Contrôle de TP

13 octobre 2015 — Durée 1h45

 Lisez tout l'énoncé, et en particulier les informations pratiques avant de commencer. 

Informations pratiques.

I) Lors de ce TP nous utiliserons le programme StarUML (sous Windows).

II) On utilisera dans ce TP un seul projet et un seul modèle (mais bien sûr plusieurs diagrammes).

III) N'oubliez pas de sauvegarder régulièrement.

IV) Si à l'issue du temps imparti le diagramme sur lequel vous êtes en train de travailler n'est pas fini, l'indiquer par une note sur le diagramme.

V) Le fichier .mdj ainsi que le projet doivent être nommés avec votre nom de famille.

VI) À l'issue du TP, le sauvegarder puis l'envoyer par e-mail : `mathieu.sassolas@u-pec.fr` avec comme objet « [L3P – UML] TP solo ». Attendre l'accusé de réception avant de se déconnecter et de quitter le poste et la salle.

VII) Le TP est un examen, il est donc individuel. Les notes de cours sont autorisées. Aucun autre document n'est autorisé.

Exercice. Une station de ski

Une station de ski souhaite équiper ses remontées mécaniques d'un système informatisé permettant à la fois au gérant d'effectuer des mesures statistiques sur leur utilisation et de facturer les skieurs en fonction de leur consommation s'ils n'ont pas choisi de forfait.

Chaque skieur dispose d'un badge individuel qu'il présente devant une borne du système lors du passage à la remontée mécanique. Le système doit alors répondre si le skieur est autorisé ou non à passer. S'il est autorisé, son passage est enregistré dans le système (estampillé de la date et de l'heure). La décision prise par le système dépend de plusieurs choses.

Tout d'abord, les remontées mécaniques (téléski et télésiège) sont répertoriées dans le système avec – outre leur nom, leur longueur, le dénivelé qu'elles permettent d'effectuer – la durée de la montée. Un skieur dont le dernier passage est plus récent que la durée de la montée a visiblement prêté son badge à un autre skieur, ce qui est strictement prohibé ; il n'est donc pas autorisé à passer.

Autrement, le passage est autorisé pour un skieur qui paye à la consommation. Un skieur disposant d'un forfait ne peut passer que si la remontée en question fait partie du forfait choisi et qu'il est encore valide.

