }

}
```

ExtraittTest:

```
package fr.cuvellier_laurens.film_maker.test.tools;

import fr.cuvellier_laurens.film_maker.film.Film;
import fr.cuvellier_laurens.film_maker.film.Films;
import
fr.cuvellier_laurens.film_maker.test.necessary_for_testing.LaDiagonaleD
uFou;
import
fr.cuvellier_laurens.film_maker.test.necessary_for_testing.LaLigneDuFou
;
import fr.cuvellier_laurens.film_maker.film.tools.Extrait;
import org.junit.Test;
import java.util.ArrayList;
```

```java
import static org.junit.Assert.assertEquals;

class ExtraitTest {

    @Test
    void hauteur() {
        Film film = new Extrait(new LaLigneDuFou(),2,4);
        assertEquals(2, film.hauteur());
    }

    @Test
    void largeur() {
        Film film = new Extrait(new LaLigneDuFou(),2,4);
        assertEquals(2, film.largeur());
    }

    @Test
    void suivante() {
        Film f2 = new Extrait(new LaDiagonaleDuFou(),1,2);

        char[][] ecran = new char[f2.hauteur()][f2.largeur()];
        ArrayList<char[][]> ordre = new ArrayList<>();

        ordre.add(new char[f2.hauteur()][f2.largeur()]);
        Films.effacer(ordre.get(ordre.size() - 1));
        ordre.get(ordre.size() - 1)[1][1] = 'a';

        ordre.add(new char[f2.hauteur()][f2.largeur()]);
        Films.effacer(ordre.get(ordre.size() - 1));
        ordre.get(ordre.size() - 1)[2][2] = 'a';

        for (char[][] chars : ordre) {
            Films.effacer(ecran);
            f2.suivante(ecran);
            for(int i=0;i<chars.length;++i)
                for(int j = 0; j<chars[i].length;++j)
                    assertEquals(chars[i][j], ecran[i][j]);
        }
    }
```

```
    @Test
    void rembobiner() {
        Film film = new Extrait(new LaLigneDuFou(),2,4);
        char[][] ecran1 = new char[film.hauteur()][film.largeur()];
        ArrayList<char[][]> ordre1 = new ArrayList<>();
        while (film.suivante(ecran1)) {
            ordre1.add(ecran1.clone());
        }
        film.rembobiner();
        for (char[][] chars : ordre1) {
            film.suivante(ecran1);
            for(int i=0;i<chars.length;++i)
                for(int j = 0; j<chars[i].length;++j)
                    assertEquals(chars[i][j], ecran1[i][j]);
        }
    }
}
```

IncrustationTest:

```
package fr.cuvellier_laurens.film_maker.test.tools;

import fr.cuvellier_laurens.film_maker.film.Film;
import fr.cuvellier_laurens.film_maker.film.Films;
import fr.cuvellier_laurens.film_maker.film.tools.Incrustation;
import
fr.cuvellier_laurens.film_maker.test.necessary_for_testing.LaDiagonaleD
uFou;
import
fr.cuvellier_laurens.film_maker.test.necessary_for_testing.LaLigneDuFou
;
import org.junit.Test;

import java.util.ArrayList;

import static org.junit.Assert.assertEquals;

class IncrustationTest {
    @Test
```

```
    void hauteur() {
        Film film = new Incrustation(new LaDiagonaleDuFou(), new
LaLigneDuFou(), 0,0);
        assertEquals(4, film.hauteur());
    }

    @Test
    void largeur() {
        Film film = new Incrustation(new LaDiagonaleDuFou(), new
LaLigneDuFou(), 0,0);
        assertEquals(4, film.largeur());
    }

