clean code

1. Follow conventions

   ```
   int iMyInteger = 10;

   float fMyFloat = 10.5;

   String SmyString = “hi bitkal”;
   ```

2. Variable scope
   a. private variables are prefixed with an two underscore
      private int __iMyVar = 10;
   b. protected variables are prefixed with an one underscore
      private int _iMyVar = 10;
   c. public variables are left as they would be normally
   d. constant values are in all caps and separated with underscores

      int I_WINDOW_SIZE = 900;

3. whitespace

```
public class ExempleIndentation {

  // Constantes pour la gestion des jours
  public static final int LUNDI = 1;
  public static final int MARDI = 2;
  public static final int MERCREDI = 3;

  // Méthode principale
  public static void main(String[] args) {
    int jour = LUNDI;

    // Vérification du jour de la semaine
    switch (jour) {
      case LUNDI:
        System.out.println("C'est lundi.");
        break;
      case MARDI:
        System.out.println("C'est mardi.");
        break;
```

```
            case MERCREDI:
                System.out.println("C'est mercredi.");
                break;
            default:
                System.out.println("Jour inconnu.");
                break;
        }

        // Appel à la méthode pour afficher un message
        afficherMessage("Bonjour, tout le monde !");
    }

    // Méthode pour afficher un message
    public static void afficherMessage(String message) {
        if (message == null || message.isEmpty()) {
            System.out.println("Le message est vide.");
        } else {
            System.out.println(message);
        }
    }

    // Méthode pour ajouter deux nombres
    public static int additionner(int a, int b) {
        return a + b;
    }
}
```

4. Keep it Small
   You should design and implement small, focused components that serve a specific purpose, rather than large, monolithic components that try to do everything. This can help to improve the maintainability and scalability of the system by making it easier to understand, test, and modify individual components.
5. Use meaningful names
   https://workat.tech/machine-coding/tutorial/writing-meaningful-variable-names-clean-code-za4m83tiesy0

git

1. Format standard des commentaires de git

<type>(<scope>): <titre résumé du commit>
<Description détaillée si nécessaire>
Refs: <id>

✅ Types de commit courants (Git Conventional Commits)

Type	Signification
`feat`	Ajout d'une nouvelle fonctionnalité
`fix`	Correction d'un bug
`docs`	Mise à jour de la documentation
`style`	Changement de mise en forme (indentation, espaces...)
`refactor`	Refactorisation du code sans changement de fonctionnalité
`test`	Ajout/modification de tests
`chore`	Maintenance ou tâches diverses

Exemple

feat(login): ajout de la connexion OAuth
- Ajout du bouton de connexion Google
- Implémentation du callback OAuth
- Gestion des erreurs de connexion

Refs: #123

2. Format standard des Branch de git

<Type>/<prenom>/<id>- <description-courte>

Exemple:

feat/khalid/101-user-dashboard
fix/abderahman/202-fix-404-error