CHEAT SHEET

Meaningful Variable Names:

Use descriptive and meaningful names for variables, functions, and classes. Aim for clarity and avoid abbreviations that may lead to confusion.

Single Responsibility Principle :

Ensure that each class or function has a single responsibility. This promotes code modularity and makes it easier to understand and maintain.

Consistent Code Formatting:

Maintain a consistent code formatting style throughout the project. Consistency improves readability and collaboration among team members.

Avoid Magic Numbers:

Replace magic numbers in your code with named constants or enums. This enhances code readability and makes it easier to understand the purpose of specific values.

Comments for Intent, Not Mechanics:

Use comments to explain the intent behind code rather than explaining the mechanics. Well-written code should be self-explanatory, and comments should provide insights into why something is done, not how.

Test-Driven Development (TDD):

Embrace TDD principles to write tests before implementing functionality. This ensures that your code meets requirements and is more resistant to bugs.

Refactor Regularly:

Refactor your code continuously to improve its design and maintainability. Refactoring should be an ongoing process rather than a one-time activity.

Avoid Long Functions:

Keep functions short and focused. If a function is becoming too long, consider breaking it into smaller, more manageable functions with distinct responsibilities.

Use Version Control Effectively:

Leverage version control systems (e.g., Git) to track changes, collaborate with team members, and roll back to previous states when necessary. Commit frequently and write meaningful commit messages.

Code Reviews:

Engage in regular code reviews with your team. Code reviews provide an opportunity for knowledge sharing, catch potential issues early, and ensure that coding standards are maintained.