# Introduction à la Programmation Objet

Correcton de l'examen du 23 janvier

12 mars 2020

Le barème est donné /40. La correction est donnée en italique sous les questions, les codes sont accessibles sur la page du cours.

## 1 Cours et compréhension (10,5 points)

### 1.1 Structure de classe

**Question 1 (0.5pt)** Désigner par leur numéros de lignes les parties du code correspondant à la déclaration d'attributs, aux constructeurs et aux méthodes.
*Déclarations d'attributs : lignes 2 à 4, constructeurs : 6 à 14, autres méthodes : 16 à 30*

**Question 2 (0.5pt)** Quel est le nom de la classe ? Le nom complet du fichier ?
*nom de la classe : User , nom du ficher : User.java*

**Question 3 (1pt)** Que signifie le mot-clé `private` à la ligne 2 ? ?
*Il s'agit de la visibilité de l'attribut : il sera invisible de l'extérieur de la classe et ne pourra être accédé qu'à travers de méthodes publiques.*

**Question 4 (1 pt)** Écrire une instruction permettant de créer une instance de cette classe.
*User u = new User(20, "alias", "admin") ;*

**Question 5 (0.5pt)** Écrire une instruction permettant d'en changer l'âge à 25.
*u.setAge(25) ;*

**Question 6 (1 pt)** Que signifie le mot-clé `this` à la ligne 7 ?
*this désigne l'instance courante de la classe, l'objet construit en mémoire*

**Question 7 (0.5 pt)** Que signifie le mot-clé `this` à la ligne 13 ?
*Ici, this désigne l'appel d'un autre constructeur de la classe : en l'occurence celui défini en ligne 6.*

## 1.2 Convention de code

**Question 8 (1.5 pt)** On vous demande d'écrire un programme gérant un jeu se déroulant en plusieurs manches. Parmi ceux-ci, quel nom de variable choisir pour désigner le score d'un joueur pour une manche ? Pourquoi ?

1. `score_d_une_manche`
2. `nb`
3. `currentpoints`
4. `scorePlayerRound`
5. `MonScoreCetteManche`

*4.scorePlayerRound : il désigne clairement le rôle de la variable et respecte le camelCase (première lettre en minscule, celles des autres mots en majuscule)*

## 1.3 Debug

À l'execution d'un programme, l'exception suivante est levée :

```
1 Exception in thread "main" java.lang.NullPointerException
2     at cours.Mail.send(Mail.java:7)
3     at cours.Mail.main(Mail.java:14)
```

**Question 9 (3pt)** Quel est le soucis ? Que faudrait-il aller regarder dans le code/-comment le corriger ?

*Le programme tente d'appeler une méthode sur un objet null lors de la méthode send. L'objet utilisé n'a pas été construit en mémoire. Il manque probablement un appel à un constructeur lors de la déclaration d'une variable de type Mail ou d'un de ses attributs. Un débuggueur peut permettre de vérifier le contenu des variables lors de l'appel à cette méthode. Il faudrait relire les constructeurs ou les déclarations de variables locales concernées en fonction des contenus observés et de l'élément sur lequel porte l'invocation fautive.*

## 1.4 static

Soit une classe `Equation` contenant la méthode :

```
public static Double solver(String equation, ArrayList<Double> parameters)
```

**Question 10 (1 pt)** Comment appeler cette méthode depuis une autre classe ?
*Equation.solver(eq, param) où eq (reciproquement param) est une variable déclarée et instanciée de type String (reciproquement ArrayList¡Double¿).*
*Remarque : la classe devra être accessible, donc soit importée soit dans le même package.*

## 2 Gestion de listes d'initiés (20,5 points)

**Question 1 (1 pt)** Écrire le constructeur `public Initie(String nom, String role)` de la classe `Initie`. À sa création, un initié ne connait encore aucune information privilégiée.

*cf code*

**Question 2 (0,5 pt)** Écrire le constructeur `public Information(String titre, LocalDate dateDivulgation)` de la classe `Information`.

*cf code*

**Question 3 (1,5 pt)** En utilisant le constructeur précédent et une méthode de le classe `LocalDate`, écrire le constructeur `public Information(String titre, int jour, int mois, int annee)` de la classe `Information`, fixant la date au jour, mois et année donnés en paramètres.

