# Image result for programmingAdvanced Programming CIT 3009

# Research Project : Semester 2 2018/2019

Tutor: Gilroy Gordon

Prepared by:

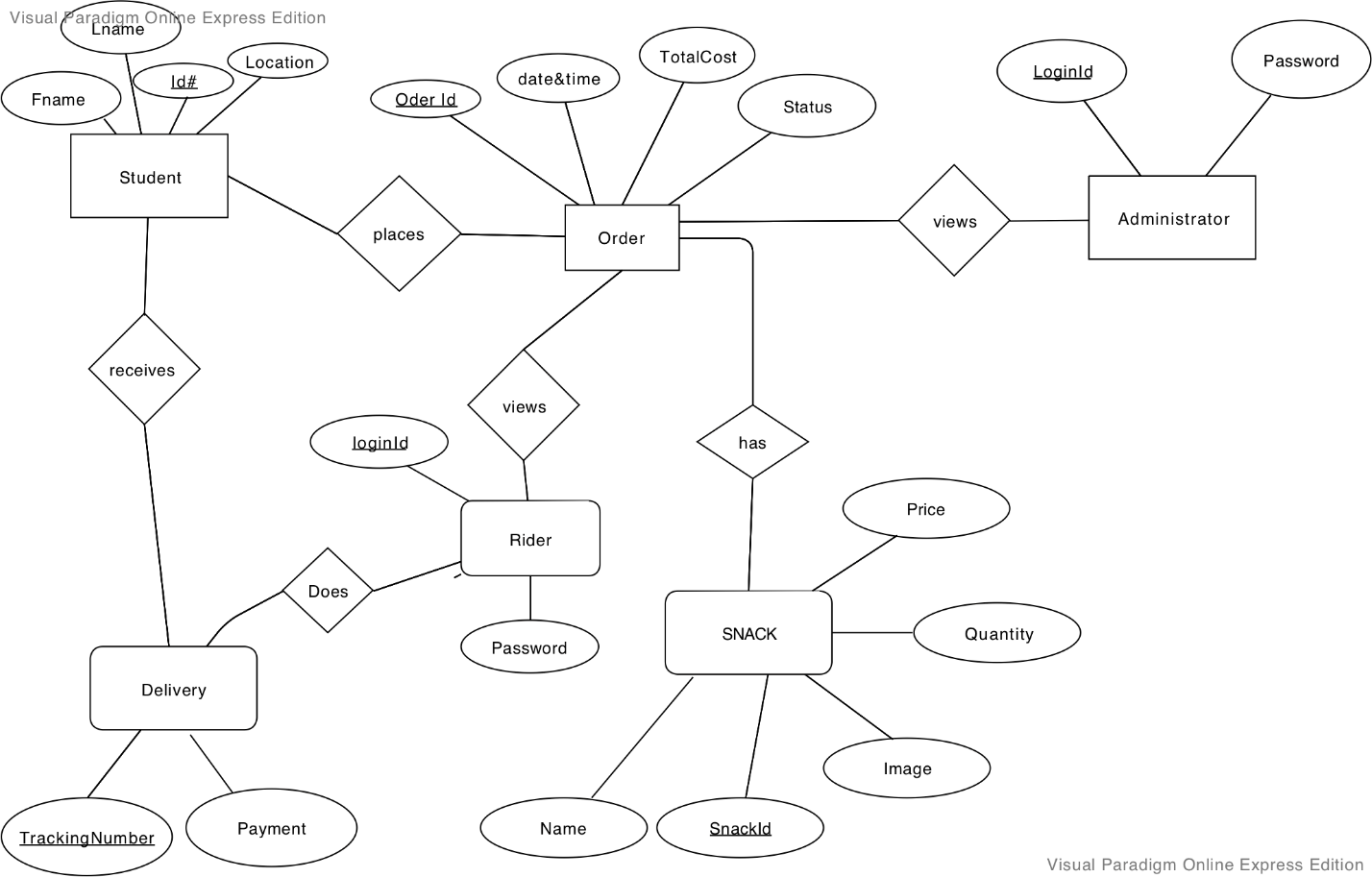
Daniel Pinto – 1306537

Rory Adamson – 0903081

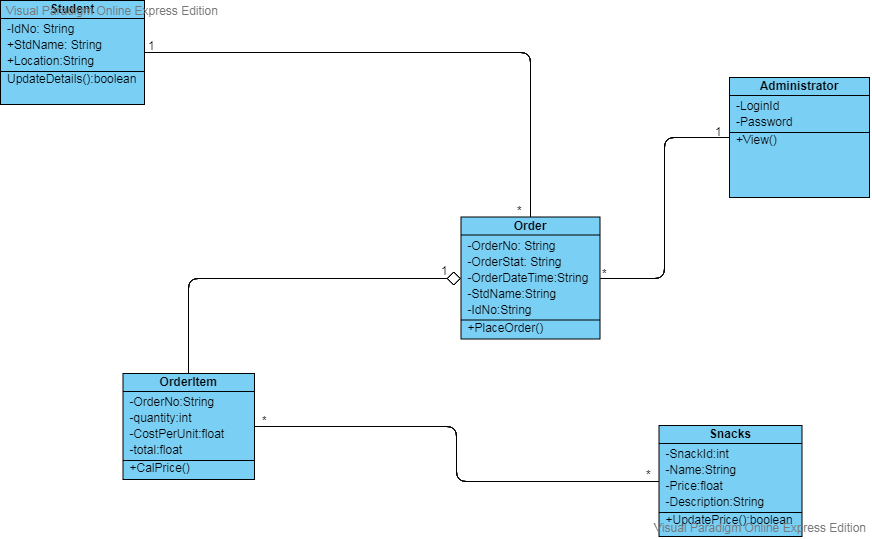
Introduction

This project focuses on a web-based solution for the proprietors of ‘John’s Shop’, which is located on the campus of the University of Technology Jamaica. The solution will create a platform where students of the university who are desirous of purchasing snacks, can do so online and have it delivered to them. The solution will demonstrate the use of the following; S.O.L.I.D principles, Repository Pattern, Model-View-Controller Pattern, Singleton Pattern, Factory Pattern, Code Generation Tool, Source Control Management Tool, Package Management Tool, Unit Testing and Test Automation.

# ERD Diagram



# Class Diagram



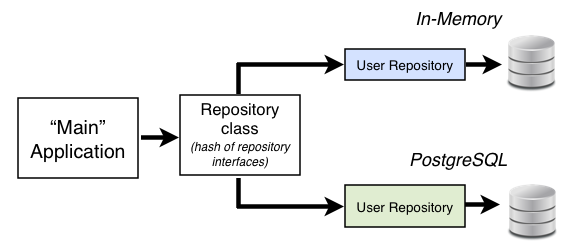
# Definitions and Details

1. **S.O.L.I.D principles**

**S.O.L.I.D** is an acronym for the first five object-oriented design **(**OOD**)** principles by Robert C. Martin. These principles, when combined together, make it easy for a programmer to develop software that are easy to maintain and extend. They also make it easy for developers to avoid code smells, easily refactor code, and are a part of the agile or adaptive software development.

* **S** - Single-responsibility principle: A class should have one and only one reason to change, meaning that a class should have only one job.
* **O** - Open-closed principle: Objects or entities should be open for extension, but closed for modification. This simply means that a class should be easily extendable without modifying the class itself.
* **L** - Liskov substitution principle: every subclass/derived class should be substitutable for their base/parent class. **For example**, Let q(x) be a property provable about objects of x of type T. Then q(y) should be provable for objects y of type S where S is a subtype of T.
* **I** - Interface segregation principle: A client should never be forced to implement an interface that it does not use or clients should not be forced to depend on methods they do not use.
* **D** - Dependency Inversion Principle: Entities must depend on abstractions not on concretions. It states that the high-level module must not depend on the low-level module, but they should depend on abstractions.

