Contrats
Exercices

Finalement, il serait plus facile de récupérer tous les fichiers sous la forme de l'archive zip suivante fichiers.

1. Ouvrez le fichier *Camion.java* dans un éditeur.
   (a) Observez que pour la méthode `estConduitPar`, il n'y a pas de précondition qui spécifie que `c != null`.
   (b) Faites charger le fichier *contrats.key* dans KeY et sélectionnez la vérification de la postcondition de la méthode `estConduitPar` de la classe *Camion.java*.
   (c) Quelle est la signification du contrat qui s'affiche dans le "Contract Configurator". Est-ce ce à quoi on s'attendrait ?
   (d) Après avoir confirmé ce contrat, vérifiez que dans la formule que KeY génère pour ce contrat, l'hypothèse `c != null` a été ajoutée. **L'outil ajoute donc des hypothèses pour simplifier l'écriture des contrats en éliminant des cas extrêmes.**
   (e) Est-ce que l'outil vérifiera ce contrat ? Essayez-le pour voir !
   (f) Toujours pour le même contrat, vérifier que même si on enlève `this != null` des préconditions, cette contrainte sera toujours présente dans les hypothèses de la formule à vérifier.
   (g) Faites faire la vérification par l'outil pour vérifier que ça ne change pas le résultat.
2. Toujours la méthode `estConduitPar`, vérifiez les choses suivantes.
   (a) Quelle est la signification des clauses `assignable` ?
   (b) Est-ce que ces clauses sont nécessaires ? Pourquoi ?
   (c) Vérifier vos réponses à la question précédente en enlevant l'une, l'autre ou les deux clauses précédentes et en faisant faire la vérification. Est-ce toujours cohérent avec votre réponse ?
3. Toujours avec le fichier *Camion.java* ouvert dans un éditeur.
   (a) Quelle est la signification du contrat de la méthode `charger` ?
   (b) Lisez le code pour le comprendre et déterminer si le contrat devrait être vérifié.
   (c) Est-ce que KeY permet de vérifier ce contrat ? Essayez-le ! Est-ce cohérent avec votre réponse précédente ?
   (d) Quel changement à la signification du contrat est-ce que le fait de remplacer `==` par `>=` dans la clause `requires prochaineLivraison == noLivraisonsMax;` produirait ?
   (e) Effectuez ce changement et faites faire la vérification. Est-ce que votre intuition est confirmée ? Sinon pouvez-vous expliquer, en relisant le code, ce que cette modification a changé dans le contrat ?

(f) Si le contrat modifié n'est plus vérifié, c'est qu'il n'est plus adéquat pour le code. Dans ce cas, quelle modification mineure (qui ne change pas fondamentalement la méthode) peut-on apporter au code pour vérifier le contrat ? Essayez votre solution avec l'outil.

4. Dans le contrat précédent, on vérifie qu'en cas d'exception l'attribut `prochaineLivraison` n'est pas modifié. Mais cette classe a aussi d'autres attributs.
   (a) Modifiez le contrat pour vous assurer que les attributs `chauffeur`, `poids` et `poidsMax` ne sont pas modifiés eux non plus.
   (b) Faites faire la vérification par l'outil.
5. Dans la classe `Chauffeur` :
   (a) Ajoutez à la méthode `conduit(Camion c)` un contrat qui spécifie qu'après l'appel `this` conduit `c` et `c` est conduit par `this`.
   (b) Faites faire la vérification par l'outil, en vous assurant de bien choisir la bonne méthode.