Advanced Programming

Jonathan Holmes

Jodene Garvey

Lecturer: Gilroy Gordon

University of Technology, Jamaica

Table of Contents

[Singleton Pattern 3](#_3dy6vkm)

[Model View Controller](#_4d34og8) 4

[Factory Pattern](#_2s8eyo1) 5

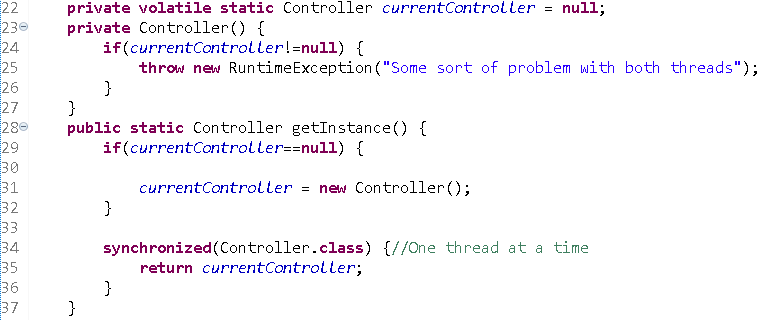
[Source Code Management and Code Generation](#_17dp8vu) 6

[References](#_3rdcrjn) 8

# **Singleton Pattern**

The introduction statement for Singleton Pattern given by the website GeeksforGeeks.org (2018, September 10) states “The singleton pattern is one of the simplest design patterns. Sometimes we need to have only one instance of our class for example a single DB connection shared by multiple objects as creating a separate DB connection for every object may be costly. Similarly, there can be a single configuration manager or error manager in an application that handles all problems instead of creating multiple managers.”

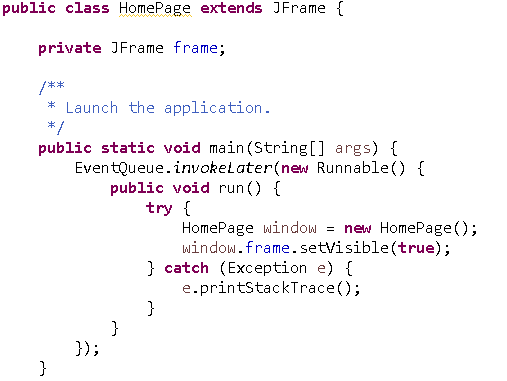
In our code we follow suite using the singleton pattern however we have decided to ensure that the user or anyone does not create multiple controller classes as it could consume much memory, also there is no need for multiple controller classes throughout the lifetime of the program. The following is where we have chosen to implement our Singleton structure.



*Figure #1 displays the snippet of Singleton Implementation of code in Java, protecting for a double thread runtime error.*

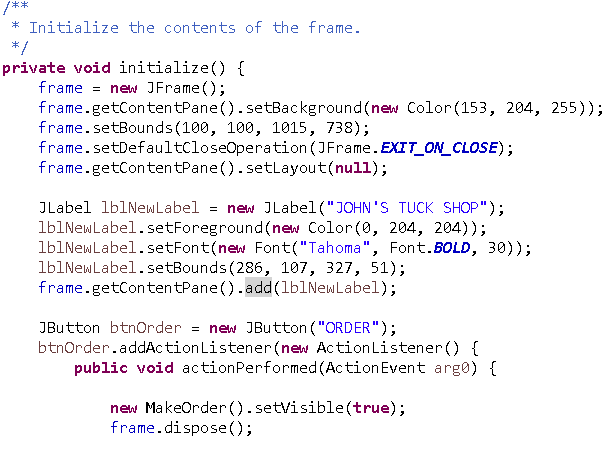
# **Model View Controller**

The purpose of the model-view-controller pattern is to separate concerns of the programmer for easier manipulation of instructions of code and easier ability to remove and inject a new function or a function to replace another without much problems or change of code (Julian J., n.d.). Our Model View Controller for our code is separated as follows: The GUI package represents the view and begins the code as shown in Figure 2.



*Figure 2 shows implementation of JFrame in the class HomePage*

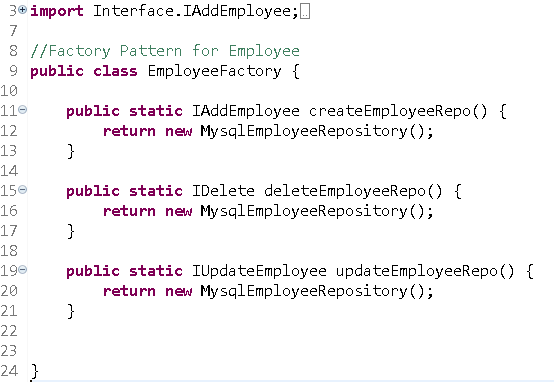
With the understanding of JFrame, the entire frame is made and navigation allowed through event handlers within the frame as shown in Figure 3.