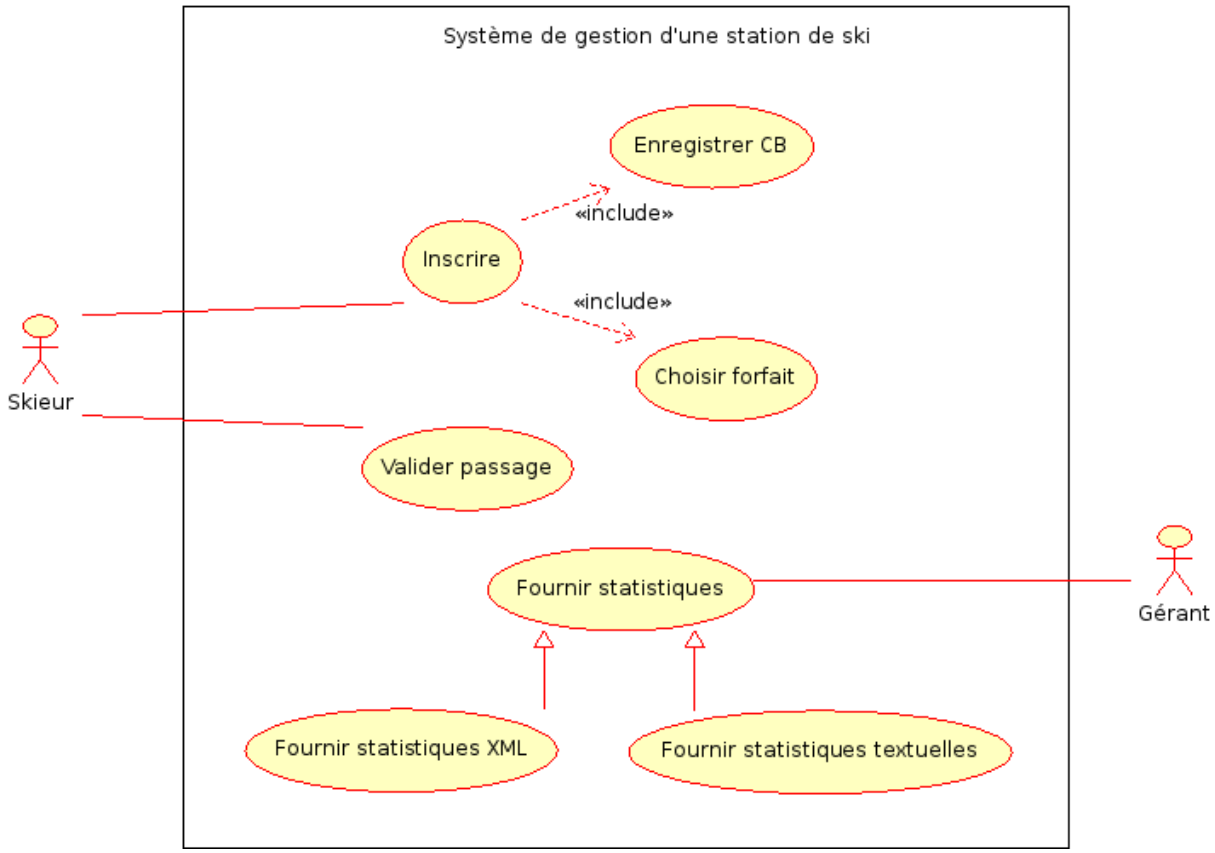
On note que le mode de facturation ainsi que le forfait éventuel est choisi dès l'inscription du skieur sur le système. Pour un forfait, le système demande au skieur les dates de début et de fin, ainsi que le forfait qu'il souhaite acheter (un forfait dispose d'un nom et d'un prix). Il enregistre alors sa carte bancaire pour le paiement. Un skieur qui paye à la consommation enregistre également sa carte bancaire car elle lui sera débitée à l'issue de la saison de ski.

Dans le but d'effectuer des statistiques, on garde pour chaque télésiège le nombre de places par nacelle. Sachant qu'il y a une nacelle ou perche tous les 20m au plus, le système permet de calculer la capacité maximale horaire de chaque remontée. La fonction de statistique d'une remontée mécanique va rechercher, pour un intervalle de temps donné, tous les passages à cette remontée, les compter en fonction de s'ils correspondent à un passage « au forfait » ou « à la consommation » et la comparer à la capacité maximale théorique. Pour les passages « au forfait », on souhaite également que les statistiques donnent le nombre de passage par forfait. Les statistiques pourront être présentées sous la forme d'un document XML ou textuelle.