1. **Repository Pattern:** Essentially, it provides an abstraction of data, so that your application can work with a simple abstraction that has an interface approximating that of a collection. Adding, removing, updating, and selecting items from this collection is done through a series of straightforward methods, without the need to deal with database concerns like connections, commands, cursors, or readers.

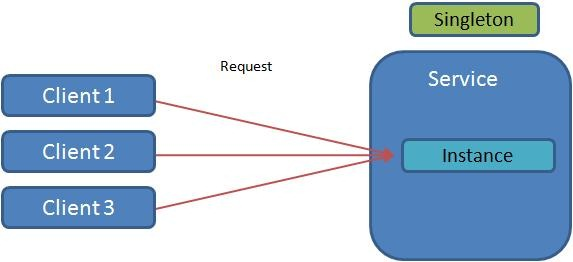
**Fig. 1: repository pattern**

1. **Model View Controller Pattern:** Thisarchitectural pattern used for developing user interfaces divides an application into three interconnected parts. This is done to separate internal representations of information from the ways information is presented to and accepted from the user.

**Fig 2: Model-View-controller**

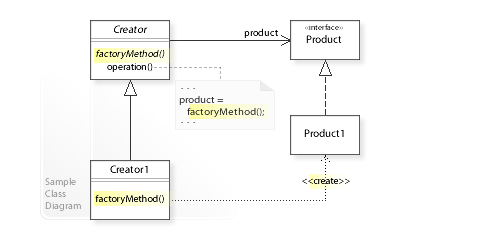


1. **Singleton Pattern** - is a software design **pattern** that restricts the instantiation of a class to one. This is useful when exactly one object is needed to coordinate actions across the system.

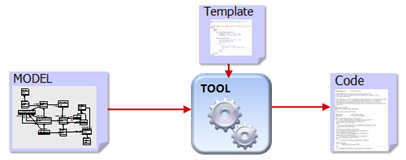


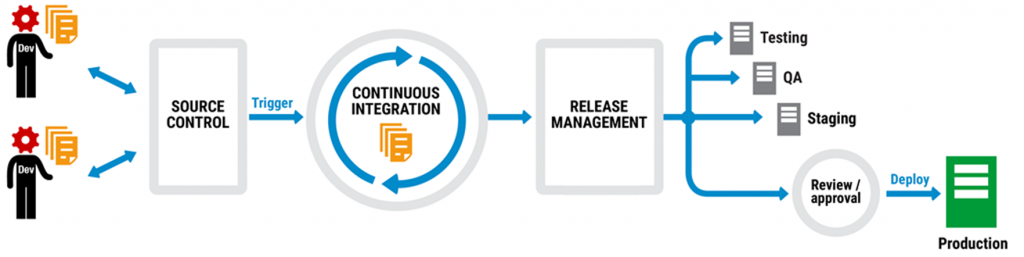
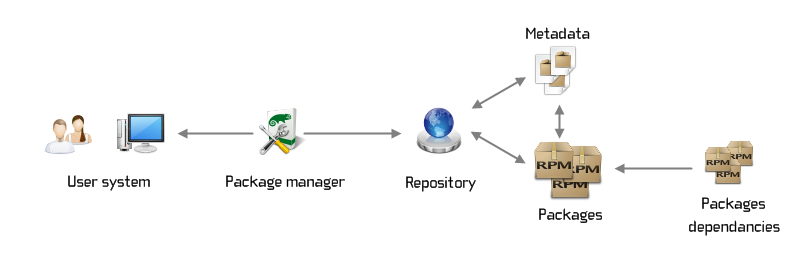
**Fig 3**: Singleton Pattern

