Projet Cartaylor - ACO

Arnaud DELOURMEL, Tom Rousseau

Projet Cartaylor - ACO
- Introduction
- Version 1
  - Choix de conception
  - User stories
  - Tests
- Version 2
  - Choix de conception
  - User Stories
  - Tests

Introduction

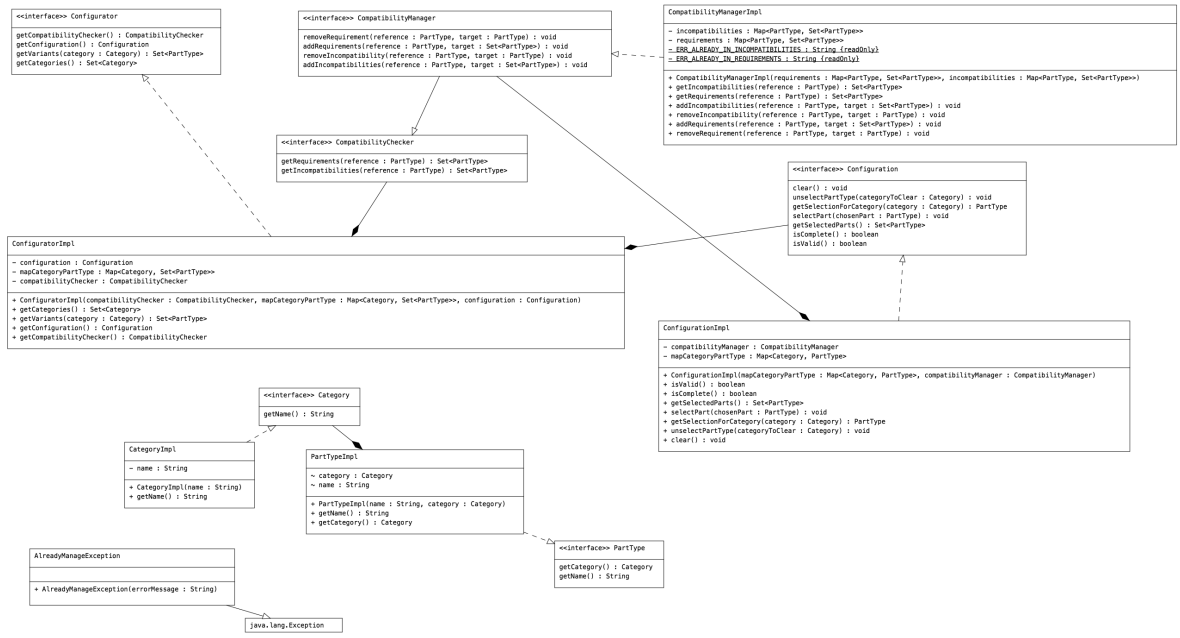
Le but de ce projet est de s'entraîner à développer une application à l'aide de nos connaissances en génie logiciel acquises lors de notre cursus. Cette application est un configurateur de voiture où l'utilisateur peut faire des choix entre quatre catégories afin de personnaliser son bolide.

Version 1

Disponible sur la branch `v1` de gitlab

Choix de conception

Diagramme de classe de la version 1 :



Pour cette première version, nous avons choisi une conception fortement ressemblante à la conception faite en TD. La partie `api` contient six interfaces décrivant chacune une partie de l'application.
Cinq classes composent la partie `engine` et implémentent les interfaces de l'API.

- La partie `Configuration` contient une map reliant chacune des quatre catégories du sujet (Engine, Transmission, Exterior et Interior) à une PartType.
- La partie `Configurator`, elle, va gérer les incompatibilités et les obligations de chaque PartType. Les méthodes de ces classes sont utilisées dans l'opération `isValide()` qui vérifie si une configuration est valide. Cette opération se trouve dans la classe `ConfigurationImpl`
- La partie `CompatibilityManager` permet de d'ajouter et de supprimer les obligations et les incompatibilités de chaque `PartType`. Les opérations d'ajout peuvent déclencher une erreur dans le cas où l'on souhaiterait ajouter une `PartType` dans la liste des obligations mais qu'elle est déjà présente dans la liste des incompatibilités.
- La partie `PartType` décrit simplement les PartType en leur donnant comme paramètres un nom et une catégorie (exemple: "tm5" pour une PartType de catégorie Transmission).