*cf code*

**Question 4 (0,5pt)** Écrire la méthode `public void prendConnaissance(Information sujet)` de la classe `Initie`, qui ajoute le sujet passé en paramètre aux sujets connus par l'initié.

*cf code*

**Question 5 (1,5)** Écrire la méthode `public String toString()` de la classe `Initie`, qui devra afficher les informations sur l'initié sous la forme suivante (des exemples vous sont fournis dans le résultat de l'execution du programme, en annexe) :

```
Nom, rôle connait les sujets :
- titre du premier sujet
- titre du deuxième sujet
- titre du troisième sujet
- ...
```

*cf code*

*Remarque : le sujet demande d'afficher, ce qui est un mauvais terme, inconsistant avec une méthode toString. Ceci est dû à une imprécision du sujet. Les réponses le signalant* ***et modifiant la signature de la méthode pour un retour void*** *ont été acceptées, des points ont été retenus pour les codes inconsistants avec le type de retour. En pratique, vous ne pouvez pas changer le type de retour de la méthode toString sans en changer le nom ou la nature des paramètres.*

**Question 6 (0,5 pt)** Cette méthode est précédée de l'annotation `@Override`, pourquoi ?

*Parce qu'elle redéfinit une méthode de la classe Object, dont Initie hérite (comme toutes les classes en java).*

**Question 7 (1,5 pt)** Écrire la classe `Reunion`, contenant :

— des attributs `date` (la date à laquelle la réunion se tient), `initiesPresents` (la liste des initiés présents à cette réunion) et `sujet` (l'information, de type `Information`, sur laquelle la réunion porte).

— un constructeur `public Reunion(LocalDate date, Information sujet)`, qui initialise la date et le sujet de la réunion à ceux passés en argument et la liste d'initiés présent à une liste vide.

*cf code*

**Question 8 (0,5 pt)** Écrire la méthode `public void ajouteInitie(Initie initie)` de la classe `Reunion`, qui ajoute l'initié passé en paramètre à la liste des présents à la réunion.

*cf code*

**Question 9 (1 pt)** Écrire le constructeur `public Reunion(Information sujet)` de la classe `Reunion`, qui crée une réunion pour le jour courrant (le "aujourd'hui" de la machine sur laquelle le code est exécuté). Pour avoir tous les points, ce constructeur devra appeler le constructeur de la question 7.

*cf code*

**Question 10 (0,5 pt)** Écrire le constructeur`public Reunion(LocalDate date, Information sujet, List<Initie> presents)` de la classe `Reunion`, qui crée une réunion pour la date et le sujet donné, avec la liste de présents passée en argument. Pour avoir tous les points, ce constructeur devra appeler le constructeur de la question 7.

*cf code*

**Question 11 (1 pt)** Écrire la méthode `public boolean estPassee()` de la classe `Reunion`, qui renvoie vrai si et seulement si la date de la réunion est antérieure ou égale à la date du jour (on considère que les réunions du jour courrant se sont toutes déroulées).

*cf code*

**Question 12 (0,5)** Écrire la méthode `public boolean estConfidentielle()` de la classe `Reunion`, qui renvoie vrai si et seulement si l'information dont il est question lors de la réunion est confidentielle : c'est-à-dire si la date de divulgation au public du sujet est dans l'avenir par rapport à la date à laquelle se tient la réunion.

*cf code*

**Question 13 (1,5)** Écrire la méthode `public void ajouteInformationConfidentiellesAuxPresents()` de la classe `Reunion`, qui teste si le sujet de la réunion est confidentiel, auquel cas elle ajoute ce sujet à la liste des sujets connus de chacun des initiés présents et ne fait rien sinon.

*cf code*

**Question 14 (1 pt)** Écrire la méthode
`private static void derouleReunionsPasseesConfidentielles(List<Reunion> reunions)`
de la classe `Entreprise`, qui ajoute aux initiés les informations confidentielles dont ils ont déjà connaissance, c'est-à-dire dont les réunions sont passées.

*cf code*

**Question 15 (1 pt)** Écrire la méthode `public boolean estObsolete()` de la classe `Information`, qui renvoie vrai si et seulement si l'information est obsolète, c'est-à-dire si 3 ans ou plus se sont écoulés depuis sa divulgation au public.

