TD1 : Retour sur java

On souhaite écrire dans ce TD la gestion d'une médiathèque. La première partie consistera en l'écriture des différentes classes liées aux médias. La seconde partie portera sur la gestion de la médiathèque elle-même.

Les classes liées aux médias

Dans cette partie nous allons mettre en place un système d'héritage. La classe **Media** sera la classe mère. Les classes **Livre**, **Film** seront les classes filles.

La classe Media

La classe **Media** est constituée :

- Données d'instance (toutes définies privées) :
  - Le **titre** du média (un objet de type String)
  - La **cote** du média (un objet de type Stringbuffer)
  - La **note** attribuée au média (un entier)
- Données de classe (définies privée) :
  - Du **nom** de la médiathèque (un objet String)
- Méthodes :
  - Les trois constructeurs : par défaut, par initialisation et par copie. Les constructeurs ne modifierons pas l'objet **nom** ;
  - Les setters et getters associés aux données d'instance ;
  - Deux méthodes de classe permettant de modifier l'objet **nom** ;
  - Le masquage des méthodes clone, equals et toString.

Question 1

Donnez la structure mémoire d'un objet de type **Media**, ainsi que les différents accès possibles aux différents éléments de cet objet ou de la classe elle-même.

Question 2

Ecrivez la classe **Media**

Question 3

Considérons le getter de l'objet StringBuffer cote . Si ce getter a été écrit comme suit :

```
public String getCote() {
        return new this.cote;
    }
```

Que donnera le code suivant :

```
Media m1 = new Media("Kill Bill vol 1","2BXX5",5);
System.out.println(m1);
m1.getCote().reverse();
System.out.println(m1);
```

Comment écrire le getter afin d'éviter cela?

Question 4

Donnez un programme d'exemple de la classe **livre**

Question 5

Donnez toutes les déclarations possibles avec les classes **Media** et **Livre**

Question 6

Rappelez les règles de résolution des méthodes en Java

Question 7

Parmi les instructions suivantes, lesquelles sont valides.

```
Media m1 = new Livre("Le livre de la jungle","DFTY4HG",5,"R. Kippling","20
81263246");
System.out.println(m1);
m1.setCote("SD77DS");
System.out.println(m1.getAuteur());
System.out.println(((Livre)m1).getAuteur());
```

Question 8

la classe **Film** hérite de la classe **Media**.

Rappel le String **titre** et le Stringbuffer **code** ne seront plus **private** mais **protected**.

La classe **Film** hérite donc de la classe **Media**, et l'augmente

- Des données d'instance (positionnés private) :
  - le **réalisateur** (un objet de type String) ;
  - l'**année** (un entier) ;
- Des méthodes :
  - les trois constructeurs : par défaut, par initialisation et par copie ;
  - les setters et getters associés aux données d'instance ;
  - le masquage des méthodes clone, equals et toString.

Ecrivez la classe **Film.**

Question 9

Ecrivez un programme d'exemple exploitant la classe **Film.**

Question 10

Une médiathèque est un ensemble de médias (**Livre** ou **Film**).

La classe **Mediatheque** est constituée de : - pour les données - le nom du propriétaire (un objet de type String) - un objet de type **Vector** de **Media** - pour les méthodes - 2 constructeurs (un par défaut et un par copie) - la méthode **void add(Media)** - la méthode toString

Ecrivez la classe **Mediathèque**

Question 11

Donnez un exemple d'utilisation de la classe **Mediatheque.**

TD2 : Modélisation

Une application de chat

On souhaite mettre en place une application de chat, basée sur un modèle client-serveur. Dans notre application, l'ensemble des clients est connecté au serveur qui assurera le point de concentration et la distribution. Seule l'adresse du serveur sera connue des clients.

À chaque client sera associé un identifiant, choisi par l'utilisateur, les identifiants seront stockés sur le serveur.

Le chat devra permettre de diffuser un message à l'ensemble des clients, mais aussi de faire des messages privés à destination d'un seul client.

Question 1 Proposez une architecture générale. Vous étudierez notamment les différentes solutions liées au transport (connecté et non connecté).

Question 2 Proposez un ensemble de fonctionnalités du client et du serveur (choix de l'identifiant, envoi de messages, envoi de messages privés, récupération de la liste des clients connectés, *blacklist*...) ainsi que la description des échanges associés.

> **Remarque** vous n'oublierez pas de décrire les échanges d'erreurs associés.

Question 3 Reprenez l'étude avec un modèle d'exécution par passage de messages.

Question 4 Lister les erreurs possibles et envisagez des solutions de gestion de ces erreurs.

Question 5 En envisageant une implémentation en java, et sans écrire le code exact du serveur, proposez une architecture logicielle pour le serveur. Vous décrirez les différents objets ainsi que de manière informelle les algorithmes et opérations réalisées.

TD3 : Boîte à outils

Nous allons dans ce TD nous focaliser sur le développement des applications à l'aide de Sockets.