    @Test
    void suivante() {
        Film f2 = new Incrustation(new LaDiagonaleDuFou(), new
LaLigneDuFou(), 0,0);

        char[][] ecran = new char[f2.hauteur()][f2.largeur()];
        ArrayList<char[][]> ordre = new ArrayList<>();

        ordre.add(new char[f2.hauteur()][f2.largeur()]);
        Films.effacer(ordre.get(ordre.size() - 1));
        ordre.get(ordre.size() - 1)[0][0] = 'a';

        ordre.add(new char[f2.hauteur()][f2.largeur()]);
        Films.effacer(ordre.get(ordre.size() - 1));
        ordre.get(ordre.size() - 1)[0][1] = 'a';

        ordre.add(new char[f2.hauteur()][f2.largeur()]);
        Films.effacer(ordre.get(ordre.size() - 1));
        ordre.get(ordre.size() - 1)[2][2] = 'a';

        ordre.add(new char[f2.hauteur()][f2.largeur()]);
        Films.effacer(ordre.get(ordre.size() - 1));
        ordre.get(ordre.size() - 1)[3][3] = 'a';

        for (char[][] chars : ordre) {
            Films.effacer(ecran);
            f2.suivante(ecran);
```

```
            for(int i=0;i<chars.length;++i)
                for(int j = 0; j<chars[i].length;++j)
                    assertEquals(chars[i][j], ecran[i][j]);
        }
    }

    @Test
    void rembobiner() {
        Film film = new Incrustation(new LaDiagonaleDuFou(), new
LaLigneDuFou(), 0,0);
        char[][] ecran1 = new char[film.hauteur()][film.largeur()];
        ArrayList<char[][]> ordre1 = new ArrayList<>();
        while (film.suivante(ecran1)) {
            ordre1.add(ecran1.clone());
        }
        film.rembobiner();
        for (char[][] chars : ordre1) {
            film.suivante(ecran1);
            for(int i=0;i<chars.length;++i)
                for(int j = 0; j<chars[i].length;++j)
                    assertEquals(chars[i][j], ecran1[i][j]);
        }
    }

    @Test
    public void test2() {
        Film f = new LaDiagonaleDuFou();
        Film film = new Incrustation(f,f, 1,1);
        char[][] ecran = Films.getEcran(film);
        int nb = 0;
        while (film.suivante(ecran))
            ++nb;
        assertEquals(4, nb);
    }


}
```

RepetitionTest:

```
package fr.cuvellier_laurens.film_maker.test.tools;

import fr.cuvellier_laurens.film_maker.film.Film;
import fr.cuvellier_laurens.film_maker.film.tools.Repetition;
import
fr.cuvellier_laurens.film_maker.test.necessary_for_testing.LaDiagonaleD
uFou;
import
fr.cuvellier_laurens.film_maker.test.necessary_for_testing.LaLigneDuFou
;
import org.junit.Test;

import java.util.ArrayList;
import java.util.Arrays;

import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertFalse;

class RepetitionTest {

    @Test
    void repetitionNull(){
        Film film = new Repetition(new LaDiagonaleDuFou(),0);
        char[][] ecran = new char[film.hauteur()][film.largeur()];
        assertFalse(film.suivante(ecran));
    }

    @Test
    void repetitionNegative(){
        Film film = new Repetition(new LaDiagonaleDuFou(),-4);
        char[][] ecran = new char[film.hauteur()][film.largeur()];
        assertFalse(film.suivante(ecran));
    }

    @Test
    void hauteur() {
        Film film = new Repetition(new LaDiagonaleDuFou(),2);
```

```
    assertEquals(4, film.hauteur());
}

@Test
void largeur() {
    Film film = new Repetition(new LaDiagonaleDuFou(),2);
    assertEquals(4, film.largeur());
}

