Exceptions, StringBuilder et Héritage

Exercice 1 - La classe Parking

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)

- 1 Pour cela, créer les répertoires `src` et `classes`.
- 2 Créer le paquetage `fr.umlv.td.parking` dans le repertoire `src` et migrer l'ensemble de vos classes dans ce paquetage.
- 3 Recompiliez l'ensemble de vos classes à partir de la racine de ce package (i.e. le repertoire `src` en indiquant au compilateur (avec l'option "-d") de rediriger les fichiers compilés dans le repertoire de destination `classes`
- 4 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 ?

- 5 Ecrire un script ANT faisant la compilation et l'exécution du code de test.

Exercice 4 - 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 lever 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 ?