PARIS DESCARTES IUT – Département Informatique

Bases de la programmation orientée objet

– Projet –

Montage de films de caractères

1 Présentation du projet

Une société de production cinématographique a décidé de révolutionner la diffusion numérique des films qu'elle produit. Pour réaliser les économies d'énergie que nous impose le réchauffement climatique, il a été décidé de réduire de façon drastique le volume de donnée nécessaire à la diffusion des films. Après les réels progrès qu'ont été les formats DSM, DCMD, DCP, etc, le nouveau format proposé a rapidement rencontré un succès mondial.

Ce succès est essentiellement dû au développement de l'application CM-PLAYER qui offre un confort de visualisation inégalable pour les films respectant ce format. D'autre part, l'adaptation du film *Star Wars*[1] à ce nouveau format a permis de toucher le grand public.

Ce nouveau format a été nommé TXT. En effet, les pixels y sont représentés par des caractères remettant ainsi au goût du jour les premières oeuvres d'ASCII-ART des années 1960.

Après le succès du logiciel CM-PLAYER (que vous trouverez sur MOODLE), la société a décidé de développer un nouveau logiciel destiné au montage des films respectant le format TXT.

2 Travail à faire

Une première analyse a conduit à mettre au point l'interface `film.Film` donnant une vue abstraite de ce qu'est un film au format TXT.

Tout film est composée d'une suite d'images. Toutes les images d'un même film ont la même définition (nombre de lignes – méthode `hauteur()` – et nombre de colonnes – méthode `largeur()`). Les différentes images composant le film peuvent être obtenues en invoquant successivement la méthode `suivante()`. La valeur retournée par cette méthode permet de savoir si une prochaine image était disponible ou si toutes les images avaient déjà été récupérées. Enfin, la méthode `rembobiner()` doit permettre de récupérer à nouveau toutes les images du film (via des appels successifs à la méthode `suivante()`).

Un ensemble de méthodes utilitaires ont aussi été développées au sein de la classe `film.Films`. En particulier, la méthode de classe `Films.projeter()` permet de visualiser un film sur `System.in` alors que `Films.sauvegarder()` permet de sauvegarder un film au format accepté par le logiciel CM-PLAYER. La classe `film.Films` contient d'autres méthodes statiques de moindre importance.

Vous trouverez en annexe de ce document, le code de l'interface et de la classe utilitaire ainsi que celui d'un programme d'exemple implémentant l'interface `Film` et invoquant les méthodes de la classe utilitaire `Films`.

Votre travail consiste à proposer un ensemble d'éléments de code Java (interfaces, classes, énumérations, méthodes, ...) permettant de monter un film à partir de films existants. Vous ne devez pas faire un programme mais uniquement une bibliothèque permettant de combiner des films existants (i.e. des instances de l'interface `Film`) pour obtenir de nouveaux films. Le développement de l'interface graphique du logiciel de montage a été confié à une autre équipe de la société. Ce développement repose lui aussi sur l'interface `film.Film` présentée ci-dessus.

1. Ce film est toujours visible ici `https://www.youtube.com/watch?v=Dgwyo6JNTDA`

Votre bibliothèque doit permettre :

1. De répéter un film un nombre donné de fois.
2. D'obtenir un extrait d'un film. L'extrait est désigné par les numéros de la première et de la dernière images à inclure et la première image du film porte le numéro 0.
3. D'encadrer un film (par quatre ligne d'étoiles – '*' – au bord de l'écran).
4. De coller deux films, l'un à la suite de l'autre.
5. D'incruster un film dans un film. Le point d'incrustation sera désigné par les numéros de ligne et de colonne que doit prendre le coin en haut à gauche du film devant être incrusté dans les images du film où il est incrusté. Si

Le résultat de chacune de ses opérations doit être un nouveau film pouvant être projeté ou sauvegardé. De plus, il doit être possible de lui appliquer de nouvelles opérations (le répéter, en obtenir un extrait, ...).

