## Assignment #1 – OOP Design, Refactoring, and Testing Practice

Student Solution/Hand-In Document

Jackson Glover

### Step 1: Problem Identification

This piece of code has multiple serious errors within it. To begin, all the code has been centralized into one file which does not follow single responsibility principles. Secondly, throughout this code there is multiple incorrectly assigned values, these are values that are potentially nullable being assigned to non-nullable variables. There is multiple compiler warnings trigger on compile, however these warnings have been ignored, specifically on lines: 62, 63. Additionally, throughout the code there is multiple variables that are assigned magic numbers upon initialization, these values should be implemented using proper CONSTANT style formatting & using global declarations. Furthermore, throughout the code there is multiple repeating lines of code that could be simplified using stronger enumeration techniques, specifically lines 62, 63, 101, 102, 103. Finally, there is no separated interface driver for this software, resulting in the PlazaController to oversee multiple responsibilities.

Because of the mistakes above it is extremely safe to say that single responsibility was not adhered to, quite clearly. Additionally, because of how interlaced all of the classes are the open-closed principle also wasn’t adhered to. Because this program does not really use polymorphism Liskov’s substitution principle was adhered to. As mentioned multiple times above, the interface segregation principle was not adhered to causing a jumbled mess of function calls. The functions are mostly independent in terms of dependency, obviously the primary handler uses all the classes, but the subclasses themselves are independent. Therefore it is safe to say that the Dependency inversion principle was followed loosely.

### Step 2: Requirements Definition

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Requirement | Description | Notes |
| R1 | Interface | The interface shall be provided as a driver file which all other functions should adhere to. Functions shall not be allowed to manage their own output without using the driver. | Should ensure singularity of the interface |
| R2 | Dependency Inversion | Classes shall only have one-way dependencies; Classes should be able to be tested in layers | Should ensures the classes are easy to test |
| R3 | Global Storage | Global storage shall be handled with a separate driver class. Functions should adhere to the drivers and never manage the ghruulobal storage directly. | Should ensure that global storage is traceable & easy to understand |
| R4 | Constants | All constant values or init values that are a specific value shall be defined as a global constant with a clear & defined name. If required, they should also include a comment above their definition. | Should ensure there is no magic numbers throughout the code |
| R5 | Execution time | The execution time of the software shall be below 5 seconds. This should be done by optimizing repeated code & memory management | Should ensure timely execution |

### Step 3: Revised Design

<paste your new UML Diagram here>

### Step 4: Testing Strategy

// usage of the Poll method

Create a charger packet object

Create a randomly assigned string

Create a check for occupied, fault, draw, price (assume these are not randomized)

Send a poll using the charger packet object

Check the return object for all checks of occupied, fault, draw.

// usage of global storage driver & displaying the data with the interface driver

Init the global storage driver

Init the display driver

Identify an expected string in the output

Setup the StringWriter to redirect the output

Pre-populate the global storage with set values

Run the dumplog method from the interface driver

Ensure the StringWriter captured the expected output