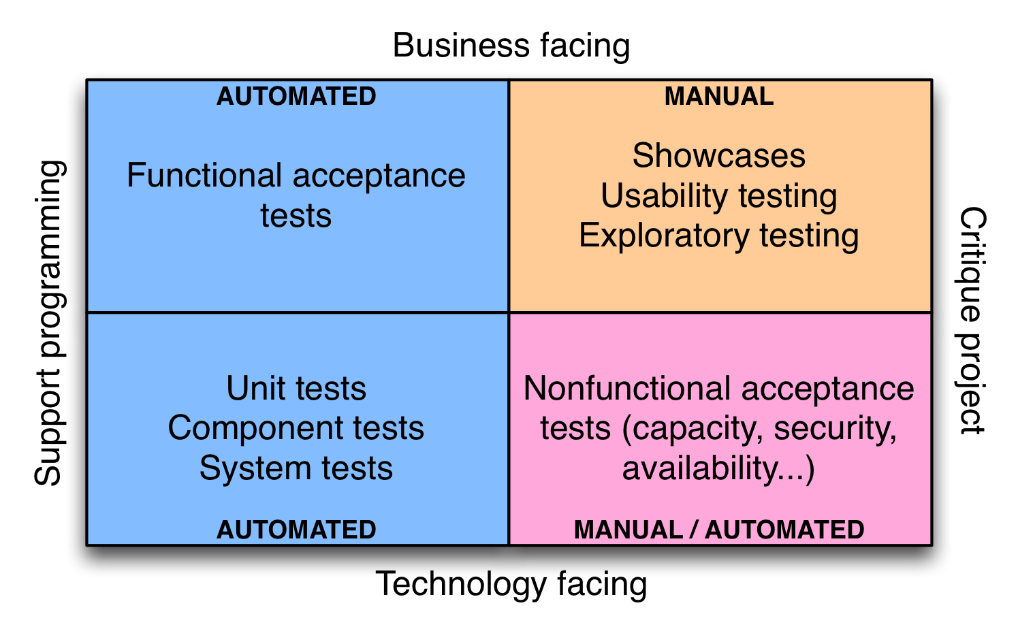
1. **Factory Pattern** - is a creational **pattern** that uses **factory** methods to deal with the problem of creating objects without having to specify the exact class of the object that will be created.

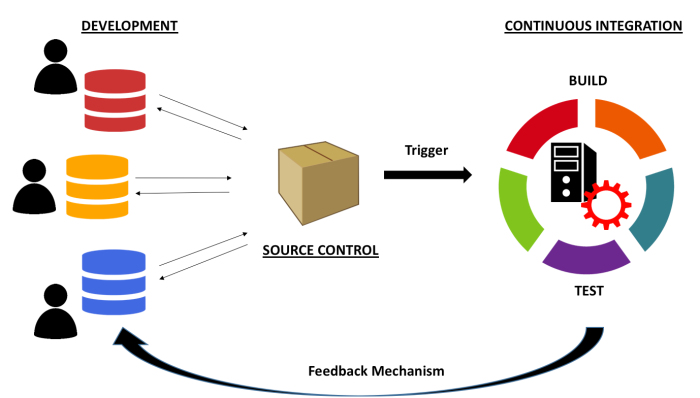
**eFig 4**: Factory Pattern

1. **Code generation tool** - Code generation is a mechanism where a compiler takes the source code as an input and converts it into machine code. This machine code is actually executed by the system.



1. **Source control management tool** - A component of software configuration management, version control, also known as revision control or source control, is the management of changes to documents, computer programs, large web sites, and other collections of information.
2. **Package management tool** - A package manager or package management system is a collection of software tools that automates the process of installing, upgrading, configuring, and removing computer programs for a computer's operating system in a consistent manner.
3. **Unit Testing and Test Automation -** is a method in software testing that makes use of special software tools to control the execution of tests and then compares actual test results with predicted or expected results.



1. **Continuous Integration -** In software engineering, continuous integration is the practice of merging all developer working copies to a shared mainline several times a day .