**Git Basics**

​The protégé will have demonstrated an understanding of the following concepts and used them while completing katas:​

|  |  |
| --- | --- |
| **Item** | **Completed** |
| Creating a repository | Yes |
| Cloning a repository | Yes |
| Creating branches | Yes |
| Merging branches | Yes |
| Pushing branches | Yes |
| Creating a pull request | Yes |
| Reviewing a pull request, commenting on, approving, and merging a pull request | Yes |

**Keyboard Confidence**

​The protégé will have demonstrated the ability to type on the keyboard as a way of interacting with the computer. The keyboard should not be seen as a hinderance, or an obstacle, simply one way which you can interact with a computer. Touch typing is not a requirement.

|  |  |
| --- | --- |
| **Item** | **Completed** |
| Has demonstrated confidence in using the keyboard | Yes |

**Editor Confidence**

​The protégé should choose an editor and become efficient using it. It would be great to be familiar with a second editor, but efficient use is not a requirement.

|  |  |
| --- | --- |
| **Item** | **Completed** |
| Can use keyboard shortcuts, and knows how to find them for commands | Yes |
| Can use the refactor (rename) function to rename variables, method names, classes, etc | Yes |
| Is aware of ways of extending the editor through the use of plugins/extensions or configuration options | Yes |
| Can open the editor from the command line​ | Yes |

**Unix Command Line**

​The protégé should be comfortable using the command line and will have demonstrated this through the following:

|  |  |
| --- | --- |
| **Item** | **Completed** |
| Has completed the four checkpoints in the Unix Command Line page in General Developer repo |  |
| Has written two different bash scripts to help with automating repeated operations |  |
| Has customised their prompt on their computer​ |  |

**Pair Programming**

​The protégé should have demonstrated their understanding of the different ways of pair programming and will have practiced this technique with different people.

|  |  |
| --- | --- |
| **Item** | **Completed** |
| Can describe the different kinds of pair programming techniques |  |
| Has participated in at least 4 sessions of pair-programming |  |

**Foundational Programming**

​The protégé should be aware of the following concepts, be able to describe what they are, and have used them while completing katas. These are all to be demonstrated in C# (the language of choice for Protégés):​

|  |  |
| --- | --- |
| **Item** | **Completed** |
| Variables & Types | Yes |
| If / Else & Case Statements | Yes |
| Arrays |  |
| Methods & Functions |  |
| Collections (List, Dictionary/HashSet) |  |
| Looping |  |
| Enums |  |
| Classes - Public / Private / Protected Methods |  |
| Classes - Static methods vs Instance methods |  |
| Classes - Constructor & passing values through a constructor |  |
| Classes - Private / Protected Variables |  |
| Inheritance with classes (and why it should usually be avoided) |  |
| Composition with classes (why this is usually better than inheritance) |  |
| Interfaces |  |
| Exceptions |  |
| Generics |  |

**Language Design**

**Four Rules of Simple Design**

​Based on Kent Beck's recommendation there are four simple rules which if you follow them will help you build good quality software. The protégé should be able to explain and demonstrate use of the following concepts:

|  |  |
| --- | --- |
| **Item** | **Completed** |
| Code Passes the Tests |  |
| Reveals Intention - it is easy to understand what the program is doing by reading the code |  |
| No Duplication - do not repeat yourself |  |
| Fewest elements - do not write more than you need to​The protégé should strive for Level 3 which is defined as:​  Classes, methods and variable names are named to reveal mainly technical concepts with occasional naming revealing specific domain examples. Comments are very rarely used.​Repeatable patterns are identified based on code duplication. Duplicate code patterns are consistently pulled out into supporting methods or classes.​ Classes and methods are broken up into small components with clear responsibilities. Methods are consistently kept small.​ |  |
| Has achieved Level 3​ |  |

**Test Driven Development**

​The protégé should follow the guidelines of Test Driven Development in completing the katas. The protégé should strive to achieve at least Level 3 which is defined as:​

The basic TDD cycle is followed where in general each time new functionality is added to the solution an appropriate test is written first.

|  |  |
| --- | --- |
| **Item** | **Completed** |
| Has achieved Level 3 |  |

**Test Doubles**

​The protégé should use where appropriate Test Doubles when testing their code. The protégé should strive to achieve at least Level 3 which is defined as:​

Leverages a test framework to utilize out of the box test doubles. Can articulate what type of test double the test framework is providing for different test scenarios. Able to swap out parts of the solution code base with appropriate test double framework implementations to isolate behaviour for testing.

|  |  |
| --- | --- |
| **Item** | **Completed** |
| Has achieved Level 3 |  |

**You Ain't Gonna Need It - YAGNI**

​The protégé should demonstrate that they understand the YAGNI concept and use it when working on the katas being worked on. The protégé should strive to achieve at least Level 3 which is defined as:​

Generally applies YAGNI principles when solving problems. Delays implementing code capabilities to support presumptive features. Solution design focusses on solving the problem at hand without gold-plating.​

|  |  |
| --- | --- |
| **Item** | **Completed** |
| Has achieved Level 3 |  |

**Command Query Separation**

​The protégé should understand and demonstrate usage of the Command Query Separation concept. The protégé should be able to explain it to other Proteges. The protégé should strive for at least Level 3 which is defined as:​

Generally applies command query separation when writing methods.

|  |  |
| --- | --- |
| **Item** | **Completed** |
| Has achieved Level 3 |  |

**Object Composition**

​The protégé should demonstrate their knowledge of object oriented programming, and be able to explain the differences between composition and inheritance, and where you should consider using either approach. The protégé should prefer object composition when solving katas. The protégé should strive for Level 3 which is defined as:​

Can implement object composition in coding problems. Consistently applies composition over inheritance in code.

|  |  |
| --- | --- |
| **Item** | **Completed** |
| Has achieved Level 3 |  |

**Whiteboarding**

​Being able to sketch out designs or diagrams to explain how code or systems work is an essential skill. You don't need to know [UML](https://en.wikipedia.org/wiki/Unified_Modeling_Language) but you should be able to sketch out diagrams which will aid in explaining systems, code, or user interfaces. The protégé should be able to demonstrate the solution of 2 katas by use of whiteboard or the equivalent whiteboard tools.

|  |  |
| --- | --- |
| **Item** | **Completed** |
| Used a whiteboard (real or virtual) as an aid to explain 2 solutions |  |