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) Une application client/serveur peut être réalisée en mode …
   - ☒ … connecté persistant
   - ☑ … connecté non persistant
   - ☐ … déconnecté persistant
   - ☑ … déconnecté non persistant
3) Le protocole UDP peut assurer …
   - ☐ … le contrôle de présence
   - ☐ … la vérification de la cohérence des données JSON
   - ☑… la gestion de la sérialisation
4) Pour échanger du JSON en *Java*…
   - ☐ … on échange un objet sérialisé
   - ☑ … on échange uniquement du texte
5) Parmi les classes suivantes, lesquels offrent des objets sérialisables
   - ☑ String
   - ☐ Thread
   - ☑ Calendar
   - ☐ Socket
6) 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

7) 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
8) 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.

Indications de correction :

Un point d'accès est composé :

- D'un mécanisme de lecture des badges
- D'un mécanisme de communication avec la centrale de gestion des accès
- D'un mécanisme de commande de l'ouverture de la porte

Le principe général de fonctionnement est le suivant

1. Le point d'accès est en attente de lecture d'un badge
2. A la lecture d'un badge, il récupère l'identifiant du badge
3. Émet la valeur de l'identifiant à la plate-forme de gestion
4. Récupère l'autorisation ou non d'ouverture
5. Effectue l'action en fonction de la valeur récupérée

Les fonctions nécessaires sont

- idbadge accept() : fonction attendant la présentation d'un badge sur le dispositif, et renvoyant sa valeur;
- send(idbadge,addr) fonction envoyant l'id d'un badge à la centrale dont l'adresse est addr
- booléen receive() : fonction d'attente de réception d'un booléen
- void open() : fonction d'ouverture de la porte

Pseudo code

```
while(1) {
      id = accept()
      send(id,adresse_centrale)
      authorisation = receive()
      if(autorisation == true){
            open()
      }
}
```

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.

Indications de correction :

Les acteurs

Le lecteur de badge assure :

- La lecture des id contenus dans les badges
- Les échanges avec le gestionnaire
- L'ouverture ou non de la porte

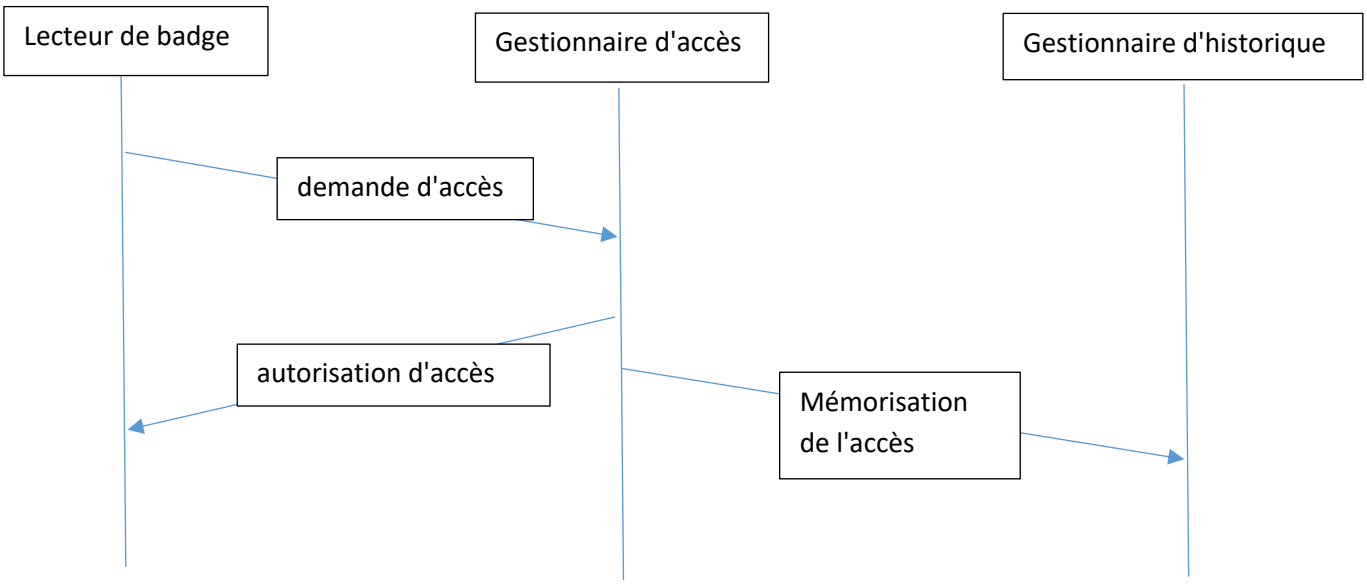
  Le lecteur de badge ne possède pas de mémoire particulière, il n'effectue pas de calculs en interne

