Projects/Devoirs

Notes:

- Pour GUI, vous pouvez utiliser AWT/Swing ou:
  - http://leepoint.net/notes-java/GUI/misc/80gui-generator.html
  - http://www.eclipse.org/swt/
- Vous devez présenter le code source (commenté) et une documentation succincte avec la description de la fonctionnalité de l'application.
- Chaque auteur soulignera la contribution personnelle.
- Le projet remplacera **la vérification finale** de TP et le dernier devoir (de la première semaine du janvier), mais seulement les étudiants qui ont obtenu des notes **de 7(6.5) à 10 au tous les tests** peuvent choisir cette forme de verification.
- Le dernier devoir est obligatoire: il este 10% de la note du TP
- Si le projet est très bon, vous pouvez obtenir un bonus de 1 point à la note finale du TP.
- Date limite pour presenter le projet/dernier devoir: première semaine du janvier
- **Thèmes du projet seront discutés en classe.**

Propositions de themes pour le dernier devoir:

1. Mini-éditeur Java (pour 2 étudiants)

Développez une application pour «simuler» un environnement de travail en Java. L'application doit permettre:

- sélection de fichiers avec l'extension “java”
- affichez le fichier sélectionné avec la possibilité d'apporter des modifications
- enregistrement des modifications apportées
- compiler le fichier et affichage des erreurs
- vous devez avoir un menu aussi

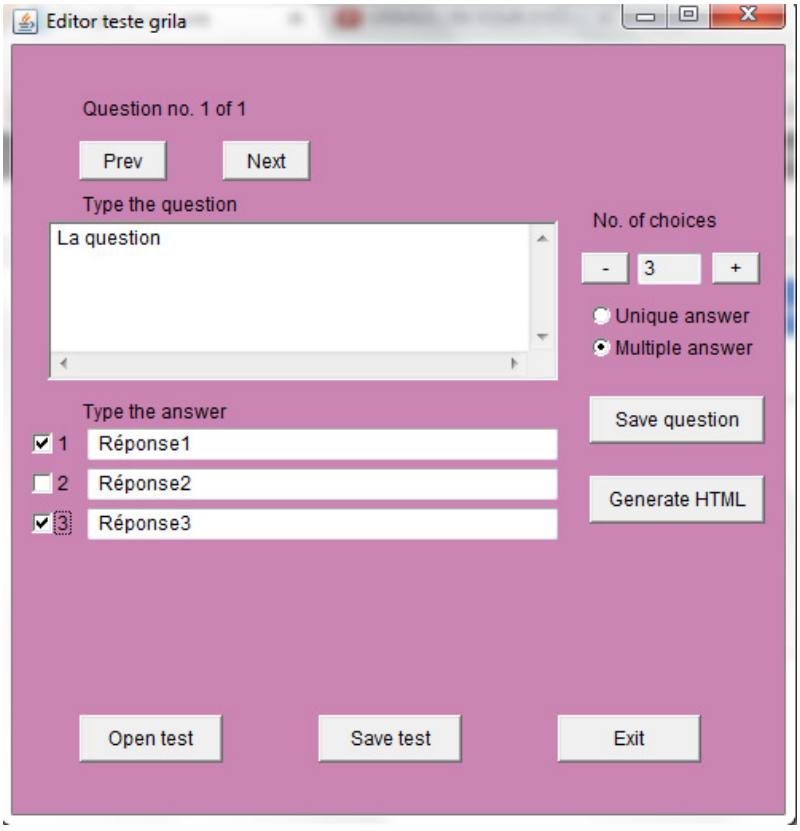
Tip: vour pouvez utiliser la classe com.sun.tools.javac.Main du paquet JAVA_HOME/lib/tools.jar pour faire la compilation.

2. Mini-éditeur de tests à choix multiples (pour 2 étudiants)

Développez un mini- éditeur de tests à choix multiples avec les caractéristiques suivantes:

- rédaction des questions en précisant les informations suivantes: la question, le nombre de réponses possibles, réponse simple / réponses à choix multiples, la bonne réponse/les bonnes réponses
- Sauvegarder / charger des testes: l'operation de sauvegarder est faite en utilisant le format de sérialisation binaire avec la mise en œuvre d'un mécanisme de chiffrement pour les correctes réponses.
- Génération d'un document HTML avec le contenu du test.

E.g.: Pour la première question, nous avons le text de la question ("La question") et 3 réponses possibles (la question est avec réponses à choix multiples); seulement la première réponse et la troisième réponse sont correctes.



3. Vidéo Club (pour 2 étudiants)

Un vidéo club qui propose des films soit sur support DVD soit sur support cassette VHS veut informatiser son système de gestion des prêts. Après une première analyse une ébauche

d'application Java a été élaborée, le diagramme de classes qui a été conçu vous est fourni en annexe 1.