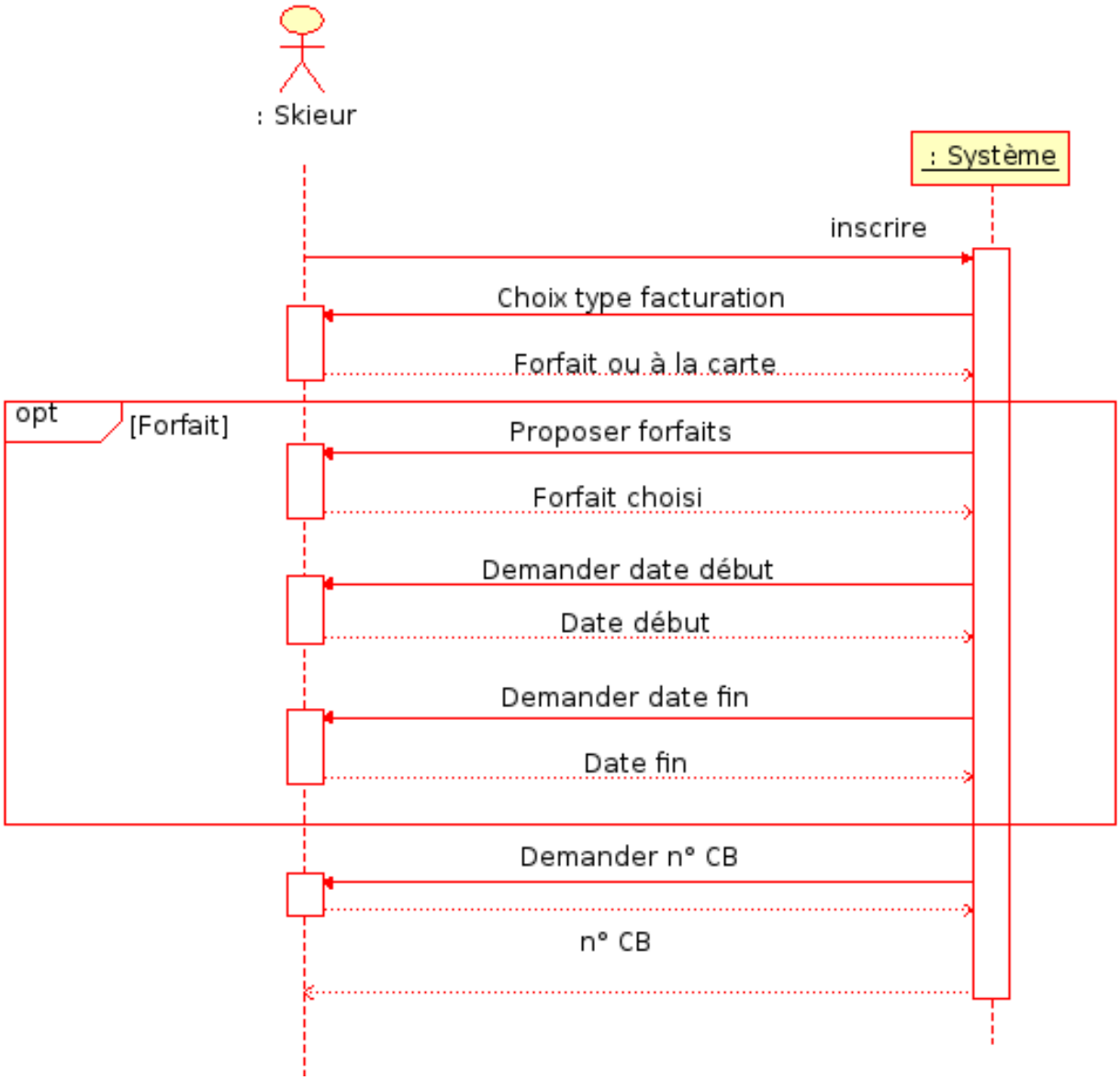
1. Construire le **diagramme de cas d'utilisation** de ce système.
2. Construire le **diagramme de séquences système** de l'inscription du skieur.
3. Construire le **diagramme de classes** de ce système. On y intégrera toutes les opérations mentionnées dans l'énoncé, mais on ne perdra pas de temps à y intégrer tous les accesseurs.
4. Construire le **diagramme de séquences** de l'opération de *statistiques* qui présente ses données sous forme textuelle. On ajoutera au diagramme de classes tous les accesseurs nécessaires. On ne s'occupera pas de détailler le fonctionnement de l'opération qui calcule la capacité maximale théorique.

Solution de l'exercice.