@Test
void suivante() {
    Film f2 = new Repetition(new LaLigneDuFou(),2);
    char[][] ecran = new char[f2.hauteur()][f2.largeur()];
    ArrayList<char[][]> ordre = new ArrayList<>();

    ordre.add(new char[f2.hauteur()][f2.largeur()]);
    effacer(ordre.get(ordre.size() - 1));
    ordre.get(ordre.size() - 1)[0][0] = 'a';

    ordre.add(new char[f2.hauteur()][f2.largeur()]);
    effacer(ordre.get(ordre.size() - 1));
    ordre.get(ordre.size() - 1)[0][1] = 'a';

    ordre.add(new char[f2.hauteur()][f2.largeur()]);
    effacer(ordre.get(ordre.size() - 1));
    ordre.get(ordre.size() - 1)[0][0] = 'a';

    ordre.add(new char[f2.hauteur()][f2.largeur()]);
    effacer(ordre.get(ordre.size() - 1));
    ordre.get(ordre.size() - 1)[0][1] = 'a';

    for (char[][] chars : ordre) {
        effacer(ecran);
        f2.suivante(ecran);
        for(int i=0;i<chars.length;++i)
            for(int j = 0; j<chars[i].length;++j)
                assertEquals(chars[i][j], ecran[i][j]);
    }
```

```
    }
    public static void effacer(char[][] ecran) {
        for (char[] ligne : ecran)
            Arrays.fill(ligne, ' ');
    }

    @Test
    void rembobiner() {
        Film film = new Repetition(new LaDiagonaleDuFou(),2);
        char[][] ecran1 = new char[film.hauteur()][film.largeur()];
        ArrayList<char[][]> ordre1 = new ArrayList<>();
        while (film.suivante(ecran1)) {
            ordre1.add(ecran1.clone());
        }
        film.rembobiner();
        for (char[][] chars : ordre1) {
            film.suivante(ecran1);
            for(int i=0;i<chars.length;++i)
                for(int j = 0; j<chars[i].length;++j)
                    assertEquals(chars[i][j], ecran1[i][j]);
        }
    }


}
```

Annexe 2, Code java du projet

Collage:

```java
package fr.cuvellier_laurens.film_maker.film.tools;

import fr.cuvellier_laurens.film_maker.film.Film;

/**
 * @author Sebastien CUVELLIER - Fleur LAURENS
 * @version 1.0 - 07/04/2020
 * Permet de mettre deux films à la suite
 */

public class Collage implements Film{
    private Film film1;
    private Film film2;

    /**
     * Constructeur qui initialise les deux films à coller
     * @param film1 premier film
     * @param film2 deuxième film
     */
    public Collage(Film film1, Film film2) {
        this.film1 = new Copie(film1);
        this.film2 = new Copie(film2);
    }

    /**
     * @see Film#hauteur()
     */
    @Override
    public int hauteur() {
        return Math.max(film1.hauteur(), film2.hauteur());
    }

    /**
     * @see Film#largeur()
```

```
     */
    @Override
    public int largeur() {
        return Math.max(film1.largeur(),film2.largeur());
    }

    /**
     * @see Film#suivante(char[][])
     */
    @Override
    public boolean suivante(char[][] ecran) {
        if(!film1.suivante(ecran))
            return film2.suivante(ecran);
        return true;
    }

    /**
     * @see Film#rembobiner()
     */
    @Override
    public void rembobiner() {
        film1.rembobiner();
        film2.rembobiner();
    }

}
```

Copie:

```
package fr.cuvellier_laurens.film_maker.film.tools;

import fr.cuvellier_laurens.film_maker.film.Film;
import fr.cuvellier_laurens.film_maker.film.Films;

import java.util.ArrayList;

/**
 * @author Sebastien CUVELLIER - Fleur LAURENS
 * @version 1.0 - 08/05/2020
```

```
 * Permet de faire une copie profonde de d'un Film
 */
public class Copie implements Film{
    private ArrayList<char[][]> images;
    private Film film;
    private int img;



    /**
     * Constructeur permet faire une copie d'un film
     * @param f film que l'on veut copier
     */
    public Copie(Film f) {
        this.film = f;
        this.img = 0;
        images = new ArrayList<>();
        lire();
    }

    /**
     * @see Film#hauteur()
     */
    @Override
    public int hauteur() {
        return film.hauteur();
    }