Les classes suivantes sont définies :

La classe **Réalisateur** qui représente un réalisateur de films identifié par son nom et prénom (on suppose qu'il n'existe pas deux réalisateurs de même nom et prénom)

La classe **Film** qui représente un film, identifié par son titre et son réalisateur (il peut exister deux films de même titre mais alors les réalisateurs sont différents). Si le film est une nouveauté son tarif de location sera plus élevé.

La classe **Article** qui représente les différents articles que le magasin propose à la location (DVD et cassettes VHS).

Chaque article est identifié par un numéro d'inventaire unique.

La classe Article possède deux sous classes :

Dvd qui représente les articles de type DVD. Un Dvd possède les attributs suivants :

• l'indication si le DVD contient des bonus ou non (booléen)

• le nombre de disques (entier)

Vhs qui représente les articles de type cassette VHS. Une cassette Vhs possède l'attribut suivant :

• le type de codage vidéo (PAL, SECAM, NTSC). Les valeurs possibles pour ce codage sont définies par l'énumération TypeVHS.

La classe **Adherent** qui représente un adhérent du vidéo club. Chaque adhérent possède les attributs suivants :

• un numéro d'adhérent qui l'identifie de manière unique,

• un nom

• un prénom

• une adresse

Les associations (relations) qui lient ces différentes classes sont les suivantes :

• à un réalisateur sont associés les films qu'il a réalisés. La cardinalité de cette association est 0..*, ce qui signifie qu'un réalisateur peut exister dans le système sans que des films lui soient forcément associés et qu'il n'y a pas de limite au nombre de films qui peuvent lui être associés.

• à un film est associé un réalisateur et un seul.

• à un film sont associés les articles qui lui correspondent. La cardinalité de cette association est 0..*, ce qui signifie qu'un film peut exister dans le système sans que des articles lui soient forcément associés, et qu'il n'y a pas de limite au nombre d'articles qui peuvent lui être associés.

• à un article est associé un film et un seul.

• à un article peut être associé un adhérent du vidéo club. Cette association indique que l'article a été emprunté par cet adhérent. La cardinalité de cette association est 0..1. L'article est disponible si il n'est pas associé à un emprunteur, et un article ne peut avoir au plus qu'un emprunteur.

- à un adhérent sont associés les articles qu'il a empruntés. La cardinalité de cette association est 0..*, ce qui signifie que (pour le moment) le système n'impose pas de limite au nombre d'articles qu'un adhérent peut emprunter.

Pierre, un programmeur java, a commencé le développement de cette application, mais il est tombé malade et n'a pu achever le codage. On vous demande de reprendre son travail.

Question 1 : Ecrivez le code des classes **Articles** et **Dvd**. Les méthodes et constructeurs qui apparaissent sur le diagramme de classes sont les suivants :

Le constructeur **Article(Film f)** crée un nouvel article pour le film *f*. Le numéro d'inventaire de l'article est fixé automatiquement en appliquant la politique suivante: les articles sont numérotés de 1 à n, n étant le nombre d'articles qui ont été créés. Lorsque un nouvel article est créé, le numéro qui lui est attribué est n+1.

Les accesseurs **getNumeroInventaire()** et **getFilm()**retournent respectivement le numéro d'inventaire de l'article et le film associé.

L'accesseur **getEmprunteur()** retourne l'adhérent du vidéo club qui a emprunté le film, ou *null* si le film n'a pas été emprunté.

La méthode **void setEmprunteur(Adherent emprunteur)** permet de définir l'emprunteur de l'article. On passera la valeur *null* à cette méthode lorsque l'article est rendu au vidéo club.

La méthode **String getDescription(boolean complete)** retourne un chaîne de caractères décrivant l'article. Si le paramètre complete vaut true, la forme de cette chaîne est la suivante :

Numéro d'inventaire : 12345

Titre du film : Million Dollar Baby

Réalisateur : Clint Eastwood

Nouveauté : non

Disponible *ou* Emprunté par le membre n° : 14

Si le paramètre complete vaut true, le titre du film et le nom du réalisateur ne sont pas inclus dans la chaîne retournée.

La méthode **double tarifLocation()** retourne le tarif de location de l'article. Le tarif de location dépend de la nature de l'article et si il s'agit ou non d'une nouveauté. Pour une cassette VHS, le tarif est de 1,50 € majoré de 0,50 € si le film est une nouveauté. Pour un DVD, le tarif de base est de 2 €. Ce tarif est majoré de 0,50 € si le DVD comporte des bonus ou si il y a plusieurs disques et de 0,75 € si il s'agit d'une nouveauté.

