CheatSheet

1. **Naming Conventions**

- Names have to reflect what a variable, field, property stands for.
- Names have to be precise.
- Names have to reflect the entire functionality

2.Exception Handling

- Exceptions should be as precise as possible. Only pay attention to the exceptions to which you have a relevant response.
- Only catch exceptions if you have the ability to respond meaningfully. Otherwise, wait for someone up in the call stack to respond.

3.Dependencies

- If one module relies on another, the relationship should be physical rather than intellectual. Make no assumptions.
- If the order of some methodcalls, for example, is significant, make sure they can't be called in the wrong order.

4.Environment

- With a single command, check out and then build
- With a single command, run all unit tests.
- Warnings, errors, and exception handling will catch you if you try to override them.

5.Don't Assume

- It's not enough to just work; you must also comprehend why it works.
- Boundaries should always be unit tested. Don't make assumptions about people's actions.

6.**Don t forget**

- It's usually preferable to keep things simple. As much as possible, keep things simple.
- Always try to find the source of an issue. Otherwise, it will seize you.
- Check guidelines for coding, architecture, and design

7.Review code

- Review your code line by line
- Try to consider suggestions and improve it
- Use integrated tools to get update on your code quality

8.Things to avoid

- Delete anything that isn't in use. They'll be in your version control system.
- Code comments should only be used for technical remarks.

9.Source Code Structure

- When compared to unnested code, nested code should be more precise or handle fewer possibilities.
- Organize everything that belongs to the same feature. Namespaces should not be used to communicate between levels. A feature may make use of another feature; for example, a business feature may make use of a core feature such as logging.

10.Single Responsibility Principle

- One function ,class or module should serve one reason.
- Refactor if it is not the case.

11.**Functions**

Functions should be small

Functions should do one thing

Methods Should Descend 1 Level of Abstraction.