Vous ferez en sorte que chacune de ces opérations ne provoque pas d'erreur. Par exemple, répéter un film un nombre de fois nul ou négatif doit conduire à un film vide. Il en est de même si le numéro de la dernière image d'un extrait de film est inférieur au numéro de la première image. Si les images de deux films mis en séquences n'ont pas la même taille, l'écran du film résultant de la mise en séquence doit être suffisamment grand pour contenir les images des deux films. Par contre, dans le cas où les images incrustées dépassent (en bas et/ou à droite) des images dans lesquelles elles sont incrustées, les images incrustées doivent être automatiquement tronquées (en bas et/ou à droite) de manière à tenir sur l'écran du film où elles sont incrustées.

Qui, quoi et quand ?

Votre projet doit être fait en binôme. Les groupes de 3 ne seront pas acceptés. Évitez de faire votre projet tout seul (soit vous êtes très fort et des personnes ont besoin de votre aide, soit vous avez des difficultés et il faut vous faire aider).

Vous devez porter une attention particulière à la rédaction de votre dossier. Sa qualité est déterminante pour l'évaluation de votre travail. Votre dossier doit être un unique document `pdf` dont la composition est la suivante :

— Une page de garde indiquant le nom et **le groupe** des membres du binôme, l'objet du dossier.
— Une table des matières de l'ensemble du dossier.
— Une brève introduction du projet.
— Le diagramme UML des classes formant vos applications. Seuls les noms des classes et leurs dépendances doivent être reportés, il est inutile de préciser les attributs et méthodes. Toutefois, vous devez y représenter l'organisation en paquetage.
— Le code Java des tests unitaires de vos classes (en précisant lesquels s'exécutent avec succès).
— Le code Java complet de votre projet. L'ordre dans lequel vous présentez vos classes facilite la lecture. Aller des classes élémentaires (celles qui ne dépendent d'aucune autre classe) aux classes plus complexes (en respectant l'ordre de dépendance) est un bon choix.
— Un bilan du projet (les difficultés rencontrées, ce qui est réussi, ce qui peut être amélioré).

Nous vous rappelons que le critère principal de notation est la structuration de votre application. Votre rapport doit mettre en avant cette qualité.

Vous devez rendre votre projet le **lundi 25 mai 2020** en déposant sur Moodle, une **archive** nécessairement au format **`zip`** contenant :

— Votre **rapport** sous la forme d'un fichier **`pdf`**.
— Un répertoire nommé **`src`** et contenant tous vos **fichiers Java**. Les sous-répertoires de **`src`** doivent refléter votre décomposition en paquetages.

3 Annexe

Le code complet (et commenté) est disponible sur Moodle.

```
package film;

public interface Film {
  int hauteur(); // hauteur (en nbre de caractères) des images du film
  int largeur(); // largeur (en nbre de caractères) des images du film
  boolean suivante(char[][] écran); // obtenir l'image suivante (si elle existe)
  void rembobiner(); // rembobiner le film
}
```

```java
package film;

public class Films {
  public static void projeter(Film f) { ... } // projection
  public static void sauvegarder(Film f, String nom)
      throws FileNotFoundException { ... } // sauvegarde au format de cm-player
}
```

```java
package exemple;

import java.io.FileNotFoundException;
import film.*;

/**
 * Un exemple basique d'implémentation de l'interface Film.
 */
public class LaDiagonaleDuFou implements Film {
  private int num = 0;
  private static final int NB_IMAGES = 100;

  @Override
  public int hauteur() {
    return 10;
  }

  @Override
  public int largeur() {
    return hauteur(); // ce sera un carré
  }

  @Override
  public boolean suivante(char[][] écran) {
    if (num == NB_IMAGES)
      return false;
    // un 'a' se balade sur la diagonale
    écran[num % hauteur()][num % hauteur()] = 'a';
    ++num;
    return true;
  }

  @Override
  public void rembobiner() {
    num = 0;
  }

  /**
   * La projection (puis la sauvegarde) d'un tel film.
   */
  public static void main(String[] args) {
    Film film = new LaDiagonaleDuFou();
    Films.projeter(film);
    film.rembobiner();
    try {
      Films.sauvegarder(film, "fou.txt");
    } catch (FileNotFoundException e) {
      System.err.println("Le fichier 'fou.txt' n'a pas pu être créé.");
    }
  }
}
```