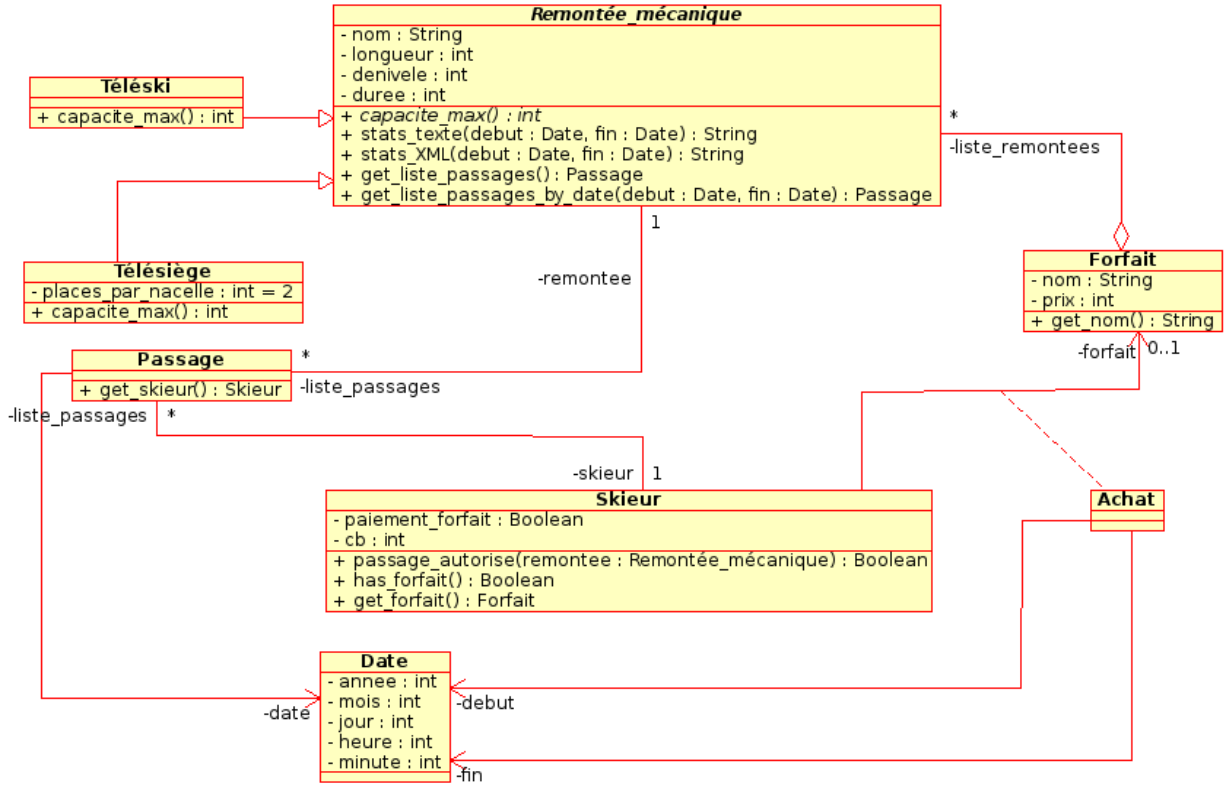
1.



2.



3. Quelques remarques :
   - La classe `RemontéeMécanique` est abstraite (en italique). L'opération `capacite_max` de cette classe est également abstraite et est concrétisée dans les classes filles `Téléski` et `Télésiège`. En termes de programmation, cela signifie que l'on devra écrire le code de cette opération séparément pour chaque classe fille. On peut cependant l'utiliser en toute généralité dans la classe mère, (cf question suivante). Ce point de détail est du bonus.
   - On a dans cette réponse les accesseurs qui ont été ajoutés pour la question suivante, à savoir :
     - `get_liste_passages` et `get_liste_passages_by_date` (un seul suffit) de la classe `Remontée_mécanique` ;
     - `get_skieur` de la classe `Passage` ;
     - `has_forfait` de la classe `Skieur` (accesseur de l'attribut `paiement_forfait`) ;
     - `get_forfait` de la classe `Skieur` (accesseur de l'attribut `forfait`) ;
     - `get_nom` de la classe `Forfait` ;



4. Quelques remarques :
   - Ici il n'est pas important de savoir dans quel ordre sont récupérées les information de capacité maximale et le comptage des passages.
   - On est par contre obligé d'accéder au détail des passages (le skieur concerné, et pour ce skieur, l'éventuel forfait) pour comptabiliser correctement. Le traitement des données n'est ici pas explicité.
   - On voit qu'on utilise l'opération abstraite `capacite_max` sans se poser de question de savoir comment elle sera implémentée (en fonction de quelle classe est la remontée en question).
   - Les objets `Passage`, `Skieur` et `Forfait` sont en réalité des instances multiples d'objets de ces classes ; on peut l'indiquer avec StarUML.