Ryan Porter

CSC 260

Mr. Jenkins

5/2/21

Final Project Write-Up

For completeness, I have this document, and in my code, I have these classes: Board, Custom, Easy, Expert, IDifficulty, Intermediate, Cell, Mine, Number, Grid, HighScores, and Timer. This makes 12 classes. Also, my code should be fully functional and complete.

For complexity, I had to create forms in my code (HighScore class, lines 99-131; Custom class, lines 81-123). Also, I used LINQ in the Cell (lines 50-52, 58-60), and Grid (lines 67-69, 80-82) classes. Other complex things that I did was create a dynamic list of cell buttons (in Grid and Board classes), have right and left clicking available (Board class), and having different difficulties.

For version control, I used GitHub. I created a repository and linked it to my Visual Studio. Each time a part of my code worked (game won, game lost, etc.), I would commit and push. This allowed for version control. My repository should also contain my UML diagram, a picture of it, this document, and my testing document.

For my UML diagram, I completed it before I started and my code and used it to generate an outline of my code. I had most of my classes already in there, but during coding, I added the Custom class and some methods that will not be in the UML diagram.

For documented test cases and testing, I included a document in my repository and when I turned it in. I tested for many different things across a couple of weeks to make sure everything was working. I also had someone else test it to make sure I did not miss anything.

For classes and their components, I already stated my 12 classes. In most of my classes, I also used constructors, class methods (HighScore class has this), instance fields, and properties.

For inheritance, my Cell abstract class inherited from the Button class so that I could get functionality of a button in that class. My Mine and Number class inherited from that Cell class. This probably doesn’t count, but my Board class inherited from the Form class.

For encapsulation, I used different access modifiers where possible. I used private modifier when only the class would use the method, or I used the internal, which is not quite as open as public. Also, I used getter and setters in most of my classes. In the Custom difficulty class, I used getters and setters to validate the data that was coming in.

For polymorphism, I overrode the RevealClick() abstract method in my Cell class when I did my Mine and Number classes. Otherwise, polymorphism was not needed much in my project because most of my classes did not override methods within them.

For abstraction, I created the Cell abstract class which has the RevealClick() abstract class.

For my first advanced topic, I used interfaces for my difficulty. I created a difficulty interface (IDifficulty) and let my Easy, Expert, Intermediate, and Custom class implement this. This made it easy to check if my classes where implementing each difficulty correctly.

For my second advanced topic, I used LINQ a few times in my code, which made it a lot easier to implement. I listed these in my second paragraph.