Le constructeur **Dvd(Film f, int nbDisques, boolean bonus)** crée un nouvel article de type DVD. *f* est le film associé à ce Dvd, *nbdisques* le nombre de disques que contient le coffret, *bonus* indique si le Dvd contient des bonus (*true*) ou non (*false*).

Les accesseurs **hasBonus()** et **getNbDisques()**retournent respectivement les valeurs des attributs bonus et nbDisques.

La méthode **String toString()** de la classe Dvd retourne un chaîne de caractères décrivant le Dvd. La forme de cette chaîne est la suivante :

DVD (1 disque avec bonus)

Numéro d'inventaire : 12345

Titre du film : Million Dollar Baby

Réalisateur : Clint Eastwood

Nouveauté : non

Disponible *ou* Emprunté par le membre n° : 14

La méthode **double tarifLocation()** de la classe Dvd retourne le tarif de location du Dvd.

Question 2 : Ecrire le code de la classe Film (voir annexe 2).

Question 3 : Pour gérer l'ensemble des articles du vidéo club il est décidé d'écrire une classe **Inventaire** qui propose les méthodes suivantes :

Article findArticle(int noInventaire) qui retourne l'article de numéro d'inventaire *noInventaire* si le vidéo club possède un article avec ce numéro, *null* si il n'existe pas d'article à ce numéro.

int nouveauDvd(Film f, int nbDisques, boolean bonus) qui permet de créer un nouvel article de type Dvd et de l'ajouter à l'inventaire. Cette méthode retourne le numéro d'inventaire de l'article crée.

int nouveauVhs(Film f, TypeVhs type) qui permet de créer un nouvel article de type Dvd et de l'ajouter à l'inventaire. Cette méthode retourne le numéro d'inventaire de l'article crée.

void supprimerArticle(int noInventaire) qui permet de retirer de l'inventaire l'article de numéro *noInventaire*. Si il n'existe pas d'article à ce numéro la méthode est sans effet. Si l'article a été emprunté, il ne peut être retiré de l'inventaire. Dans ce cas une exception de type ArticleException avec le message *« article emprunté »* est lancée. La classe ArticleException est une classe d'exceptions qui a été développée pour ce projet. Sa javadoc vous est fournie en annexe 3.

a) Quelle structure de données proposez-vous d'utiliser pour stocker les articles de l'inventaire ? Justifiez votre réponse.

b) Quelle modification faut-il apporter à la classe Film pour permettre la suppression d'un article ? Ecrivez le code à ajouter à la classe Film.

c) Ecrivez le code de la classe Inventaire.

d) Ecrivez un petit programme qui permet de tester la classe Inventaire. Ce programme exécute les traitements

suivants :

Creation d'un inventaire (vide au départ)

Création d'un objet réalisateur *r*

Création d'un objet film f associé à *r*

Ajout à l'inventaire d'un premier dvd associé à *f*, *no1* son numéro d'inventaire

Ajout à l'inventaire d'un second dvd asocié à *f*, *no2* son numéro d'inventaire

Ajout à l'inventaire d'une cassette vhs associée à *f*, *no3* son numéro d'inventaire

Création d'un adhérent *a*.

