**Université de Carthage Année universitaire 2019/2020**

**Institut National des Sciences Appliquées et de Technologie Section : LA.GL3**

Module : **UML et design Patterns**

**TP N°1**

Nous souhaitons réaliser un système de gestion de vente en ligne de voitures d’occasion. Le club en question est composé de membres partageant la même passion pour les voitures, qu’elles soient de course, de collection ou normales. L’idée est de permettre à tous les membres du club de pouvoir vendre ou acheter des voitures à des prix imbattables en leurs servant d‘intermédiaire. Pour vendre une voiture, un membre se connecte avec ses, mot de passe et login. Il doit ensuite remplir la fiche de caractéristiques des voitures. S’il n’est pas encore membre, il doit s’inscrire en remplissant un formulaire. Un message d’erreur est également affiché après chaque échec de connexion. Le message stipule le nombre d’essais restants avec un nombre maximum de 3 essais. Pour acheter une voiture, le membre doit parcourir la liste des voitures disponibles, sélectionner celle qui lui convient, en afficher les caractéristiques et enregistrer sa commande. Une fois la commande validée, le catalogue des voitures doit être mis à jour. Le système doit également être muni d’une interface qui permettra à l’administrateur de gérer les membres, les voitures et les commandes (ajout, suppression, modification).

Le diagramme de cas d’utilisation est donné par la figure 1

**Questions**

1. Déterminer le modèle du domaine de l’application (diagramme de classe qui se fait dans la base d’analyse pour cerner les objets sans attributs et méthodes)
2. Déterminer le modèle d’analyse pour le cas d’utilisation « **s’inscrire »**
3. Déterminer le modèle d’analyse pour le cas d’utilisation « **commander voiture »**
4. Déterminer le diagramme de séquence pour le cas d’utilisation **«s’inscrire »**
5. Déterminer le diagramme de séquence pour le cas d’utilisation « **commander voiture »**
6. Déterminer le diagramme de classe pour chacun des deux cas et appliquer le pattern *« factory method »* pour la création des membres ainsi que la création des commandes

Design Pattern : un modele de conception dans la conception il ya des problemes qui se repetent 🡺 une solution a ce probleme qu’on applique sur notre probleme

Factory method : But : introduire une methode abstraite de création d’un objet en reportant aux sous-classes concrètes la création effective

Abstract Creator Concrete creator Abstract Product Concreate Product

methode de fabrique / créer des instances



*Figure 1*