Module 8: Group Assignment

# JMeter load testing on To Do Web UI:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

# Code smell table:

|  |  |  |
| --- | --- | --- |
| Name | Category | Severity (1-10) |
| Data Class | Class | 1 |
| Dead Code | Logic | 3 |
| Temporary Fields | Class | 4 |
| Switch Statements | Class | 5 |
| Feature Envy | Class | 6 |
| Long Method | Class | 7 |
| Divergent Change | Architectural | 7 |
| Shotgun Surgery | Application | 8 |
| Large Class | Class | 8 |
| Duplicate Code | Application | 8 |

|  |  |  |
| --- | --- | --- |
| Name | Category | Severity (1-10) |
| Wolf Tickets | Project Management | 4 |
| Dead End | Application | 5 |
| Corncob | Code | 6 |
| Jumble | Code | 6 |
| Stovepipe System | Architectural | 7 |
| Golden Hammer | Developer | 7 |
| Analysis Paralysis | Project Management | 7 |
| Cut and Paste | Application | 8 |
| Death by Planning | Project Management | 8 |
| Lava Flow | Project Management | 8 |
| Boat Anchor | Architectural | 9 |

# How are code maintainability and simplicity related?

Code maintenance and simplicity are related since having the latter will make the former easier to do. Simpler code is easier for developers to understand because developers will have an easier time deciphering what the code does. When fixing any issues that arise or adding new features, developers will have a better understanding of how the program works and know that they can maintain the software without the fear of breaking everything.

Simple code is the opposite of spaghetti code by being organized in a modular way. By breaking down the code into smaller modules, it becomes easier to maintain since the maintainer can take their time to understand how the code works bit by bit instead of having to rely on other developers with experience. This also decreases the maintainer’s reliance on documentation which may be poor if the code is complicated. Any changes made to maintain the code won’t require extensive new documentation since the simplicity of the code will make it speak for itself. Modular code can also be restructured by having modules swapped in and out without breaking the entire program. This reduces the work and time needed to maintain the software in the long run.

Another advantage to writing simpler code is that it makes debugging easier. Unit testing is easier to implement as it’s recommended that code be structured in a simple and modular way so that chunks of code can be tested in isolation. This makes maintaining the software easier since unit testing helps the developers quickly pinpoint if any changes to the code produce unintended results.

# Runtime efficiency and code simplicity are often competing goals. How can you deal with this problem? Is it possible to have code that is both simple and efficient?

A way to solve the problem of choosing performance over simplicity is to review what the software requirements are and prioritize the method that makes the most sense for the given application. If performance isn’t a high priority, then it makes more sense to prioritize simple code as it’s easier to maintain in the long run due to it being easier to understand and implement.

If balance is necessary, then it’s best to see which sections of code take up most of the runtime and only optimize those chunks. To find these performance bottlenecks, developers can use profiling tools like JMeter to find the offending sections of code and see exactly how much runtime they take up. It makes the most sense to optimize only the problematic sections of code since they take up most of the runtime. This way, a majority of the project can retain a simple and easy-to-understand structure, which will make it more bearable for future developers to deal with the small section of complex code.

One way to have both simple and efficient code is by using existing code libraries. These libraries contain methods that can do complex code operations without the developer having to reinvent the wheel to do it themselves. Since the complex logic is hidden from the developer, it makes the code simpler to maintain since the developer doesn’t have to worry about inadvertently changing the inner workings of these methods. Also, if the problem in question has been solved before, then it is likely that a library contains a more efficient way of solving the problem than whatever the developer came up with themselves.