Une messagerie interne

Modélisation client / serveur

On souhaite développer une messagerie interne, permettant à des personnes d'une même organisation d'échanger des messages. Cette application doit permettre l'envoi, la réception, l'archivage et l'effacement de messages.

Question 1 Définissez la structure d'un message.

Question 2 Définissez les différents acteurs en précisant leur rôle et leurs charges.

Question 3 Proposez l'architecture des échanges et étudiez les différentes requêtes et réponses possibles.

Développement en mode connecté

En s'appuyant sur la modélisation précédente, on souhaite maintenant proposer une implémentation java exploitant des échanges TCP.

Question 4 Décrivez l'architecture de la messagerie dans le cadre du client/serveur à l'aide de sockets TCP.

Question 5 Proposez l'implémentation :

- D'un serveur de messagerie ;
- D'un client d'émission de messages ;
- D'un client de lecture de messages.

Développement en mode non connecté

En s'appuyant sur la modélisation précédente, on souhaite maintenant proposer une implémentation java exploitant des échanges UDP.

Question 6 Décrivez l'architecture de la messagerie dans le cadre du client/ serveur à l'aide de datagrammes UDP.

Question 7 Proposez l'implémentation :

- D'un serveur de messagerie ;
- D'un client d'émission de messages ;
- D'un client de lecture de messages.

TD4 : JSON

Nous désirons réaliser une application permettant de gérer l'entreposage et le transport de conteneurs de 20 pieds. Un conteneur est caractérisé par un numéro d'identification unique (une suite de lettres et de chiffres), le propriétaire du contenu (une entreprise possédant un nom et une adresse), le poids total et la feuille de route. Une feuille de route permet de connaître les différentes étapes du trajet du conteneur (adresse et propriétaire de l'entrepôt, trajets effectués).

Il existe des conteneurs de différents types :

- Les conteneurs simples (les plus courants) contiennent une liste de marchandises déclarées.
- Les conteneurs réfrigérés sont équipés d'un système de refroidissement caractérisé par sa température et son autonomie (pour simplifier, il suffit de mémoriser la durée maximale de fonctionnement ainsi que la date de mise en route du système). Ils contiennent aussi une liste de marchandises déclarées.
- Les conteneurs liquides permettent de transporter du liquide (produit chimique, gaz). Ils sont caractérisés par le type de liquide et son volume.
- Les conteneurs pour vrac permettent de transporter des matières en vrac (comme des graines). Ils contiennent donc un type de contenant.

Les conteneurs sont stockés dans des entrepôts. Ces derniers sont caractérisés par un nom, un nombre de rangées (chacune caractérisée par une lettre) possédant différentes places (numérotées). Chaque place correspond à une pile de conteneurs (la hauteur de la pile dépend de l'entrepôt, ce qui correspond généralement à la hauteur maximale qu'un engin de levage peut atteindre, en général de 6 à 8).

L'application répartie est constituée des différentes applications qui sont exécutées sur les différents sites d'entreposage des conteneurs, appartenant à des entreprises différentes. Chaque application est donc indépendante et conserve ses informations localement. Lorsqu'un conteneur est déplacé, les applications peuvent communiquer alors les informations. De même, il est possible d'interroger les différents sites pour obtenir des informations sur les conteneurs ; dans ce cas, une authentification est nécessaire (toutes les entreprises n'ont pas nécessairement un accès vers toutes les autres entreprises). Toutes les données stockées et échangées sont au format JSON.

Données

Question 1. Les informations étant stockées dans des fichiers au format JSON, proposez une architecture de fichiers. Pour chacun, expliquez l'utilité.

Question 2. Proposez un format JSON permettant de représenter les conteneurs.

Question 3. Proposez maintenant une structure générale en JSON pour tous les fichiers.

Modélisation

Question 4. Proposez une architecture de l'application basée sur le modèle client-serveur et sur le modèle par passage de messages. Vous détaillerez le rôle de chaque composant du système.

Question 5. Déterminez les différents échanges possibles entre les différents sites (ou autres composants comme vu dans la question précédente). Vous donnerez le format JSON de chaque message.

Programmation

Question 6. Rappelez ou proposez un diagramme de classes type UML permettant de représenter les entrepôts et les conteneurs. Nous souhaitons également faire apparaître les différents types de message.

Question 7. Proposez une solution permettant de sérialiser automatiquement les objets en présence au format JSON.

Question 8. Ecrivez la méthode `toJSON` d'un des objets.

Question 9. Proposez une solution permettant, à partir d'un flux au format JSON, la création automatique des objets correspondant.

INFO0503 QCM

Nom :	**Prénom :**	**Numéro étudiant :**

Aucun document autorisé.
Vous remettrez dans votre copie le QCM.

Les questions peuvent avoir plusieurs réponses possibles. Une question est validée lorsque l'ensemble des bonnes réponses est coché et qu'aucune mauvaise réponse n'est cochée. Une bonne réponse rapporte 0,5 point, une mauvaise retire 0,25 point, l'absence de réponse n'est pas pénalisée.

