Research Project- Project Report

Nicholas Jumpp- 1501298

Kimone V. Roper Ist -1403541

University of Technology Jamaica

S.O.L.I.D. Principles

“Object-Oriented type of programming brought a new design to software development. This enables developers to combine data with the same purpose/functionality in one class to deal with the sole purpose there, regardless of the entire application.” (Nnamdi, 2018).

The five principles are as follows:

* S – Single Responsibility Principle (SRP)
* O – Open Closed Principle (OCP)
* L – Liskov Substitution Principle (LSP)
* I – Interface Segregation Principle (ISP)
* D – Dependency Inversion Principle (DIP)

The Single Responsibility Principle (SRP) states that “A class should have one, and only one job.”(Barber, 2015)

* The SRP is seen in the adminlogin.js file as its only job is to verify a user for successful login.

The Open Closed Principle (OCP) states that “You should be able to extend a classes behavior, without modifying.”(Barber, 2015)

* OCP was is seen in the cart.js file as to which for the printCart finction is open to to extensions but closed for modifications. So it is extended and used by other functions that will call it but its base is still the same and is not modified

The Liskov Substitution Principle (LSP) states that “Derived classes must be substitutable for their base classes.”( Barber, 2015)

The Interface Segregation Principle (ISP) states “Make fine grained interfaces that are client specific.”( Barber, 2015)

* ISP was not implemented as to that interfaces are not able to be created due to there is no built in support for traditional abstraction in JavaScript.

The Dependency Inversion Principle (DIP) states “Depend on abstraction, not on concretions.”( Barber, 2015)

* This is seen in buttons.js file in which any card that has the name “btnsnackitem” on the homepage will be grabbed and stored along with it’s id name. What this does is to help make it less concrete in which coding for each snack on page it will just look for all snacks that are on the page and store them in a variable.

MVC Pattern

MVC Pattern stands for Model-View-Controller Pattern. This pattern is used to separate application's concerns. (Tutorialspoint.com)

* Model - Model represents an object carrying data. It can also have logic to update controller if its data changes.
* View - View represents the visualization of the data that model contains.
* Controller - Controller acts on both model and view. It controls the data flow into model object and updates the view whenever data changes. It keeps view and model separate.

Model

The storage medium used in this research project is localStorage that is within modern browsers which stores necessary information from the user to be either presented back to them or to the manager or delivery man.

View

In our research project the view is our UI presented to the customer and manager/ delivery man of John Shop which was coded using HTML & CSS for styling.

Controller

The controller works in conjunction with the model and the view as to update the page as necessary if there has been any changes made from the view portion or from the model portion to the view. This is seen when a customer selects an item from the home page by clicking on the add to cart button on any particular item card. Upon selection an alert pops up for successful addition to customer cart which simply means the item has been added to the localStorage of the browser by the controller that is the buttos.js file. A controller is also implemented for the cart and admin page. The cart controller simply gets the items that was added to localStorage and prints it back on the page in a tabular format on the view. Upon completion of an order a list of orders object is created and stored in localStorage. This list of orders objects is to store the different orders made and to printed to the view.

Repository Pattern

The Repository Pattern restricts us to work directly with the data in the application and creates new layers for database operations, business logic, and the application’s UI. The button.js file utilizes the repository pattern this can be seen as to it adding a business logic for the customer which is adding a snack to the cart and having it added to the localStorae (database).

Singleton Pattern

“The singleton pattern is a design pattern that restricts the instantiation of a class to one object” (Singleton, 2018). Singleton is an object which can only be instantiated once. Singleton pattern is seen when creating the listofOrders object only once and there is only one instance of that object. This is seen in the cart.js file in the function listofOrdersFunction(orderObj).

Factory Pattern

“A Factory Pattern says that just define an interface or abstract class for creating an object but let the subclasses decide which class to instantiate” (JavaTPoint, n.d.).

The factory pattern is implemented in the cart.js file in the createOrderObj(studentObj) and createStudentObj(form) function in which it creates the object needed to store the information for both order and student respectively.

References

Barber, A. P. (2015, August 06). SOLID Principles - Five Principles of Object-Oriented Programming and Design. Retrieved from <https://alanbarber.com/post/solid-principles-five-principles-of-objectoriented-pro> gramming-and-design/

JavaTPoint. (n.d.). Factory Method Design Pattern - Javatpoint. Retrieved from https://www.javatpoint.com/factory-method-design-pattern

Nnamdi, C. (2018, October 09). SOLID Principles every Developer Should Know. Retrieved from <https://blog.bitsrc.io/solid-principles-every-developer-should-know-b3bfa96bb68> 8

Singleton Design Pattern | Implementation. (2018, September 10). Retrieved from https://www.geeksforgeeks.org/singleton-design-pattern/

Tutorialspoint.com. (n.d.). Design Patterns MVC Pattern. Retrieved from https://www.tutorialspoint.com/design\_pattern/mvc\_pattern.htm