La centrale de gestion assure

- La réception des id des badges émis par les lecteurs de badges
- La recherche dans la base des informations de l'id envoyé, la vérification des droits d'accès, le retour de l'autorisation d'ouverture

Le gestionnaire d'historique qui assure

- La mémorisation des accès



Remarque : on peut, dans le cas d'un accès non autorisé, choisir de ne pas renvoyer de réponse

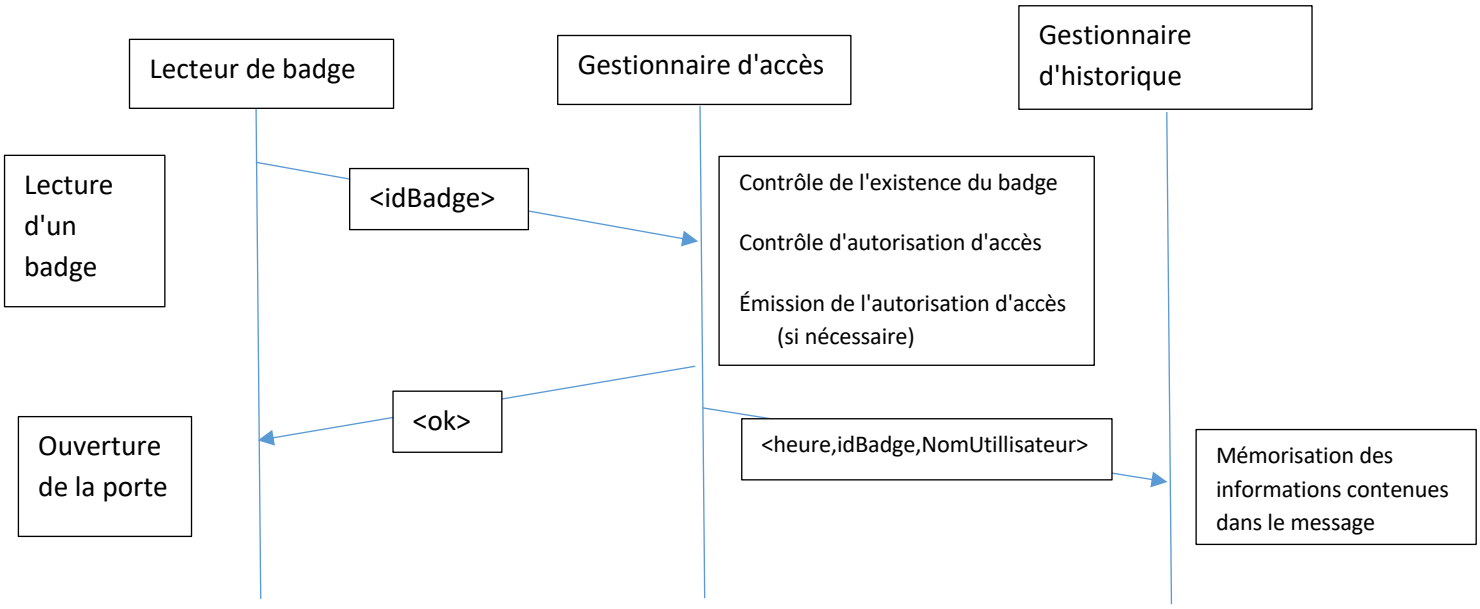
3) Proposez la modélisation des échanges de messages, ainsi que la description des traitements effectués à la réception de ceux-ci.