    /**
     * @see Film#largeur()
     */
    @Override
    public int largeur() {
        return film.largeur();
    }

    /**
     * @see Film#suivante(char[][]) ()
     */
    @Override
```

```
    public boolean suivante(char[][] ecran) {
        if (this.img >= this.images.size())
            return false;
        for(int i = 0; i< this.images.get(this.img).length; ++i)
            System.arraycopy(this.images.get(this.img)[i], 0, ecran[i],
0, this.images.get(this.img)[i].length);
        ++this.img;
        return true;
    }
    /**
     * @see Film#rembobiner()
     */
    @Override
    public void rembobiner() {
        img = 0;
    }

    private void lire(){
        boolean test = true;
        this.images.add(new char[hauteur()][largeur()]);
        Films.effacer(this.images.get(this.images.size() - 1));
        while (test) {
            test = film.suivante(this.images.get(this.images.size() -
1));
            if (test){
                this.images.add(new char[hauteur()][largeur()]);
                Films.effacer(this.images.get(this.images.size() - 1));
            }else{
                this.images.remove(this.images.size() -1);
            }
        }
        film.rembobiner();
    }
}
```

Encadrement:

```java
package fr.cuvellier_laurens.film_maker.film.tools;

import fr.cuvellier_laurens.film_maker.film.Film;
import fr.cuvellier_laurens.film_maker.film.Films;

/**
 * @author Sebastien CUVELLIER  - Fleur LAURENS
 * @version 3.0 - 03/05/2020
 * Permet d'encadrer un Film
 */
public class Encadrement implements Film{
    private Film film;
    private char motif;
    private int tailleBordure;

    /**
     * Constructeur qui permet d'encardrer un film
     *
     * @param film film que l'on veut encadrer.
     */
    public Encadrement(Film film) {
        this(film, 1, '*');
    }

    /**
     * Constructeur qui permet d'encardrer un film
     *
     * @param film          film que l'on veut encadrer.
     * @param tailleBordure taille de la bodure.
     * @param motif         motif avec lequel on veut entourer.
     */
    public Encadrement(Film film, int tailleBordure, char motif) {
        this.tailleBordure = tailleBordure;
        this.motif = motif;
        this.film = film;
    }
```

```
    /**
     * @see Film#hauteur()
     */
    @Override
    public int hauteur() {
        return film.hauteur() + 2 * tailleBordure;
    }

    /**
     * @see Film#largeur()
     */
    @Override
    public int largeur() {
        return film.largeur() + 2 * tailleBordure;
    }

    /**
     * @see Film#suivante(char[][])
     */
    @Override
    public boolean suivante(char[][] ecran) {
        char[][] sousEcran = new char[film.hauteur()][film.largeur()];
        Films.effacer(sousEcran);
        for (int i = 0; i < this.largeur(); ++i)
            for (int j = 0; j < tailleBordure; ++j) {
                ecran[j][i] = motif;
                ecran[this.hauteur() - j - 1][i] = motif;
            }
        for (int i = 0; i < this.hauteur(); ++i)
            for (int j = 0; j < tailleBordure; ++j) {
                ecran[i][j] = motif;
                ecran[i][this.largeur() - j - 1] = motif;
            }
        boolean suivant = film.suivante(sousEcran);
        for (int i = 0; i < sousEcran.length; ++i)
            System.arraycopy(sousEcran[i], 0, ecran[i + tailleBordure],
tailleBordure, sousEcran[i].length);
        return suivant;
    }
```

```
    /**
     * @see Film#rembobiner()
     */
    @Override
    public void rembobiner() {
        film.rembobiner();
    }
}
```

Extrait:

```
package fr.cuvellier_laurens.film_maker.film.tools;

import fr.cuvellier_laurens.film_maker.film.Film;
import fr.cuvellier_laurens.film_maker.film.Films;


