Code Review – Peter Pasieka

Week 4

|  |  |
| --- | --- |
| **Code Formatting** | **Comments** |
| Indentation:   * Use of white space * Start and end point of code block easily identifiable | Good use of white space  Is identifiable |
| Naming Conventions:   * Meaningful names for classes, methods and variables   + Intention revealing names   + One word per concept * Use of case   + Variables = lowerCamel   + Classes = CamelCase   + Methods = lowerCamel   + Functions = lowerCamel | Good naming conventions  Good use of case |

|  |  |
| --- | --- |
| **Best Practices** | **Comments** |
| Comments:   * Class/files headers * Methods * Why, not what? | Class headers are there but no comments on methods |
| No hard coding | No hard coding |

|  |  |
| --- | --- |
| **Non Functionality** | **Comments** |
| Maintainability:   * Readability   + Class: Variables, Constructor, Methods   + Logical order of methods * Understandability * Consistancy | Good use of maintainability |
| Reusability   * Use of classes * Reusability of functionality * Avoid repetition of code – DRY (Do Not Repeat Yourself) * Functions should be small * Functions should do one thing * Classes should be small | Good use of reusability |
| Easy to test | Easy to test |

|  |  |
| --- | --- |
| **Fundamentals** |  |
| Use of frameworks (ie java.util.\*) | No frameworks were used |
| Use of encapsulation   * Getters and Setters | Used getters and setters |
| Use of constructor overloading as appropriate | Used constructor overloading |
| Similar values grouped (Enum) | No enum |
| Input validation implemented | No need for input validation |
| Use try-catch loop for user input to program (Exception Handling) | Used a try-catch loop |

|  |  |
| --- | --- |
| **Functionality** | **Comments** |
| Does it meet the requirements? | Does not meet requirements as hour doesn’t tick over, easy to fix |
| Does it work? | Technically works, again hour doesn’t increment. |