TP1: Diagrammes de cas d'usage

Exercise 1. Initiation à STARUML : site web d'un journal

Nous allons nous initier à l'outil STARUML et aux diagrammes de cas d'usage avec un exemple simple : le site web d'un journal.

1) Lancez STARUML (via terminal ou dans applications/developpement)

2) Ajoutez un diagramme de cas d'usage en utilisant Model/Add Diagram/Use case diagram.

3) Ajoutez un Use Case Subject : il s'agit de l'entité que l'on représente, ici le site web du journal.

4) Ajoutez un Acteur : ici l'internaute qui va consulter le site.

5) Ajoutez un cas d'usage : Lire un article.

6) Associez l'acteur au cas d'usage avec un lien d'association.

Félicitations ! Vous avez réussi votre premier Diagramme de Cas d'Usage !

Enrichissons maintenant ce cas en séparant articles libres et articles réservés aux abonnés.

7) Le site ne se comporte pas de la même façon envers un abonné ou un internaute lambda.

8) Ajoutez un nouveau cas d'usage : lire un article abonnés, et reliez-le à l'acteur correspodant.

9) S'il y a des abonnés, le site web nécessite qu'il y ait une procédure d'authentification. Ajoutez-la dans un nouveau cas d'usage. Quel(s) acteur(s) peuvent lancer la procédure d'authentification ?

10) Ajouter un commentaire n'est possible que pour les abonnés. C'est une option qui peut être effectuée pendant qu'on lit un article. Ajoutez le cas d'usage correspondant, et reliez-le par une relation d'extension. On rappelle qu'une relation d'extension signifie que la source (départ de la flèche) peut être effectuée lorsque l'on effectue la destination (arrivée de la flèche).

Exercise 2. Gestion de transports en commun

On s'intéresse à la modélisation du processus de gestion du traffic dans les transports en commun d'une ville. Nous allons modéliser différents systèmes faisant partie de ce domaine, et réaliser les *diagrammes de cas d'usage* associés à chacun d'eux.

Pour chaque cas, identifiez les acteurs qui interagissent avec le système, les cas d'usage, et leurs relations, puis modélisez le diagramme de cas d'usage.

1) **Les bus.** Un bus accueille des passagers, qui peuvent payer leur course de deux façons : soit en passant un ticket, soit en passant leur badge. Afin d'éviter les fraudes, cette borne, qui fait partie du bus, communique à chaque fois le code d'identification du badge à la centrale, qui vérifie que le code est bon et non utilisé ailleurs sur le réseau. Le bus peut également etre reroutée par la centrale.

2) **Les bornes automatiques.** Une borne automatique permet à des utilisateurs d'acheter des tickets ou un abonnement, ainsi que de consulter des horaires. Lors de ces actions, la borne effectue des requetes de lecture et/ou d'écriture sur les base de données des serveurs qui sont dans la centrale. Enfin, un dépanneur peut ouvrir la borne remplir la machine de tickets ou lancer un diagnostic matériel.

3) **La centrale.** Il s'agit de la centrale évoquée dans les questions précédentes. Elle n'est pas en contact direct avec les usagers, mais interagit néanmoins avec les les bornes automatiques et les bus, selon les usages définis aux questions précédentes. Elle interagit également avec des controleurs du traffic qui peuvent ordonner le reroutement de bus.

Exercise 3. Un premier diagramme de classe
Un concessionnaire voiture a besoin d'une solution pour gérer ses données. Chacune des questions exprime un besoin du concessionnaire. Créez un *diagramme de classe* que vous compléterez au fur et à mesure afin de répondre à ses besoins. On complétera ce diagramme avec un *diagramme d'objet* avec des données de votre choix. Attention à bien représenter les classes, mais également les types des attributs et des opérations ainsi que les relations entre ces classes.

1) On veut tout d'abord gérer le parc automobile. Une voiture possède une marque, un modéle, une année, ainsi que des données techniques. Ces données sont cachées mais doivent pouvoir être obtenues par une méthode renvoyant les données demandées.

2) On veut également pouvoir gérer le personnel dans une même classe. Les personnels ont un nom, et sont soit managers, soit vendeurs, soit mécanos. Les vendeurs sont sous la tutelle d'un manager.

3) On désire également un fichier client, contenant le nom, l'adresse et le numéro de téléphone des clients du concessionnaire.

4) Enfin, on souhaite modéliser les ventes. La vente d'une voiture à un client doit être faite par un vendeur. Elle comporte également un prix.

Exercise 4. Code et Diagramme

1) Rendez vous sur `https://docs.oracle.com/javase/8/docs/api/java/awt/Point.html` pour découvrir la documentation sur la classe Point. Représentez-la dans un diagramme de classe, avec ses attributs et 4 opéarations dont getLocation, translate et distance, héritée de la classe Point2D. On fera attention aux type des attributs et aux signatures des opérations.

2) Comment est défini un segment ? Faites-en une classe et ajoutez-la à votre diagramme. Ajoutez des opérations (avec leur signature) permettant de réaliser les opérations suivantes:

- créer un nouveau segment,
- calculer et renvoyer la longueur du segment,
- changer les point d'origine et d'arrivée du segment,
- tester l'égalité de deux segments,
- (bonus) determiner si le segment est parallèle à un autre donné en argument.

3) Implémentez ces méthodes dans une classe Segment.java. Vous aurez besoin d'importer la classe point avec la commande `import java.awt.Point;` en début de fichier.

4) Testez votre classe Segment dans une méthode `main` en y déclarent des points puis des segments, et en appelant les méthodes implementées sur ces objets.

5) Que se passe-t-il si je translate le point d'origine d'un segment ? Et si un même point appartient à deux segments ?

6) (bonus) Définissez une classe Quadrilatère. Elle devra notamment contenir une méthode permettant de décider s'il s'agit d'un parallélogramme.