User stories

1. As a user I can get the list of part categories (see table above) to select a category.
    - Cela se fait à l'aide l'opération `getCategories()` dans l'interface `Configurator`.
2. As a user I can select a part variant for a given part category to put that part in the configuration.
    - Cela se fait avec l'opération `setVariants()` se trouvant aussi dans l'interface `Configurator`.
3. As a user I can see if my current configuration is valid or not.
    - Cette user story est implémentée à l'aide de l'opération `isValid()` dans `Configuration`.
4. As a user I can remove any part from the current configuration.
    - Cela se fait à l'aide de l'opération `unselectPartType(Category categoryToClear)`, dans l'interface `Configuration`.
5. As an application admin I can edit the set of incompatible parts and required part stored in the application.
    - L'admin peut accéder aux différentes méthodes se trouvant dans la classe d'implémentation `CompatibilityManagerImpl` pour gérer les différentes incompatibilité et obligations.

Tests

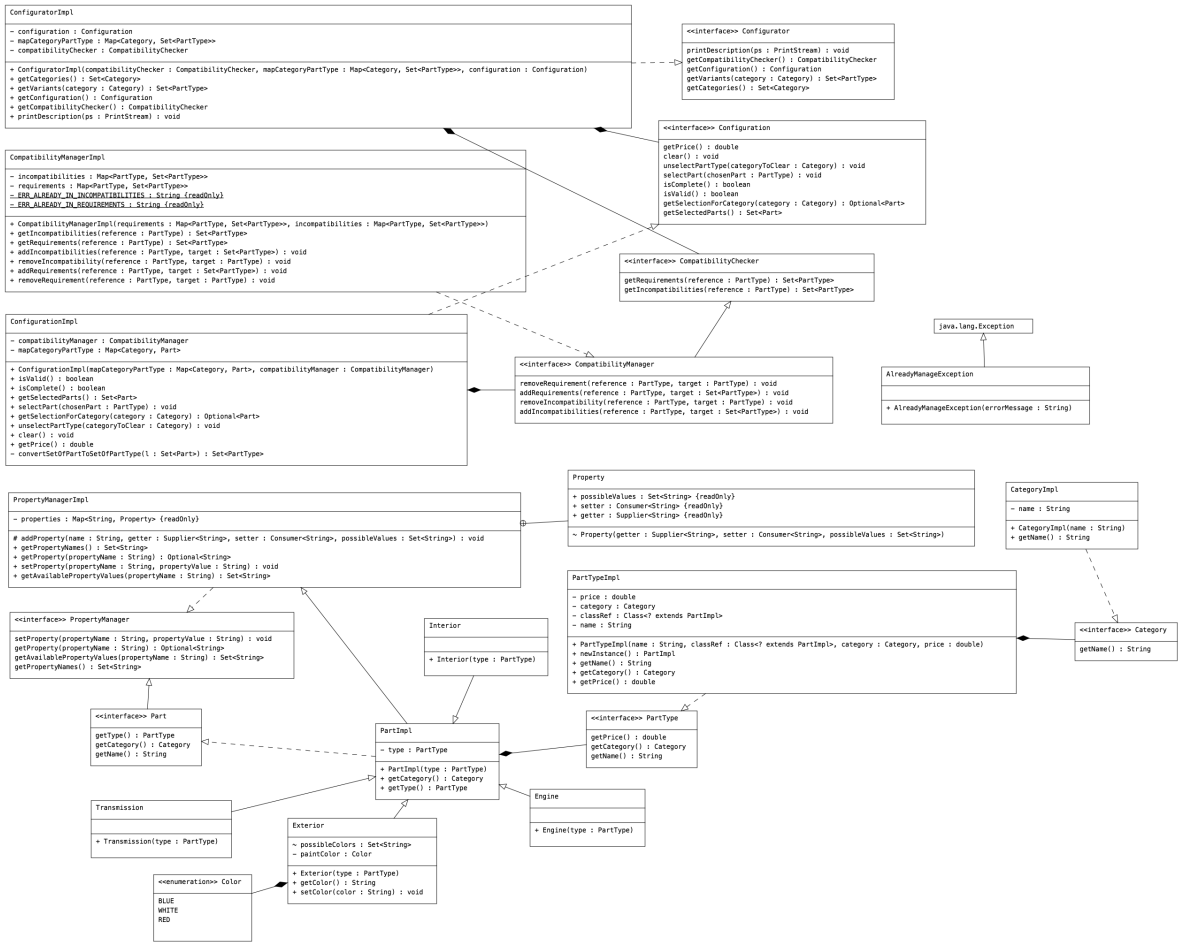
Notre version 1 de l'API, a une couverture de test de 100%. Nous avons généré le rapport de couverture de tests. Ce rapport est disponible dans le fichier *"testCoverage"*.

