Exceptions, StringBuilder et paquetage

Exercice 1 - La classe Parking

Avant propos, à partir de maintenant il vous est demandé d'utiliser eclipse ou netbeans et d'écrire votre code directement dans ces environnements.
On veut maintenant écrire une classe `Parking` qui sert à stocker des voitures. Les voitures seront stationnées sur des places numérotées en partant de 0. Le nombre de places est fixé une fois pour toutes à la construction du garage.
Dans les questions suivantes une méthode qui prend en paramètre un indice de place de garage qui n'existe pas devra lever une exception de type `java.lang.IndexOutOfBoundsException`.

1 Écrire le constructeur de la classe `Parking`.
2 Écrire une méthode `park` qui prend en paramètres une voiture, un numéro de place, qui gare la voiture à la place donnée si elle est libre, lève une exception de type `java.lang.IllegalStateException` si la place est occupée.
3 Écrire une méthode `unpark` qui prend en paramètre un numéro de place et qui retire du garage la voiture à cette place. La méthode retourne l'objet Voiture récupéré. Elle lève une exception de type `java.lang.IllegalStateException` si la place est vide.
4 Écrire une méthode `toString` affichant l'état du garage : pour chaque place, le numéro, ainsi que les caractéristiques de la voiture qui l'occupe si elle existe.
Vous pouvez regarder la documentation de la classe `java.lang.StringBuilder`.

Exercice 2 - La classe Parking mais mieux

1 On souhaite obliger le programmeur qui utilise la classe `Parking` à récupérer les exceptions.
Que doit-on faire ?
Fabriquer trois classes `NoCarAtThatPlaceException`, `AlreadyACarAtThatPlaceException` et `OutOfParkingException`.
2 En lisant la documentation que la classe `java.lang.Exception`, on remarque que celle-ci possède 4 constructeurs.
Dans les classes que vous venez de concevoir, quels constructeurs vous proposez-vous d'implémenter ?
Ré-écrire la classe `Parking` pour qu'elle lève les bonnes exceptions.

Exercice 3 - Package et compilation

On souhaite mettre en place une architecture permettant de séparer les fichiers source (.java) des fichiers bytecode (.class).
Pour cela, créer les répertoires `src` et `classes` s'il n'existe pas déjà !
Désactiver la compilation automatique d'eclipse (`Project > Build automatically`) puis supprimer l'ensemble des classes du répertoire `classes`.
Pour cet exercice, vous travaillerez en compilant les classes en utilisant `javac` dans une console et non en utilisant le compilateur interne d'eclipse.

1. Créer le paquetage `fr.umlv.td.parking` dans le repertoire `src` et migrer l'ensemble de vos classes dans ce paquetage.
2. Recompilez l'ensemble de vos classes à partir de la racine de ce package (i.e. le répertoire `src` en indiquant au compilateur (`javac` avec l'option "-d") de rediriger les fichiers compilés dans le répertoire de destination `classes`
3. Déplacer la classe de test (contenant le `main`) dans le paquetage `fr.umlv.td.main`
   Comment doit on compiler et exécuter le code de test pour que celui-ci marche ?

Exercice 4 - Travail de fourmi [à la maison]

Ecrire un script ANT (cf le cours) faisant la compilation et l'exécution du code de test

1. Ecrire une target `compile` qui compile l'ensemble des fichiers de `src` et qui stocke ceux-ci dans `classes`.
   Que doit-on faire si le répertoire `classes` n'existe pas ?
   Crée un target `prepare` pour cela et indiquer que la target `compile` dépend de la target `prepare`.
2. Ecrire une target `clean` qui efface l'ensemble des `.class` générée.
3. Ecrire une target `run` qui execute le code de test.
4. Ecrire une target `jar` qui cré un jar exécutable correspondant au code de test.
   Pour tester votre jar :

   ```
   java -jar parking.jar
   ```

5. Ecrire une target `all` qui sera la target par défaut et qui lancera la compilation ainsi que la création du jar.

Exercice 5 - La classe Parking megabien!!

1. Écrire une méthode `getLocation` qui prend en paramètre un numéro d'immatriculation et qui retourne le numéro de place occupée par la voiture ou -1 si elle n'existe pas.
2. Écrire une méthode `remove` qui prend en paramètre un numéro d'immatriculation et qui retire du garage la voiture trouvée. La méthode retourne l'objet `Car` récupéré ou `null` si la voiture n'est pas dans le parking.
   Quelle exception doit lever cette méthode ?
   Attention : cette méthode ne doit lèver que les exceptions qui sont logiques.
3. Déclarer une super-classe aux exceptions levées par les méthodes du parking. Quel est l'intérêt d'une telle exception pour la méthode `remove` ?
4. Réorganiser votre code pour éviter de devoir attraper les exceptions qui ne peuvent être levées.
5. Quelle est la complexité de l'implantation de `getLocation` que vous avez fournie ?