1) Dans le modèle client / serveur :
   - ☐ Une réponse est attendue à chaque requête
   - ☐ Un message ne peut pas être reçu sans requête préalable
   - ☐ Ne peut être appliqué qu'avec des sockets en mode non connecté
   - ☐ Ne peut être appliqué avec une application *Java* RMI
2) En Java RMI, les paramètres de l'appel distant doivent …
   - ☐ … être sérialisables
   - ☐ … être instanciés obligatoirement par le client
   - ☐ … être instanciés obligatoirement par le serveur
3) En *Java* RMI, l'interface permet…
   - ☐ … d'indiquer au client, les méthodes utilisables
   - ☐ … donne des informations sur le transfert de l'objet
   - ☐ … est créée à l'exécution sur le client
   - ☐ … est présente sur le serveur et le client
4) Une application client/serveur peut être réalisée en mode …
   - ☐ … connecté persistant
   - ☐ … connecté non persistant
   - ☐ … déconnecté persistant
   - ☐ … déconnecté non persistant
5) Le protocole UDP peut assurer …
   - ☐ … le contrôle de présence
   - ☐ … la vérification de la cohérence des données JSON
   - ☐ … un transfert "léger" des données
   - ☐ … la gestion de la sérialisation

6) Pour échanger du JSON en *Java*…
   - ☐ … on échange un objet sérialisé
   - ☐ … on échange uniquement du texte
7) Parmi les classes suivantes, lesquels offrent des objets sérialisables
   - ☐ String
   - ☐ Thread
   - ☐ Calendar
   - ☐ Socket
8) Le protocole UDP est caractérisé par …
   - ☐ … l'absence du contrôle de présence
   - ☐ … l'utilisation de datagrammes UDP
   - ☐ … la forte dépendance avec le langage Java
   - ☐ … l'utilisation de chronogrammes UDP
9) Le processus de modélisation comprend …
   - ☐ … la description des acteurs
   - ☐ … la description des échanges
   - ☐ … la représentation du format des données
   - ☐ … l'ensemble du code du programme
10) Le serveur http intégré à l'API Java …
   - ☐ … permet de mettre en place des applications WEB
   - ☐ … assure de manière autonome la gestion des en-têtes HTTP
   - ☐ … permet la définition de contextes utilisateurs
   - ☐ … assure la connexion à une base de données pour toute requête

DS 2017 – 2018 (2 heures)

Ce sujet comporte 2 pages.

Aucun document autorisé.

La clarté des réponses sera prise en compte lors de la correction et de la notation.

Vous remettrez dans votre copie le QCM, en précisant votre nom.

Une application de gestion des accès

On souhaite développer une application permettant la gestion d'une centrale d'accès. Dans cette application nous avons différents acteurs : les points d'accès qui assurent l'ouverture et la fermeture des portes ; la centrale de gestion des accès qui assure la gestion des autorisations d'ouvertures ; le gestionnaire de l'historique des accès qui mémorise l'ensemble des accès (heure, numéro de badge, utilisateur).

Remarques :

1. Les badges sont activés ou non par l'administrateur, seuls les utilisateurs ayant des badges activés peuvent entrer ;
2. Le fonctionnement du lecteur de badge peut être considéré comme le fonctionnement d'un *ServerSocket*, la présentation d'un badge provoquera le passage du *accept*, dans ce cas, ce n'est pas une connexion *Socket* qui sera retournée mais le numéro du badge, qui dans notre cas est un entier.

Chaque utilisateur détient un badge qu'il présente au point d'accès. Après un échange avec la centrale de gestion des accès, l'autorisation d'ouverture de la porte est obtenue, ou non obtenue. Dans tous les cas, la centrale de gestion des accès transmet au gestionnaire d'historique la demande d'accès, ce dernier la mémorise.

Modélisation

1) Proposez une modélisation d'un point d'accès. Vous expliquerez le fonctionnement général et donnerez le pseudo-code associé. Vous proposerez le prototype des fonctions nécessaires au fonctionnement d'un point d'accès.
2) Proposez une modélisation générale du système, vous préciserez à ce stade les acteurs, ainsi que les flux de données. Vous ne décrirez pas encore les messages, ni les traitements.
3) Proposez la modélisation des échanges de messages, ainsi que la description des traitements effectués à la réception de ceux-ci.
4) Expliquez les différences d'implémentation, ainsi que les différents moyens à mettre en œuvre, entre un développement de cette application dans un mode client/serveur et un mode de communication par messages.

Implémentation et représentation des données

1) Expliquez quel(s) protocole(s) peut/peuvent être utilisé(s) entre chaque entité du système, en fonction de ce qui a été présenté en cours / travaux dirigés / travaux pratiques. Expliquez pour chaque choix ce que cela implique au niveau du développement.
2) Proposez un format JSON pour l'ensemble des messages du système.
3) Proposez un format JSON pour l'ensemble des données enregistrées sur chaque acteur système.