/**
 * @version 1.0 - 08/04/2020
 * @author Sebastien CUVELLIER - Fleur LAURENS
 * Permet d'obtenir l'extrait d'un Film
 */
public class Extrait implements Film {
    private Film film;
    private int premiereImage;
    private int derniereImage;
    private int imageEnCour;

    /**
     * Constructeur qui permet d'initialiser le film,la première image
inclus et la dernière image inclus du film.
     * @param film Le film dont on veut un extrait
     * @param premiereImage La première image que l'on veut pour notre
nouveau film
     * @param derniereImage La dernière image que l'on veut pour notre
nouveau film
     */
    public Extrait(Film film, int premiereImage, int derniereImage) {
```

```
        this.film = film;
        this.premiereImage = premiereImage;
        this.dernieresImage = dernieresImage;
        this.imageEnCour = 0;
    }

    /**
     * Constructeur qui permet d'initialiser le film et  la dernière
image inclus du film.
     * @param film Le film dont on veut un extrait
     * @param dernieresImage La dernière image que l'on veut pour notre
nouveau film
     */
    public Extrait(Film film, int dernieresImage) {
        this(film,0, dernieresImage);
    }

    /**
     * @see Film#hauteur()
     */
    @Override
    public int hauteur() {
        return film.hauteur();
    }

    /**
     * @see Film#largeur()
     */
    @Override
    public int largeur() {
        return film.largeur();
    }

    /**
     * @see Film#suivante(char[][])
     */
    @Override
    public boolean suivante(char[][] ecran) {
        if (imageEnCour < premiereImage) {
            for (int i = imageEnCour; i < premiereImage; ++i) {
```

```
                film.suivante(ecran);
                Films.effacer(ecran);
                ++this.imageEnCour;
            }
        }
        if (dernieresImage < imageEnCour) {
            return false;
        }
        ++this.imageEnCour;
        return film.suivante(ecran);
    }

    /**
     * @see Film#rembobiner()
     */
    @Override
    public void rembobiner() {
        film.rembobiner();
        this.imageEnCour = 0;
    }
}
```

Incrustation:

```
package fr.cuvellier_laurens.film_maker.film.tools;

import fr.cuvellier_laurens.film_maker.film.Film;
import fr.cuvellier_laurens.film_maker.film.Films;

/**
 * @author Sebastien CUVELLIER - Fleur LAURENS
 * @version 4.0 - 27/05/2020
 * Permet d'incruster un Film dans un autre Film
 */
public class Incrustation implements Film{
    private Film filmBase;
    private Film filmAIncruster;
    private int colone;
```

```
    private int ligne;
    private boolean filmVide = false;

    /**
     *
     * @param filmBase film dans lequel on incruste un film
     * @param filmAIncruster film que l'on incruste
     * @param colone colone à laquel on incruste le film
     * @param ligne ligne à laquel on incruste le film
     */
    public Incrustation(Film filmBase, Film filmAIncruster, int colone,
int ligne) {
        this.filmBase = new Copie(filmBase);
        this.filmAIncruster = new Copie(filmAIncruster);
        this.colone = Math.max(colone, 0);
        this.ligne = Math.max(ligne, 0);
        testerFilmVide();
    }

    /**
     * @see Film#hauteur()
     */
    @Override
    public int hauteur() {
        return this.filmBase.hauteur();
    }

    /**
     * @see Film#largeur()
     */
    @Override
    public int largeur() {
        return this.filmBase.largeur();
    }

    /**
     * @see Film#suivante(char[][])
     */
    @Override
    public boolean suivante(char[][] ecran) {
```

```
        if (filmVide)
            return filmBase.suivante(ecran);

        char[][] sousEcran1 = new
char[this.filmBase.hauteur()][this.filmBase.largeur()];
        char[][] sousEcran2 = new
char[this.filmAIncruster.hauteur()][this.filmAIncruster.largeur()];