**

*Figure 3 shows the use of the JFrame and shows how the graphic user interface is initialized*

# **Factory Pattern**

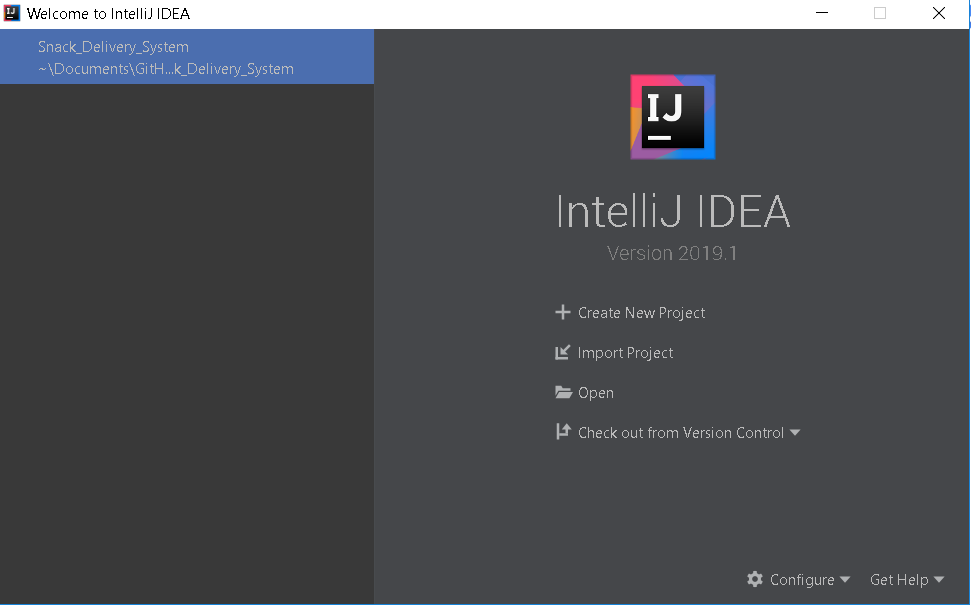
The factory pattern has not been implemented in our code however the purpose of the factory pattern is to ensure that the user or client does not understand the production of the item produced, it is used to create a particular type of item at runtime and return to the abstract class type in which it was constructed: Example: If there are monsters in a game called Goblins and there are two types of them: Big Goblin and Small Goblin. The Big Goblin will have different characteristics than the small goblin and therefore needs its own information to store it. However, when the time comes for the goblin to appear on the screen, if produced with the factory pattern, the user or the hero or main character will not know what goblin will appear on the screen but all they know is that one will appear. However in the library a function can be used to determine if the goblin is big at all times, small at all times, if it is random which one is produced or if it produces one 80% of the time, etc. However we have found that in our code it will not be useful to construct the factory pattern since it produces randomness. Derek Banas (2012, September), a youtuber who creates and explains code and coding concepts has allowed us to understand the utility of Factory Pattern more as well as our facilitator, Gilroy Gordon. Figure 4 includes a snippet of the project implementing the factory pattern using a model class called Employee.