Emprunt du dvd *no1* par *a*

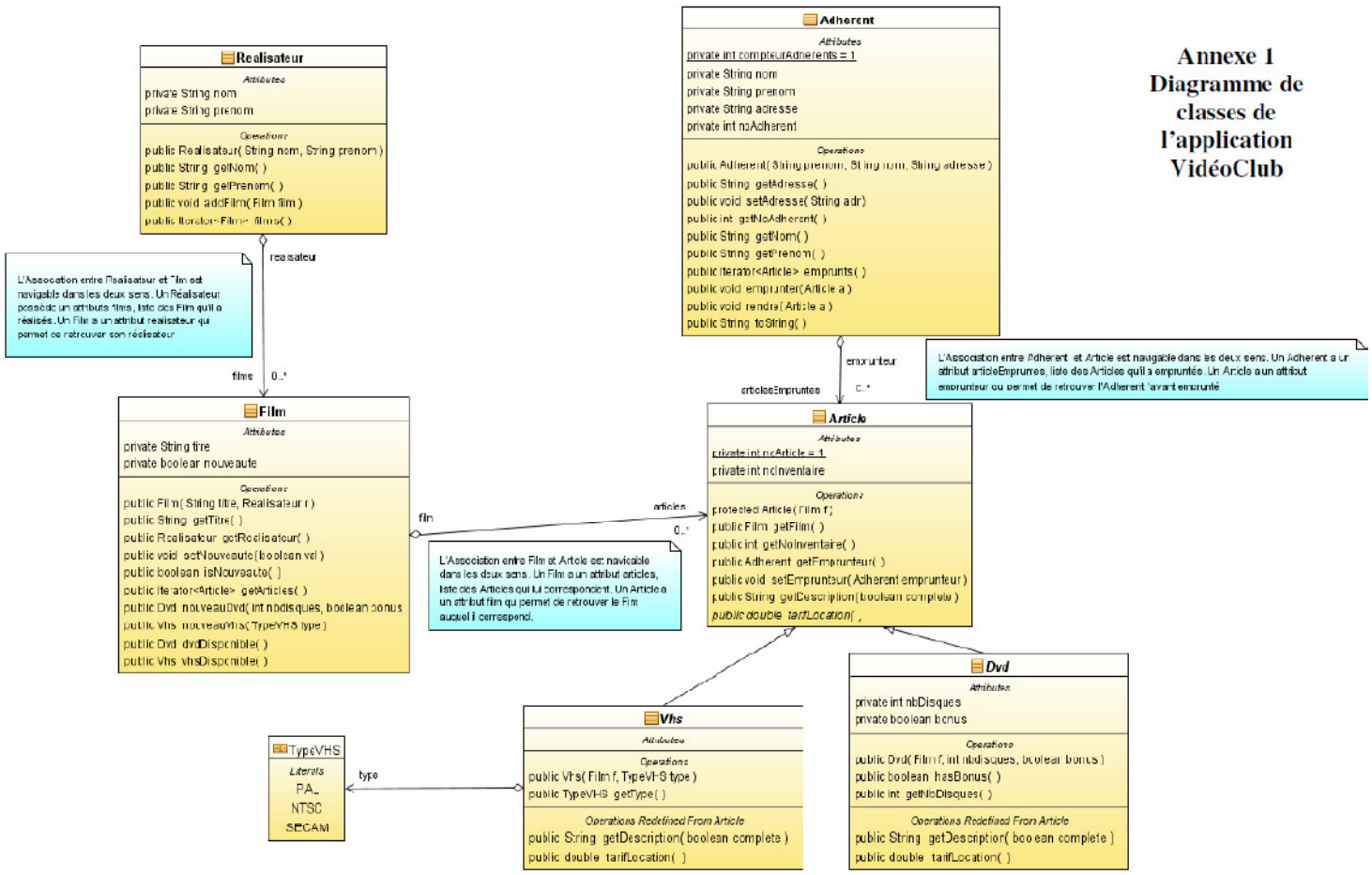
Affichage sur la console de la liste des articles associés à *f*

Suppression du second dvd (*no2*) de l'inventaire

Suppression premier dvd (*no1*) de l'inventaire. Cette suppression doit échouer car le dvd est emprunté.

Affichage sur la console de la liste des articles associés à *f*

Question4 : Mettez en œuvre une GUI appropriée.



Annexe 2 : spécification des constructeurs et méthodes de la classe `Film`

```
/**
 * Création d'un film
 * @param titre le titre du film
 * @param r le réalsateur du film
 */
public Film (String titre, Realisateur r)

/**
 * @return le titre du film
 */
public String getTitre ()

/**
 * @return le réalisateur du film
 */
public Realisateur getRealisateur ()

/**
 * fixe la valeur de l'attribut nouveauté
 * @param val true si le film est une nouveauté, false sinon
 */
public void setNouveaute(boolean val)

/**
 * @return true si le film est une nouveauté
 */
public boolean isNouveaute()

/**
 * création d'un nouveau dvd et ajout de celui-ci à la liste des
 * articles de ce film
 * @param nbdisques nombre de disques du DVD
 * @param bonus
 * @return l'objet Dvd crée
 */
public Dvd nouveauDvd (int nbdisques, boolean bonus)

/**
 * création d'une nouvelle cassette vhs et ajout de celle-ci à la liste
 * des articles de ce film
 * @param type le type de codage vidéo
 * @return l'objet vhs crée
 */
public Vhs nouveauVhs (TypeVHS type)
```

```
/**
 * nombre d'articles pour ce film
 * @return le nombre d'articles
 */
public int getNbArticles()

/**
 * recherche parmi les articles pour ce film un Dvd disponible
 * @return un dvd disponible (non emprunté) ou null si aucun
 */
public Dvd dvdDisponible ()

/**
 * recherche parmi les articles pour ce film un Dvd disponible
 * @return un dvd disponible (non emprunté) ou null si aucun
 */
public Vhs vhsDisponible ()

@Override
public String toString()

/**
 * renvoie un iterateur sur les articles associés à ce Film
 * @return iterateur sur les articles du film
 */
public Iterator<Article> iterator()
```