        Films.effacer(sousEcran1);
        Films.effacer(sousEcran2);
        Films.effacer(ecran);
        boolean retourSuivant = this.filmBase.suivante(sousEcran1);
        filmAIncruster.suivante(sousEcran2);
        if (retourSuivant){
            for (int i = 0; i < hauteur(); ++i)
                for (int j = 0; j < largeur(); ++j){
                    if (i >= this.ligne && j >= this.colone && i -
this.ligne +1 <= this.filmAIncruster.hauteur() && j - this.colone + 1
<= this.filmAIncruster.largeur()){
                        if ( i - this.ligne  < sousEcran2.length && j -
this.colone < sousEcran2[i - this.ligne].length ){
                            ecran[i][j] = sousEcran2[i - this.ligne][j
- this.colone];
                        }
                    }
                    else
                    if( i < sousEcran1.length && j <
sousEcran1[i].length )
                        ecran[i][j] = sousEcran1[i][j];
                }

        }
        return retourSuivant;
    }

    /**
     * @see Film#rembobiner()
     */
    @Override
    public void rembobiner() {
```

```
        filmBase.rembobiner();
        filmAIncruster.rembobiner();
    }

    private void testerFilmVide(){
        //on verifie si le film est vide
        char[][] Ecran = new
char[this.filmAIncruster.hauteur()][this.filmAIncruster.largeur()];
        Films.effacer(Ecran);
        filmAIncruster.suivante(Ecran);
        boolean test = false;
        for (char[] images : Ecran)
            for (char image : images)
                if (image != ' ') {
                    test = true;
                    break;
                }
        if(!test)
            filmVide = true;
        filmAIncruster.rembobiner();
    }

}
```

Repetition:

```
package fr.cuvellier_laurens.film_maker.film.tools;

import fr.cuvellier_laurens.film_maker.film.Film;
import fr.cuvellier_laurens.film_maker.film.Films;

/**
 * @author Sebastien CUVELLIER - Fleur LAURENS
 * @version 2.3 - 27/05/2020
 *Permet de répéter un Film n fois
 */
public class Repetition implements Film {
    private int nombreDeRepetition;
    private int repetitionRestante;
```

```
    private Film film;
    private boolean filmVide = false;

    /**
     *Constructeur qui initialise le film et le nombre de répétition de
ce dernier.
     * @param film film que l'on veut répéter.
     * @param nombreDeRepetition nombre de fois que l'on veut le
répéter.
     */
    public Repetition(Film film, int nombreDeRepetition) {
        this(film);
        this.nombreDeRepetition = nombreDeRepetition;
        this.repetitionRestante = nombreDeRepetition;
    }
    /**
     *Constructeur optionnel qui initialise le film de ce dernier et le
nombre de répétition a une .
     * @param film film que l'on veut répéter.
     */
    public Repetition(Film film) {
        this.film = new Copie(film);
        this.nombreDeRepetition = 1;
        this.repetitionRestante = 1;
        testerFilmVide();
    }

    /**
     * @see Film#hauteur()
     */
    @Override
    public int hauteur() {
        return film.hauteur();
    }

    /**
     * @see Film#largeur()
     */
    @Override
    public int largeur() {
```

```
        return film.largeur();
    }

    /**
     * @see Film#suivante(char[][])
     */
    @Override
    public boolean suivante(char[][] ecran) {
        if(filmVide)
            return false;
            if (!film.suivante(ecran)) {
                --repetitionRestante;
                film.rembobiner();
                if (repetitionRestante !=0)
                    film.suivante(ecran);
            }
        return repetitionRestante > 0;
    }

    /**
     * @see Film#rembobiner()
     */
    @Override
    public void rembobiner() {
        repetitionRestante = nombreDeRepetition;
    }

    private void testerFilmVide(){
        //on verifie si le film est vide
        char[][] Ecran = new
char[this.film.hauteur()][this.film.largeur()];
        Films.effacer(Ecran);
        this.film.suivante(Ecran);
        boolean test = false;
        for (char[] images : Ecran)
            for (char image : images)
                if (image != ' ') {
                    test = true;
                    break;
                }
```