Indications de correction :

On a trois types de messages :

- Les demandes d'accès : <idBadge>
- L'autorisation d'accès : <ok>
    - on peut considérer qu'en cas d'un d'accès interdit on ne renverra aucun message
- La mémorisation de l'accès : <heure,idBadge,NomUtilisateur>

On peut représenter le système comme suit :



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.

Indications de correction :

Modélisation Client / Serveur

Dans le cadre d'une modélisation client / serveur, on a obligatoirement une requête et une réponse associée. Il est impossible d'avoir une requête sans réponse.

- Dans le cas de l'implémentation en mode connecté, il faut mettre en œuvre un serveur de réception de message, à la fois sur le gestionnaire d'accès et sur le gestionnaire d'historique. Il faut de plus que la communication puisse être établie, depuis le lecteur de badge vers le contrôleur d'accès et depuis le contrôleur d'accès vers le gestionnaire d'historique.
- Dans le cas de l'implémentation en mode non connecté, il faut mettre en œuvre un serveur de réception sur chacun des acteurs. De plus, il faut assurer les communications puissent s'établir de manière bi directionnelles : entre le lecteur de badge et le gestionnaire d'accès, et entre le gestionnaire d'accès et le gestionnaire d'historique.
- Si on considère les échanges entre le lecteur de badge et le contrôleur d'accès, il est nécessaire à chaque requête de demande d'accès de répondre, soit par une réponse d'acceptation d'accès, soit par une réponse de refus d'accès
- Si on considère les échanges entre le contrôleur d'accès et le gestionnaire d'historique, il est indispensable de mettre en place une réponse à la requête de mémorisation. Cette réponse sera un simple acquittement de la requête

Modélisation par messages

Dans le cadre d'une modélisation par échange de message, il n'y a pas d'obligation de réponse. Donc, on choisira de ne pas renvoyer de message en cas de non autorisation d'accès et de ne pas donner de réponse au message d'enregistrement de l'historique.

- Dans le cas de l'implémentation en mode connecté, il faut mettre en œuvre un serveur de réception de message, à la fois sur le gestionnaire d'accès et sur le gestionnaire d'historique. Il faut de plus que la communication puisse être établie, depuis le lecteur de badge vers le contrôleur d'accès et depuis le contrôleur d'accès vers le gestionnaire d'historique.
- Dans le cas de l'implémentation en mode non connecté, il faut mettre en œuvre un serveur de réception sur chacun des acteurs. De plus, il faut assurer les communications puissent s'établir de manière bi directionnelles : entre le lecteur de badge et le gestionnaire d'accès, et uni-directionnelle entre le gestionnaire d'accès et le gestionnaire d'historique.
- Si on considère les échanges entre le lecteur de badge et le contrôleur d'accès, à une demande d'accès peut être soit associé une réponse d'autorisation d'accès, ou aucune réponse.
- Si on considère les échanges entre le contrôleur d'accès et le gestionnaire d'historique, aucune réponse n'est attendue.

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.

Indications de correction :

Les échanges entre les différents acteurs peuvent se faire, soit en "transport brut" (TCP ou UDP) soit en exploitant le protocole http.

Cas du transport "brut"

Dans le cas de la communication brute, on peut mettre en œuvre aussi bien un modèle client / serveur qu'un modèle par passage de messages.

