Remplacer des calculs conditionnels avec Strategy

Mécanique générale(à appliquer)

1 . Sur une classe (que nous appellerons « A »), identifiez une méthode de calcul, ou des méthodes d'assistance pour une telle méthode, qui contiennent beaucoup de logique conditionnelle. Cette classe sera connue sous le nom de votre classe de contexte car ce sera le contexte d'un objet Strategy.

2. Créez une classe concrète et nommez-la en fonction du comportement effectué par la méthode de calcul choisie. Ce sera vote Strategy.
Vous pouvez ajouter le mot « Strategy » au nom de la classe si vous trouvez que cela aide à communiquer. le but de ce nouveau type.

3. Appliquez la « Move Method » (Fowler ou https://refactoring.guru/move-method) pour déplacer la méthode de calcul principale et toutes les méthodes d'assistance vers votre classe de Strategy. Si le code que vous déplacez doit obtenir des informations de A, transmettez A en tant que paramètre à la méthode de calcul principale ou en tant que paramètre au constructeur de la classe Strategy et assurez-vous que les informations sur A sont accessibles au public.
Vous pouvez également transmettre les informations nécessaires de A à la stratégie, sans passer une référence de A à la stratégie. Cela entraînera moins de couplage entre A et votre stratégie, mais peut vous obliger à transmettre beaucoup d'informations.

4. Créez un champ (que nous appellerons « S ») dans A pour la Strategy et l'instancier.

5. Mettez à jour la méthode de calcul principale dans A pour déléguer le calcul à S.

6. Compilez et testez

7. Sur la classe Strategy, appliquez *Replace Conditional with Polymorphism* [Fowler] sur la méthode de calcul principale et sur toutes les méthodes d'assistance que vous avez déplacées de A. Il est préférable d'effectuer cette étape lentement, en vous concentrant sur l'extraction d'une sous-classe à la fois, puis en effectuant les étapes 8 et 9 ci-dessous, puis en répétant cette étape.

Une fois terminé, vous aurez considérablement réduit la logique conditionnelle dans votre classe de stratégie et vous aurez défini des classes de stratégie concrètes pour chaque variété de calcul avec lequel vous avez commencé.
Pensez à appliquer *Form Template Method* [Fowler] pour la méthode de calcul principale de votre stratégie.
Vous pouvez également faire de votre Strategy originale une classe abstraite.

Ajoutez du code à A pour utiliser sa logique interne pour définir la valeur de S ou pour permettre à un client externe de transmettre une valeur pour S.
Si vous optez pour cette dernière approche, laissez les clients transmettre une valeur pour S via des appels au constructeur si les clients n'ont pas besoin de modifier la valeur de S au moment de l'exécution. Sinon, fournissez une méthode *setter* pour permettre aux clients de définir la valeur de S au moment de l'exécution. Pour plus de commodité, vous pouvez également faire les deux. Si les clients peuvent transmettre une valeur de S à A, vous devrez mettre à jour le code pour chaque client existant de A.

9. Compilez et testez