ENSEEIHT 2ième année Département Informatique. Année 2016-2017

Nom : ……..

Intergiciels

Examen (session1)

Daniel Hagimont

Durée: 2 heures, documents autorisés

*Lire l'ensemble des énoncés avant de commencer à répondre. La **clarté**, la **précision** et la **concision** des réponses, ainsi que leur **présentation matérielle**, seront des éléments importants d'appréciation. Donnez essentiellement les programmes Java demandés. Dans vos programmes, vous n'avez pas à programmer les imports et le traitement des exceptions.*

PROBLÈME I (RMI) (6 points)

On veut implanter un service de streaming de film. Ce service est composé de deux parties : une vidéothèque (question RMI) et un serveur de streaming (question socket).

Dans une première phase, on veut implanter la vidéothèque enregistrant pour un film l'adresse (host, port) de son serveur de streaming.
Un client (un programme de visionnage de film ou un programme voulant enregistrer un film) peut accéder à distance avec RMI à un serveur implantant une interface Videotheque fournissant deux méthodes :

```
void enregistrer(Film f) ;
Film rechercher(String titre) ;
```

Film est une classe implantant une structure de donnée (passée par copie en paramètre de ces méthodes) contenant des champs :

- String titre ;
- String host ;
- int port ;

Ces informations enregistrées dans le serveur indiquent que le film identifié par *titre* est accessible en streaming sur la machine *host* et le port *port*.

Pour l'instant, on ne s'intéresse pas à la partie streaming du film qui est traitée dans la question suivante.
On implante juste la partie vidéothèque.
Vous devez programmer (en RMI) l'interface Videotheque, la classe VideothequeImpl, l'interface Film et la classe FilmImpl. L'implantation de la classe VideothequeImpl reposera simplement sur une table HashMap pour l'enregistrement des films.
On donnera également un programme simple d'un client (une classe Client) réalisant deux appels aux deux méthodes ci-dessus.

PROBLÈME II (Socket) (6 points)

Nous voulons maintenant implanter la partie streaming avec des sockets.

Une classe StreamingServer s'utilise de la façon suivante :
java StreamingServer UnTitre UnFichier UnPort

Les paramètres donnent respectivement un titre de film, un nom de fichier local à la machine contenant le film, et un port de connexion à la machine pour accéder en streaming au contenu du film. Un StreamingServer ne sert donc qu'un seul film.
La classe StreamingServer doit enregistrer le film dans la vidéothèque de la première question, puis être à l'écoute de connexions provenant des clients. StreamingServer est multi-threadée.

Lors d'une connexion, StreamingServer envoie directement le contenu du film sur la connexion.
Pour accéder au fichier contenant le film, la classe StreamingServer utilisera simplement un FileInputStream.
Pour la programmation des entrées/sorties (que ce soit pour un fichier ou un socket), on utilisera :

InputStream

public int read(byte[] b); // bloquante, retourne le nombre de bytes lus, -1 si end of stream

OutputStream

public void write(byte[] b, int off, int len); // écrit les len bytes à la position off

Donner une implantation de la classe StreamingServer.
Donner l'implantation d'une classe StreamingClient qui prend en paramètre un titre de film, consulte la vidéothèque et se connecte au StreamingServer. Pour chaque buffer reçu, StreamingClient appellera une méthode statique : *VLC.show(buffer);* qui provoque l'affichage du film.

PROBLEME III (JEE)

Les réponses doivent être rédigées dans les cadres sur cette feuille. Être concis pour tenir dans le cadre. Toute réponse hors de ces cadres ne sera pas prise en compte.

Question 1 (2 points)
Expliquer en quelques mots l'utilité du paramètre d'annotation fetch=FetchType.EAGER

.....
.....
.....

Question 2 (2 points)
Si en JPA, on dispose de deux Entity Person et Address. Je veux programmer que tchana et hagi sont tous les 2 résidents à l'enseeiht.

```
@Entity
public class Person {
    @Id
    String name;
    ...
```

```
@Entity
public class Address {
    @Id
    String place;
    String town;
    @ManyToOne
    Person resident;
    ...
```

Et dans une facade, si j'écris :

```
Person p1 = em.find(Person.class, "tchana");
Person p2 = em.find(Person.class, "hagi");
Address a = em.find(Address.class, "enseeiht");
a.setResident(p1);
a.setResident(p2);
```

Avec une relation **unidirectionelle**, ce code est il correct ? Proposez une modification si nécessaire.
Avec une relation **bidirectionelle**, ce code est il correct ? Proposez une modification si nécessaire.
NB : si une modification est nécessaire, il y a plusieurs solutions ...

unidirectionnelle

bidirectionnelle

…… …… …… …… …… …… …… …… …… ……..	…… …… …… …… …… …… …… …… …… ……..

Question 3 (2 points)
Avec les Entity initiaux de la question 2 (pas ceux que vous avez proposés, ceux dans l'encadré), j'ai une servlet qui a récupéré la saisie d'une adresse pour une personne et qui appelle une facade qui propose la méthode suivante :

```
public void setAddress(String person_name, Address a) {        // méthode de la facade qui reçoit un
    Person p = em.find(Person.class, person_name);              // objet Address de la servlet
    a.setResident(p);
}
```

Quels problèmes peut poser la méthode ainsi codée ? Expliquez et proposez une correction.

.....
.....
.....

Question 4 (2 points)
Toujours avec les Entity initiaux de la question 2, j'ai une servlet qui appelle une facade qui propose la méthode d'interface : public Collection<Person> listPersons();
puis elle redirige vers une JSP showList.jsp

Donnez la séquence de code de cette Servlet (appel à la facade et redirection vers la JSP), ainsi qu'une implantation de la JSP (elle affiche juste une liste de noms).

.....
.....
……
……
……
……
……
……
……
……
……
……
……