**Agile Assignment-1**

**By**

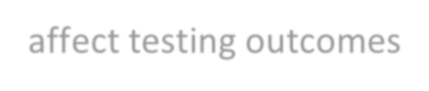
**ABHISHEK SHARMA**

**R171218005**

**500067644**

**Q.1 What are various refactoring rules? Explain.**

Refactoring rules:-

* + **Try refactoring automation:** As with most processes, the more it can be automated, the easier and faster refactoring becomes.
  + **Test often:** Make sure you have proper tests in place before starting any refactoring project.
  + **Refactor first before adding any new features: I**t will take longer to finish the project, but it will also reduce the amount of technical debt you or the product owner will have to deal with in the future.
  + **Focus on Progress, not Perfection:** You have to start thinking about refactoring as an ongoing maintenance project. Just like you have to clean and organize your home throughout the week, you will need to clean and organize your code on many different occasions.
  + **Plan your refactoring project and timeline carefully:** The most important outcome of refactoring is that not only is the code cleaner but that it actually works. It’s going to take longer than you think, so plan accordingly and give yourself a little extra cushion of time.
  + **Get your QA team involved:** get your QA and testing team involved in the refactoring process. Whenever you’re making changes to existing code, even as a clean-up project, it can [affect testing outcomes.](https://www.altexsoft.com/blog/engineering/8-ways-to-improve-software-testing-through-planning-work-environment-automated-testing-and-reporting/)

**Q.2 What are various clean code practices ?**

Various clean code practices:-

* + **Write Meaningful Names for Variables and Functions:** Declare proper names for variables, functions, classes etc so that function and class names are meaningful which means they should be purposeful, substantial, useful, and worthwhile.
  + **Methods and Functions:** Methods and functions should fulfil only one task at a time. The single responsibility principle is mission-critical in software development.
  + **Comments:** Sometimes your code might require a comment, especially when it performs some specific task, or the method intention cannot be understood by its name only. In this case, write a single line and don’t go into too much detail.
  + **Code Formatting**: Formatting and readability have a direct influence on keeping the code up to date and can help you save tons of time in the long run.
  + **Unit Test**: Unit tests will help to detect weak spots and identify where the behaviour of the program is distorted or broken.
  + **Refactoring**: This should be done carefully and in small parts, so you don’t mess up one part while trying to fix another. Step by step, go through every single line of code in order to improve your code.

**Q.3 What are various rules of TDD?**

Rules of TDD:-

• Rules by Kent Beck:-

1. Don't compose a line of new code except if you initially have a faltering computerized test.
2. Eliminate duplication.

• Rules by Uncle Bob:-

1. Don't compose any creation code except if it is to make a faltering unit test pass.
2. Don't compose anything else of a unit test that is adequate to fizzle, and assemblage disappointments are disappointments.
3. Don't compose any more creation code that is adequate to pass the one bombing the unit test.

• Rules by Martin Fowler:-

1. Write a test for the following piece of usefulness you need to add.
2. Write the utilitarian code until the test passes.
3. Refactor both new and old code to make it all around organized.

**Q.4 Differentiate Lean, Scrum and Kanban**

Differences are as follows:-

* + **Lean approach:** A lean organization understands customer value and focuses its key processes to continuously increase it. The ultimate goal is to provide perfect value to the customer through a perfect value creation process that has zero waste.
  + **Scrum approach:** Scrum is an agile way to manage a project, usually software development. Agile software development with Scrum is often perceived as a methodology; but rather than viewing Scrum as methodology, think of it as a framework for managing a process.
  + **Kanban approach:** Kanban is an agile methodology that is not necessarily iterative. Kanban allows the software be developed in one large development cycle.

**Q.5 What is the role of Kanban in Lean Manufacturing?**

Kanban has a role in lean manufacturing because it eliminates labour and inventory waste. One way that Kanban reduces waste is through a pull production model that leads to item production based on consumer’s supply and demand. Kanban produces items in direct relation to the number requested by the market.

**Q.6 What are the never-ending conflicts between Developers and IT Operations?**

At the end of the day, the developers and IT operators are taking a stab at something very similar: to make the association as beneficial as could reasonably be expected. Notwithstanding their comparative destinations, however, it's anything but difficult to perceive how these clashing jobs can impede one another. Engineers are attempting to make and improve applications as fast as could reasonably be expected, and tasks is doing all that it can to keep changes from happening in the climate. Something needs to provide all together for the association to work viably and productively.