# Contrôle Final Développement Orienté Objet

22 Juin 2022
durée 2h

Consignes :

- Vous avez droit à la documentation Java, les sujets de TPs ainsi que vos programmes et notes de cours.
- La communication est évidemment interdite.
- Chaque exercice sera dans son propre dossier avec les classes correspondantes.
- Les exercices peuvent être faits dans n'importe quel ordre.
- Les diagrammes UML peuvent être faits via StarUML ou sur papier libre à rendre à la fin, avec votre nom et prénom.
- Les réponses à l'exercice 5 seront dans un fichier exercice5.txt.
- Vos réponses doivent être envoyées à luc.dartois@u-pec.fr avec le sujet :

  [ControleJava] votre_Prenom votre_Nom
- Vous y joindrez une unique archive s'appelant *reponse.tar* contenant les dossiers de vos exercices. Je rappelle la commande : **tar cvf reponses.tar ***
- Le non-respect des consignes exactes entraîne une perte de 2 points (vous aurez directement 0 si vous communiquez).

**Exercice 1.** (3 points)

Écrivez un programme Extension.java qui lit le premier argument passé en ligne de commande, et détecte son extension, selon qu'il s'agisse d'un fichier compilé (.class), non compilé (.java), ou inconnu (autres). Le programme renvoie alors un message adapté selon le résultat de la détection.

Vous pourrez vous appuyer sur la méthode endsWith(String suffix) de la class String.

**Exercice 2.** (5 points)

*1)* Écrire un programme ouvrant une fenêtre et affichant au milieu un cercle plein de couleur bleue et de 50 pixels de diamètre.

*2)* Faites en sorte que le cercle change de couleur à chaque fois que l'on clique dessus, alternant entre le bleu, le vert et le rouge.

*3)* Ajoutez deux boutons + et - permettant respectivement d'augmenter ou diminuer la taille de 10 pixels en diamètre. La taille ne devra jamais être plus petite que 10 ni plus grande que 100.

*4)* Enfin, faites en sorte qu'un clic droit de la souris repositionne le cercle sous la souris.

**Exercice 3.** (2 points)

Réalisez le diagramme objet de l'application de l'exercice précédent. Vous n'indiquerez que les méthodes contenant effectivement du code.

**Exercice 4.** (6 points) Le but de cet exercice est de modéliser le mythe de Thésée et le Minotaure (Asterius). Un Labyrinthe est une grille rectangulaire possédant une longueur et une largeur. Thésée est un héros parcourant ce labyrinthe. Son but est de retrouver Ariane, mais d'éviter le Minotaure.

*1)* Créez une classe Labyrinthe. Elle devra comprendre :

- un constructeur créant un Labyrinthe dont la taille est passée en paramètre. Thésée est placé dans le coin en haut à gauche, tandis que Ariane et Asterius sont sur des cases quelconques mais différentes.
- une redéfinition de toString() qui affiche le contenu du labyrinthe (on ne se souciera pas d'afficher les murs).
- une méthode bouger, qui prend en argument Thésée, Ariane ou Asterius, ainsi qu'une direction (n,s,e,w), et effectue le déplacement si possible.

*2)* Créez une classe Test qui crée un Labyrinthe, bouge Thésée et Ariane, et affiche le résultat.

*3)* Que se passe-t-il si quelqu'un essaye de bouger hors du Labyrinthe ? Gérez ce cas avec une exception adaptée.

*4)* Similairement, si Thésée bouge sur la case du Minotaure, il perd. Ajoutez du code gérant cette situation.

**Exercice 5.** (3 points) Le diagramme séquence ci-dessous représente la procédure suivie par un client pour réserver un vol sur un site d'agrégateur de vol.

*1)* Comment s'appelle le site sur lequel le client réserve son vol.

