Tp refactoring 2018

1 Création de programmes nécessitant un refactoring

1.1 Move type

Ce refactoring cherche à éviter l'introduction d'un nouveau type à l'interrieur d'un autre (types imbriqués inutiles). Pour ce faire on "move" (déplace) un type.

Le refactor proposé est de "move" la classe Roue en dehors de la classe Voiture. En effet, il n y a nul besoin d'une classe imbriquée Roue dans Voiture.(Voir dossier RefactorMT)

1.2 Introduce Parameter Object

Ce refactor propose de regrouper des arguments allant ensemble. On peut constater ces groupes d'arguments après coup ou après une analyse.

L'exemple classique proposé sont tous les arguments formant une adresse utilisés à travers plusieurs classes. Il est alors utile de regrouper ceux ci.

2 Application de refactorings sur les programmes d'un autre développeur

2.1 Refactor de Pierre Misse : "Push Down"

But: move la méthode Etudier vers la classe fille Student.

- Etre dans le fichier où la méthode à push down se situe
- Accéder au menu refactor dans la barre des menus
- Choisir Push Down puis la méthode à push down
- Suivant puis finish.

La méthode Etudier() a été déplacer vers la classe fille de Person, Student. L'attribut skill appartenant à Person est passé en visibilité protected afin d'être utilisable dans la classe Fille.

Il aurait été plus judicieux de générer un getter/setter pour l'attribut et faire appel à ses derniers plutôt que de changer la visibilté de l'attribut.

2.2 Refactor de Bruno Verley : "Encapsulate Field"

But : encapsuler l'attribut parcours de la classe Etudiant.

- Sélectionner le champ à refactor (clique sur parcours)

- Accéder au menu refactor dans la barre des menus
- Choisir Encapsulate field
- Suivant puis finish.

Des getters et setters sont créés pour l'attribut parcours. Ce dernier est rendu privé.

Attention, dans le main de la classe Parcours, le changement en l'appel du setter n'a pas été fait automatiquement et on continue d'appeler le champ, un autre mini refactor a été nécessaire pour arranger celà (et donc compiler).

3 Réflexion libre sur les refactorings d'eclipse

Mon choix pour cette question s'est porté sur les refactoring de type "Moving features", tag sur le site https://refactoring.com, ces derniers sont aux nombres de 10.

Nous les traiterons dans l'ordre que propose le site. Les problèmes et buts sont ceux proposés, nous expliciterons ensuite si Eclipse les implémente.

- Extract class:
  Problème: Une classe fait le travail prévu pour deux.
  But: Créer une nouvelle classe et y déplacer les propriétés concernées.
  Ce refactoring existe sur Eclipse. Il a l'air de faire le nécessaire pour résoudre le problème, c'est à dire, créer un nouveau "container" et y déplacer les propriétés. La résolution des références est faite, les getters et setters sont apperement en options.

- Extract module:
  Problème: Du code est dupliqué dans une ou plusieurs classe.
  But: Supprimer le code doublon ou le réunir par généralisation.
  Note: inverse du "inline module"
  Ce refactoring n'est pas proposé en tant que tel. Il est cependant possible d'obtenir un résultat assez similaire au but avec une combinaison d'autres refactorings (extract class, move property ...).

- Hide delegate:
  Problème: Un client appel une classe déléguée d'un objet.
  But: Créer des méthodes sur le serveur pour cacher le delégué.
  Note: inverse du "remove middle man".
  Eclipse ne faisant pas la distinction entre client et serveur, il est du devoir du programmeur de faire le necéssaire. D'autres refactorings peuvent être utilisé dans ce but.

- Introduce foreign method:
  Problème: Ajouter une méthode sur une classe du serveur ou une classe

non modifiable.
But: Créer la méthode dans une classe du client avec une instance de celle du serveur en tant que premier argument (pas sur).
Eclipse ne faisant pas la distinction entre client et serveur, il est du devoir du programmeur de faire le necéssaire. D'autres refactorings peuvent être utilisé dans ce but.

- Move field/method:
  Résumable à "move properties". Problème: Une propriété n'est pas dans la bonne classe.
  But: Déplacer la propriété vers la bonne classe.
  Celui là a été travaillé par Eclipse. Il est possible de "move" plus que les propriétés (classe, package, compilation units, etc...). Ce refactoring est la base de départ pour beaucoup d'autres.

- Inline class:
  Note: inverse du "Extract module".
  Pareil que pour son inverse.

- Remove middle man:
  Note: inverse du "Hide deleguate".
  Pareil que pour son inverse.

La plupart des refactorings impliquant un "move" ont l'air d'avoir été étudié par Eclipse et sont plutôt efficaces. Seul ceux en rapport avec un réseau ne le sont pas mais il semble peu évident à part avec un IDE spécialisé ou/et une bonne distinction des parties clients ou serveurs de les rendre possible aisément.