*Figure 4 shows the implementation of the factory pattern*

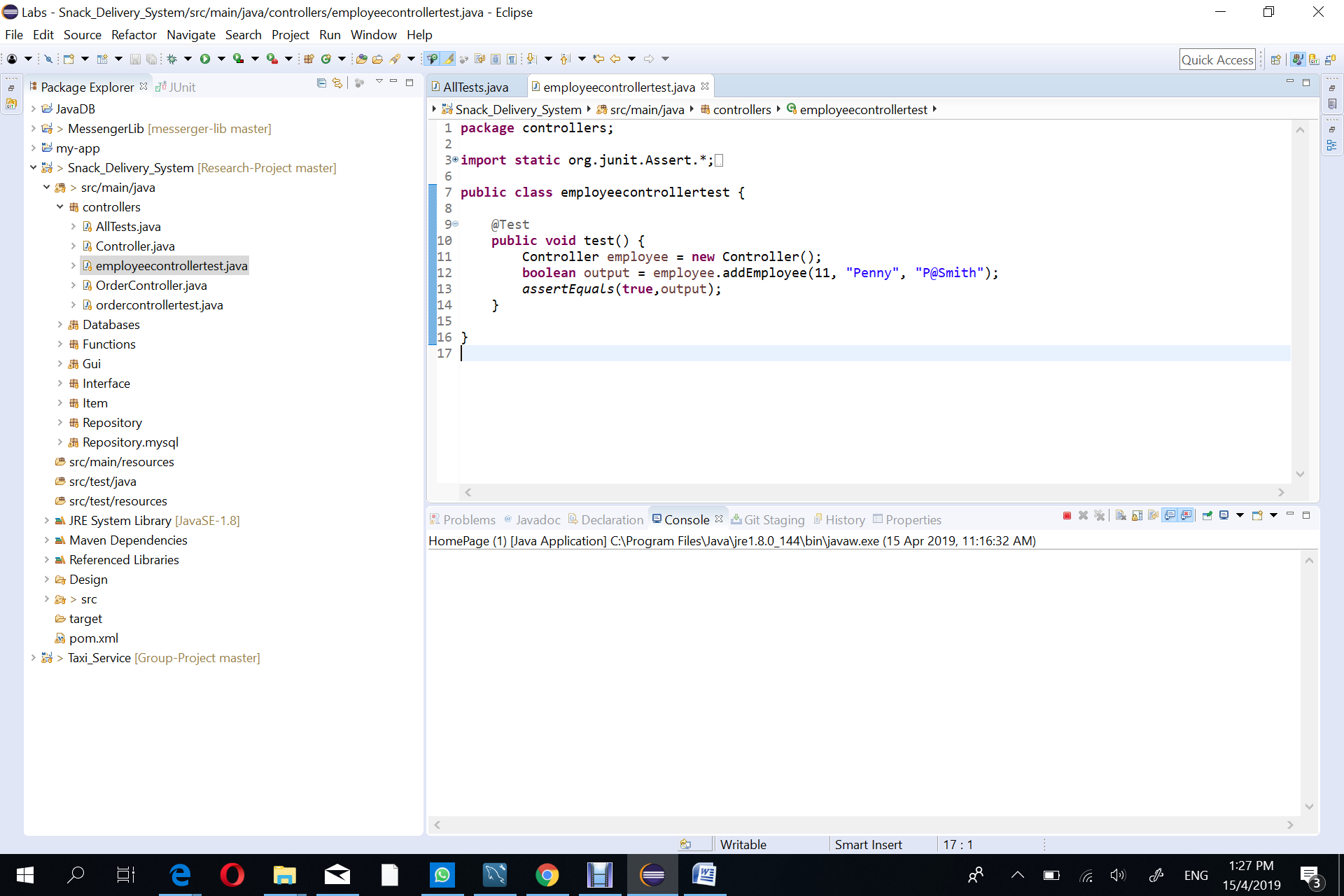
# **Source Code Management and Code Generation**

IntelliJ IDEA application has been used to manage Code Generation and the use of Github and the Gitlab Application has been used to manage of source code on the different computers and well as to continuously integrate our software. JetBrains owns IntelliJ, the program does not belong to this group upon the production of our project or code. It should be kept in mind that these are installable applications to the desktop and to use Gitlab requires a better understanding of the Git enterprise. IntelliJ IDEA has made the production of the code much easier than using the standard Eclipse IDE alone. This project has been used to weigh a comparison between the two as well as to meet the requirements of the Research Project.



*Figure 5 shows the welcome screen for IntelliJ IDEA and proof of the use of the project within it.*

Unit Testing and Test Automation



# **References**

Banas, D. (2012, September 01). Factory Design Pattern. Retrieved April 14, 2019, from https://www.youtube.com/watch?v=ub0DXaeV6hA

Build software better, together. (n.d.). Retrieved April 14, 2019, from https://github.com/about

Download IntelliJ IDEA: The Java IDE for Professional Developers by JetBrains. (n.d.). Retrieved April 13, 2019, from https://www.jetbrains.com/idea/download/

Singleton Design Pattern | Implementation. (2018, September 10). Retrieved April 14, 2019, from https://www.geeksforgeeks.org/singleton-design-pattern/

Tomassetti, G. (2018, May 10). A Guide to Code Generation. Retrieved April 14, 2019, from https://www.javacodegeeks.com/2018/05/a-guide-to-code-generation.html

Tutorialspoint.com. (n.d.). Design Patterns MVC Pattern. Retrieved April 14, 2019, from https://www.tutorialspoint.com/design\_pattern/mvc\_pattern.htm

Jarrett, J. (n.d.). Graphical User Interfaces Pt1. Retrieved April 14, 2019, from University of Technology, Jamaica.

AP Research Project (Snack Delivery System John’s Shop) https://www.youtube.com/watch?v=wjmttcgMc\_k