Version 2

Disponible sur la branche `v2` de gitlab

Choix de conception

Diagramme de classe de la version 2 :



Pour cette deuxième version du projet, nous nous sommes aussi basés sur ce que nous avons fait en cours de TD. La différence majeure qu'il y a avec la première version est l'interface `Configuration`, qui change de signature. En effet, les nouvelles user stories que nous devons implémenter montrent qu'un `PartType` ne contient pas assez d'informations pour traiter des choix individuels tels que la couleur. Comme expliqué précédemment, et tout comme dans les sites web de commerce en ligne, un client choisit un type de pièce mais obtient une instance de celui-ci, qui est elle-même configurable par le biais de propriétés individuelles (par exemple, la couleur, la taille, le goût, etc.). En conséquence, les opérations permettant d'obtenir des éléments de la configuration actuelle et doivent retourner une instance et non un type.

- Pour contrer cela, nous ajoutons une interface `Part` qui va être notre instance de nos `PartType`.
- La création d'instance se fait dans la classe d'implémentation `PartTypeImpl` à l'aide de la méthode `newInstance()`.
- Pour ajouter la propriété de "couleur" demandée, nosu créons un package `car` se trouvant dans l'`engine` de notre projet. Dans ce package, quatre `Part` son créées par le biais de classes. Dans la class `Exterior` nous ajoutons la possibilités pour qu'un utilisateur puisse ajouter une couleur parmis bleu, blanc ou rouge.
- Les propriétés peuvent-elles êtres ajoutées par l'admin à l'aide de l'interface `PropertyManager`.

User Stories

6. As a user I can get an HTML description of my configuration, if it is complete and valid.
    - Cela se fait à l'aide de la méthode `printDescription(PrintStream ps)` se trouvant dans l'interface `Configurator` de l'API.
7. As a user I can get the current price (in euros) of my configuration, even if it is incomplete (but it must be valid).
    - Cela se fait à l'aide de la méthode `getPrice()`, renvoyant le prix de la configuration. Cette méthode se trouve dans l'interface `Configuration`.
8. As a user I can select the color of the exterior paint, from a limited subset provided by the application (the subset can depend on the exterior part type that I have selected).
    - Cela est possible, comme expliqué précedemment, dans la `Part` `Exterior` se trouvant dans le package `car`.

Tests

Pour la version 2, notre couverture de test des méthodes est de 100%. Cependant, nous avons 97% des lignes qui sont couvertes par nos tests pour les classes d'implémentation. En effet, nous n'avons pas réussi à tester l'intégralité de la méthode `newInstance()` dans la classe `PartTypeImpl`

Nous avons mis à jour différents tests pour qu'ils soient en accord avec les nouvelles spécifications. Nous avons également ajouté des tests pour tester les nouvelles méthodes qui ont été implémentées. Enfin, comme pour la v1, le rapport test généré est disponible dans le fichier *"testCoverage"*.