**Question 16 (0,5 pt)** On veut ajouter dans la classe `Information` un attribut `connaisseurs` représentant la liste des initiés ayant connaissance ayant connaissance de l'information. Écrire la ligne de code déclarant cet attribut.
Faut-il ajouter quelque chose pour son initialisation ?

*private List¡Initie¿ connaisseurs = new ArrayList¡Initie¿() ;*
*Il n'est rien nécessaire de rajouter pour son initialisation, elle est faite lors de l'appel au constructeur en utilisant le paramètre par défaut défini*
***OU BIEN***
*private List¡Initie¿ connaisseurs ;*
*Il faut ajouter l'initialisation de le liste ( this.connaisseurs = new ArrayList¡Initie¿() ; ) dans le constructeur.*

**Question 17 (0,5 pt)** Écrire la méthode `public void devientConnuPar(Initie initie)` de la classe `Information`, qui ajoute l'initié passé en paramètre à la liste de ceux connaissant l'information.

*cf code*

**Question 18 (1 pt)** Ré-écricre la méthode `public void prendConnaissance(Information sujet)` de la classe `Initie`, pour que la liste des connaisseurs du sujet soit également mise à jour lorsqu'un initié prend connaissance d'une information.

*cf code*

**Question 19 (0,5 pt)** Écrire la méthode `public void supprimeSujet(Information sujet)` de la classe `Initie`, qui supprime l'information passée en paramètre de la liste des sujets connus par l'initié. Cette méthode ne sera appelée que pour des sujets obsolètes, il n'est donc pas nécessaire de vérifier l'obsolescence ici.

*cf code*

**Question 20 (1 pt)** Écrire la méthode `public void miseAJour()` de la classe `Information`, qui teste si une information est obsolète, et supprime le cas échéant l'information de la liste des sujets connus des initiés concernés.

*cf code*

**Question 21 (2,5 pts)** Nous voulons maintenant introduire un nouveau type de réunion : les réunions de suivi. Ces réunions sont des cas particuliers de réunion, qui ont pour particularité d'être liées à une réunion précédente sur le même sujet. Tous les participants à une réunion le seront également aux réunions de suivi, d'autres peuvent être ajoutés.

Ecrivez pour cela une classe ReunionDeSuivi héritant de la classe réunion, ayant pour attribut supplémentaire `reunionPrecedente`. Cette classe comprendra un constructeur prenant une réunion (la réunion dont elle fait suite) et une date (de type `LocalDate`) en paramètres. Le sujet de la réunion est le même que celui de la réunion précédente, la liste des présents contient initialement les mêmes participants que la réunion précédente.

Avez-vous besoin de définir autre chose dans cette classe pour pouvoir l'utiliser de manière cohérente avec le reste du programme ? Si oui, précisez votre réponse.

*cf code. Les méthodes de la classe mère sont accessibles depuis les instances de la classe fille, si on veut utiliser les attributs directement il faut les passer en protected ou ajouter setter/getter (la réponse exacte dépend de votre code)*

**Question Bonus (max 2 pts, dont 0,5 pt accordés pour les justifications ou élément supllémentaire apportés tout au long de l'exercice)** Voyez-vous des soucis avec cette implémentation ? Quels tests supplémentaires voudriez-vous faire ? Comment amélioreriez-vous ce programme ?

*Multiples réponses possibles*

# 3 Bowling (8 pts + 1 pour appréciation générale de la copie)

**Question 1 (2 pts)** Écrire la méthode `public int score()` de la classe `OnePlayerGame`, qui renvoie le score de la partie, c'est à dire la somme du score des 10 premières frames.

*cf code*

**Question 2 (1pt)** Écrire la méthode `public boolean isStrike()` de la classe `Frame`, qui renvoie vrai si la frame contient un strike. Attention, les frames 11 et 12 ne doivent jamais être considérés comme des strikes.

*cf code*

**Question 3 (1pt)** Écrire la méthode `public boolean isSpare()` de la classe `Frame`, qui renvoie vrai si la frame contient un spare. Attention, les frames 11 et 12 ne doivent jamais être considérés comme des spares.

*cf code*

**Question 4 (4 pts)** Écrire la méthode `public int score()` de la classe `Frame`, qui renvoie le score de la frame, selon les règles énoncées ci-dessus.

*cf code*