Refactorisation de la classe TennisGame2 (C#)

1. **Simplification et refactorisation du code**
    - **Pourquoi ? :** Refactoriser le code pour le rendre plus simple et lisible est essentiel avant d'ajouter de nouvelles fonctionnalités. Un code propre et bien structuré réduit les risques d'erreurs futures et facilite les ajouts de fonctionnalités.
    - **Importance** : Cette étape est primordiale pour rendre le code plus modulaire et plus facile à modifier. Cela assure une base solide avant l'ajout de nouvelles fonctionnalités.
2. **Utilisation de tableaux pour les descriptions des points**
    - **Pourquoi ?** Les tableaux permettent de centraliser les descriptions de points, réduisant la répétition et augmentant la lisibilité.
    - Importance : Les constantes augmentent la lisibilité du code et réduisent les erreurs potentielles. C'est une bonne pratique de programmation qui facilite la maintenance du code et assure que les valeurs de score sont cohérentes et facilement modifiables.
3. **Élimination des variables temporaires inutiles**
    - **Pourquoi ?** : En supprimant les variables inutiles, on réduit la complexité du code, le rendant plus facile à suivre.
    - Importance : En simplifiant le code et en supprimant les variables inutiles, on rend le code plus propre et plus facile à suivre, facilitant ainsi l'ajout de nouvelles fonctionnalités et la correction des erreurs.
4. **Ajout d'un historique des points**
    - **Pourquoi ?** : Enregistrer l'historique des scores permet de suivre l'évolution du jeu et d'analyser les performances des joueurs.
    - Importance : Cette fonctionnalité améliore l'utilisabilité et fournit des outils supplémentaires pour les utilisateurs et les développeurs.
5. **Ajout de la réinitialisation du jeu**
    - **Pourquoi ?** Réinitialiser les scores à zéro permet de recommencer un jeu sans redémarrer l'application.
    - Importance : Cette fonctionnalité est cruciale pour les tests et les sessions de jeu répétées, rendant l'application plus flexible et plus pratique.

Amélioration des tests unitaires

1. **Extension de la couverture des tests**
    - **Pourquoi ?** : Les tests doivent couvrir toutes les fonctionnalités, y compris les nouvelles, pour assurer que le code fonctionne correctement dans tous les scénarios.
    - Importance : Assurer que toutes les fonctionnalités, y compris les nouvelles, sont couvertes par des tests est essentiel pour maintenir la robustesse et la fiabilité du code.
2. **Organisation des tests**
    - **Pourquoi ?** Réorganiser les tests pour qu'ils soient clairs et faciles à suivre permet de faciliter la maintenance et l'extension des tests à l'avenir.

    - Importance : Une bonne organisation des tests assure une meilleure lisibilité et maintenabilité, facilitant l'ajout de nouveaux cas de test et la correction des erreurs.
3. **Vérification des nouvelles fonctionnalités pour TennisGame2**
    - **Pourquoi ?** Tester les nouvelles fonctionnalités comme l'historique des scores et la réinitialisation du jeu assure que ces ajouts fonctionnent correctement.
    - **Importance : Ces tests garantissent que les nouvelles fonctionnalités sont correctement intégrées et fonctionnelles, améliorant ainsi l'expérience utilisateur et la qualité globale du produit.**