Journal de bord Tennis Refactoring Kata

Membres du groupe E: modifié le 02/04/2024

Kenza Merzouk - Mohammed Daoud - Ahmed Bouzidia - Younes Djebali - Adam Achabar

Partie analyse:

L'analyse et classement des odeurs selon leur gravité et récurrence lors de la première séance, en se basant sur le modèle que le prof a fourni.

fait le 26/02/2024

Analyse et détection d'anti pattern et odeurs

modifié le 02/04/2024

Odeur	Gravité	Réccurence	Risque
Noms d'attributs et de méthodes non descriptifs	Moyenne	Courante	4
(IF TREE)	Moyenne	Courante	5
God Class / God Method	Moyenne	Faible	4
Utilisation magique de chaînes de caractères	Moyenne	Fréquente	5
Absence de validation des entrées	forte	Courante	4
Absence de documentation	Moyenne	Faible	3
absence de catch d'erreurs (else prend tous les autres cas possible y compris les erreurs)	Faible	Faible	1
manque de couverture des tests (pas tous les scénarios traités)	Moyenne	Faibe	3

Le document Analyse_Tennis2.md disponible sur le repository git détaille chaque odeur, les problèmes qu'elle cause, les améliorations qu'on propose.

Partie résolution:

Nous avons procédé à l'application des améliorations proposées dans le document Analyse_Tennis2.md dans cet ordre selon leur priorité: supprimer ce qui n'a aucune plus value comme les getters ainsi alléger ma classe, renommer les variables afin que le code soit plus clair, diviser la god méthode en plusieurs petites méthodes et surveiller la dépendance des une des autres (l'imbrication), implémenter la golden master pour tester le fonctionnement actuel, ajouter la feature de langue à ma classe et modifier les tests afin qu'ils prennent en compte cette feature.

Note: j'ai procédé de manière itérative et incrémentale, donc pour chaque modification faite, je relançais les tests afin de m'assurer que rien n'a été cassé, mais surtout j'ai créé une

autre classe lanGame2 dans laquelle j'applique mes améliorations au lieu d'écraser l'existant TennisGame2, ainsi je fais le record en faisant appel à l'originale, ensuite je lance le replay avec ma classe améliorée.

1. **Relecture de notre analyse:**
    a. revoir les odeurs et anti patterns que nous avons identifié

justification: prioriser le travail selon le niveau de gravité et criticité (dépendance d'une méthode du bon fonctionnement d'une autre)

2. **Traiter les odeurs: application des améliorations notées lors de la 1 ère séance:**
    a. Supprimer les getters et setters non nécessaires pour notre petit programme

justification: Nul besoin de getter ni setter, car tout le programme se résume dans le même fichier, on ne fait pas d'appel depuis une autre classe aux attributs ou méthodes

    b. Renommer les attributs et noms de méthodes non clairs

justification: Important pour la clarté du code et sa maintenance

    c. Revoir le traitement de won_point: les 2 if sont explicites avec un else qui catch les erreurs

justification: il ya de la permission, un seul if explicite, le else prend tous les autres cas possibles y compris les erreurs de frappe etc, donc important de bien définir les if et le cas d'erreur

3. **Modifier la méthode score:**
    a. Remplacer le « if tree » par des petites fonctions

justification: Reconnaître les patterns qui se répétèrent afin de supprimer le « if tree » et les diviser en petites méthodes où 1 fonction -> 1 traitement -> un pattern Ainsi la god class est distribuée

4. **Ajouter des tests**
    a. Implémenter une golden class avec méthode test _record et test_replay

justification: Mode itératif : Cette classe golden_master, on lui rajoute des bouts au fur et à mesure qu'on factorise le code de Tennis2 dans une autre classe lanGame2, afin de bien valider que les méthodes qu'on implémente ne cassent rien et fonctionne toujours comme celles qui existent dans Tennis2

5. **Ajouter une feature de langue (traduction)**
    a. ajouter à ma classe lanGame2 l'attribut de langue, et je modifie toutes les méthodes concernées par un affichage (print)

justification: La classe lanGame2 qui a besoin d'adaptation, rajouter une énième méthode, 2 tableaux pour les affichages « love, fifteen, etc » en anglais et français Gérer ce tableau en haut niveau (fonctions prédéfini) plutôt qu'indice en dur pour afficher les scores

6. **Modifier golden master**
    a. ajouter le traitement des 2 langues à ma classe golden_master, dans les méthodes record et replay, je gère le stockage dans 2 sous dossiers différents selon la langue

justification: Je trouve que c'est plus pertinent d'un point de vue organisation de travail de séparer les résultats de tests selon langue dans 2 sub dossiers différents surtout pour se retrouver si bug il y a et que l'IDE ne sauve pas et devoir debug manuellement.
Sinon la méthode est itérative, on ajoute quelques lignes de codes à la fois, on teste on lance record pour FR et replay avec les modifications, pour EN je fais pareil mais je vérifie manuellement également (car il s'agit d'une nouvelle feature)