Communications en mode connecté (TCP)

En mode connecté, il faut gérer :

- Sur le gestionnaire d'accès et sur le gestionnaire d'historique un ServerSocket. Le lecteur de badge se connectera par l'intermédiaire d'une connexion de type Socket au gestionnaire d'accès et les échanges se feront par l'intermédiaire de la connexion établie.
- Sur le gestionnaire d'historique un ServerSocket. Le lecteur gestionnaire d'accès se connectera par l'intermédiaire d'une connexion de type Socket au gestionnaire d'historique et les échanges se feront par l'intermédiaire de la connexion établie.

Les échanges se feront en JSON, ils pourront soit être effectué par des échanges de chaînes de caractères de type String, soit par le biais d'une classe message contenant les informations JSON et dans ce cas la sérialisation sera utilisée.

Communications en mode non connecté (UDP)

En mode non connecté il faudra gérer sur chaque acteur pouvant recevoir des messages un DatagramSocket et la méthode receive.

Les échanges se feront en JSON, ils pourront soit être effectué par des échanges de chaînes de caractères de type String, soit par le biais d'une classe message contenant les informations JSON et dans ce cas la sérialisation sera utilisée.

Exploitation du protocole http

Si on utilise le protocole http les échanges se feront forcément selon un mode connecté non persistant. Par contre en exploitant le protocole http, on sera donc forcément dans un modèle client / serveur. Il y aura donc une réponse à chaque message émis.

Il faudra gérer sur le gestionnaire d'accès comme sur le gestionnaire d'historique un serveur http. On pourra utiliser soit une solution de type PHP, soit une solution de type Java.

Solution PHP

Dans le cas d'une solution PHP on devra donc avoir un serveur WEB capable d'exécuter des scripts PHP installés sur le gestionnaire d'accès comme sur le gestionnaire d'historique. Les données JSON transiteront dans la partie de données, et de la requête http, et de la réponse http. Le type MIME de chaque message échangé sera donc application/json. Il sera nécessaire de gérer le type MIME des réponses http dans les scripts PHP.

Solution Java

Dans le cas d'une solution java, on pourra utiliser le serveur http de l'API. On exploitera donc le HttpServer. Ensuite on devra masquer la méthode handle dans le cadre d'un héritage sur chacun des

serveurs. Comme dans le cas précédent, les données JSON transiteront dans la partie de données, et de la requête http, et de la réponse http. Le type MIME de chaque message échangé sera donc application/json . Il sera nécessaire de gérer le type MIME des réponses http dans les méthodes handle.

2) Proposez un format JSON pour l'ensemble des messages du système.

Indications de correction :

L'envoi de l'identifiant du badge (en replaçant XXX par l'identifiant du badge) :

```
{
  "mode" : "idBadge",
  "idBadge" : "XXX"
}
```

La réponse du gestionnaire d'accès :

```
{
  "code" : 1,
  "message" : "autorisation d'accès"
}
```

Chaque erreur peut avoir son propre code d'erreur. Par exemple :

```
{
  "code" : 2,
  "message" : "Identifiant de badge inconnu"
}
```

Pour la mémorisation de l'accès :

```
{
  "date" : "04/11/2018",
  "heure" : "8:00",
  "idBadge" : "XXX",
  "code" : 1
}
```

3) Proposez un format JSON pour l'ensemble des données enregistrées sur chaque acteur système.

Indications de correction :

Lecteur de badge

Aucune donnée.

Gestionnaire d'accès

Le gestionnaire d'accès a besoin de la liste des badges (les identifiants) avec les accès autorisés sous la forme de plages horaires. Un seul fichier peut être suffisant (si le nombre de badges n'est pas trop grand.

```
[
  {
```

```
    "idBadge" : "000001",
    "heure" : "7h59",
    "code" : 3
  },
  {
    "idBadge" : "000001",
    "heure" : "8h00",
    "code" : 1
  },
  {
    "idBadge" : "000001",
    "heure" : "17h30",
    "code" : 1
  },
  …
]
```