```
        if(!test)
            filmVide = true;
        this.film.rembobiner();
    }

}
```

Optionnel (TextFileInterpreter):

```
package fr.cuvellier_laurens.film_maker.film.optional;

import fr.cuvellier_laurens.film_maker.film.Film;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Scanner;

/**
 * @author Sebastien CUVELLIER - Fleur LAURENS
 * @version 2.3 - 08/04/2020
 * Permet de convertir un fichier txt en Film
 */
public class TextFileInterpreter implements Film {
    private ArrayList<String[]> images;
    private int hauteur;
    private int largeur;
    private String nom;
    private int marge;


    /**
     * Constructeur permetant d'initialiser le nom et la marge.
     * @param nom nom du fichier à ouvrir
     * @param marge marge si ligne sauté après chaque image.
     */
    public TextFileInterpreter(String nom, int marge) {
```

```
        this(nom);
        this.marge = marge;
    }

    /**
     * Constructeur permetant d'initialiser le nom.
     * @param nom nom du fichier à ouvrir
     */
    public TextFileInterpreter(String nom) {
        this.marge = 0;
        this.images = new ArrayList<>();
        this.nom = nom;
        try {
            lire(nom);
        } catch (FileNotFoundException e) {
            System.out.println("le fichier est introuvable");
        }
    }

    /**
     * @see Film#hauteur()
     */
    @Override
    public int hauteur() {
        return this.hauteur;
    }

    /**
     * @see Film#largeur()
     */
    @Override
    public int largeur() {
        return this.largeur;
    }

    /**
     * @see Film#suivante(char[][])
     */
    @Override
    public boolean suivante(char[][] ecran) {
```

```
        if (images.isEmpty())
            return false;
        for (int i = 0; i < this.images.get(0).length - this.marge;
++i) {
            char[] chars = this.images.get(0)[i].toCharArray();
            ecran[i] = chars;
        }
        this.images.remove(0);
        return true;
    }

    /**
     * @see Film#rembobiner()
     */
    @Override
    public void rembobiner() {
        while(!images.isEmpty())
            this.images.remove(0);
        try {
            lire(nom);
        } catch (FileNotFoundException e) {
            System.out.println("le fichier est introuvable");
        }
    }

    /**
     * Permet de lire un fichier qui respecte les règles du format
     *
     * @param nom Nom du fichier a lire
     * @throws FileNotFoundException Permet permet de dire si le
fichier est introuvable
     */
    private void lire(String nom) throws FileNotFoundException {
        int cpt = 0;
        Scanner in = new Scanner(new FileInputStream(nom));
        Scanner ligne = new Scanner(in.nextLine());
        this.largeur = Integer.parseInt(ligne.next());
        this.hauteur = Integer.parseInt(ligne.next());
        this.ajoutImage();
        ligne.close();
```

```
        while (in.hasNextLine()) {
            String ligneSuivante = in.nextLine();
            if (ligneSuivante.equals("\\newframe")) {
                this.ajoutImage();
                cpt = 0;
            } else {
                this.images.get(this.images.size() - 1)[cpt] =
ligneSuivante;
                ++cpt;
            }
        }
        in.close();
        this.ImageDeFinVide();
    }

    /**
     * Permet d'ajouter la place pour une image dans la liste des
images
     */
    private void ajoutImage() {
        this.images.add(new String[this.hauteur + this.marge]);
        Arrays.fill(images.get(this.images.size() - 1), " ");
    }

    /**
     * Permet de regarder si la dernière image est vide pour la
supprimer pour éviter une image vide de transition en cas d'assemblage
     */
    private void ImageDeFinVide() {
        int cpt = 0;
        for (int i = 0; i < this.hauteur + this.marge; ++i)
            if (this.images.get(this.images.size() - 1)[i].equals(" "))
                cpt++;
        if (cpt == this.hauteur + this.marge)
            this.images.remove(this.images.size() - 1);
    }
}
```