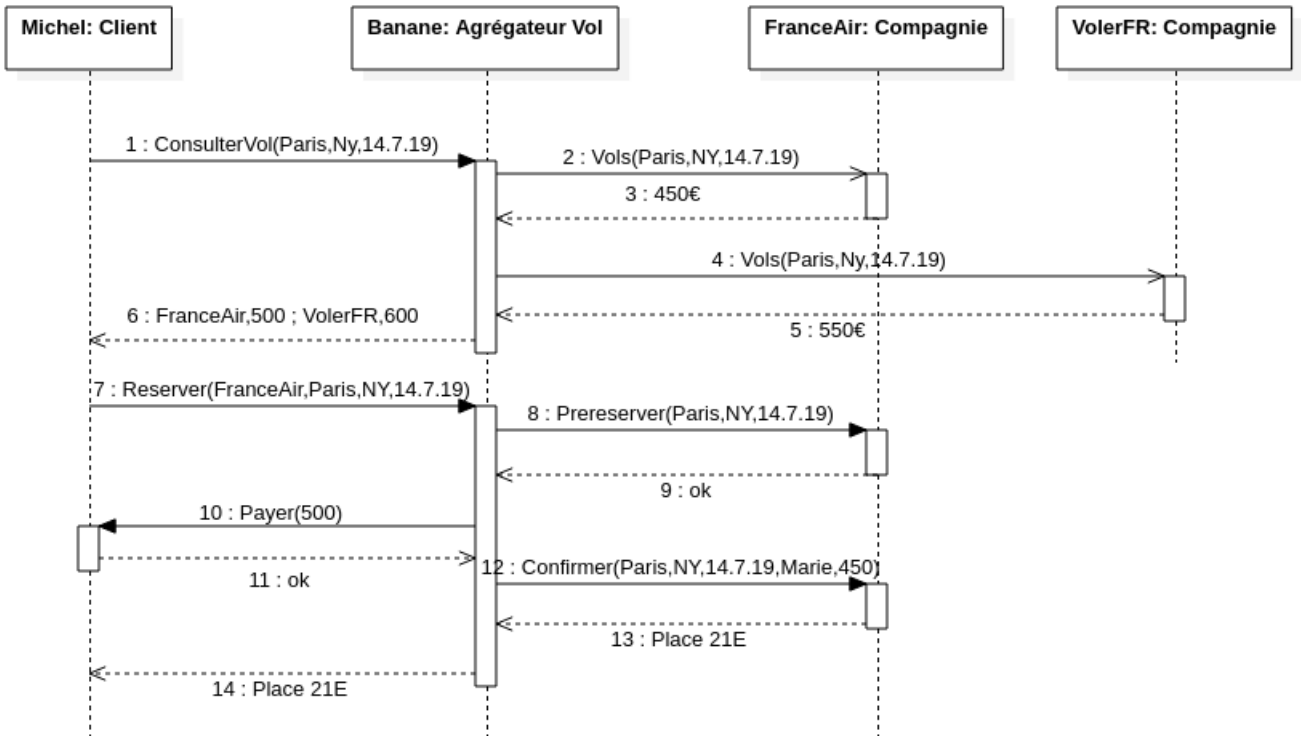
*2)* Quelles compagnies le site consulte-t-il ?

*3)* Quel objet appelle l'opération ConsulterVol du message 1 ? Sur quel objet ?

*4)* Quelle classe implémente l'opération Préreserver du message 8 ?

*5)* Écrire un *synopsis* de cette interaction du point de vue de la compagnie FranceAir.

*6)* On suppose que la compagnie FranceAir ne vend pas de place non disponible. On suppose également que quelqu'un d'autre (utilisant un autre agrégateur) a réservé la dernière place pour le vol de FranceAir entre les messages 3 et 8. Quelles opérations sont modifiées par ce changement ? Comment ?




**Exercice 6.** (4 points)

Écrire un programme ViMacsPadNano.java qui prend en argument un nom de fichier .txt, et permettant à son utilisateur d'écrire dans ce fichier, en utilisant l'invite de commande du terminal. Le programme doit s'arrêter lorsque l'utilisateur entre la ligne "assez!".

Rappels : La classe FileWriter permet d'écrire dans un fichier. La classe InputStreamReader permet de lire des InputStream tels que System.in. Enfin, les classes BufferedWriter et BufferedReader peuvent être utilisés comme surcouche et permettent de fluidifier le processus avec des méthodes permettant respectivement d'écrire une nouvelle ligne ou de lire une ligne entière par exemple.