UML Editor Specifications

**Problem Statement**   
UML is an important tool for creating documents to visualize complex systems with graphical notation. Using UML, a user is able to grasp the basic understanding of various classes and utilities that a program brings to the table, without having to look at a single line of code. UML is widely used in the software engineering business to provide a quick and easy interpretation of a system at a glance. We’ve set out to create a UML editor for the purpose of aiding developers in creating the very documents that would help them in their design process.

**System Personnel**

**Target users**

Ideally, the audience of this program would be other software developers. A person with a background in programming and software engineering and a background in UML would be able to understand the data that is being portrayed in the diagram very easily with little learning curve.

**System developers**

Andrew

Bri

Don

Lukas

Nick

**Operational Setting**

**Target Platforms**

Linux, Mac, Windows

**Software Environment**

Java, as the entirety of the program is written in it, it is required to run it.

**Useful Optional Software Environment**

An image viewer would aid in reading the UML documents that are created with the editor itself. A file sharing service would be effective in sharing the documents created with this editor.

**Function Requirements**

**Functional Description**

***Overview***

A GUI where the user can paint (to a default location), and afterwards drag, objects such as class boxes and UML relationships.

***Feature List***

Objects can be painted onto right panel

Objects can be moved by click and drag

**User Interfaces**

***Overview***

The left panel is a list of object buttons that the user can click. On the right pane is the space where the objects will be drawn and interacted with.

***Menus***

The normal window menus, file, and view menus are included on the menubar. File has functionality for sub-menu options: “New”, “Print”, “Save-As”, and “Close”; it also has limited “Open” functionality where it will open files with their default program. Edit has sub-menu options, but no functionality in this iteration. View has no functionality in this iteration.

***Inspectors***

No inspectors in this iteration

***Use Cases***

No use cases in this iteration

**Non-Functional Requirements**

**Reliability**

Reliability is something should always be a primary concern of good software engineers. To achieve a good level of reliability, with copious amounts of testing to back our claim, is a conscious effort to make sure all errors are caught and thrown exceptions for, with a prime example being using save as in a directory without proper credentials, the program catches the error without crashing and terminating the active window.

**Performance**

With performance being our primary goal, we want to make sure this will always be our front and foremost priority, with as little sacrifice to reliability and portability as possible. What we have done to achieve this is by organizing the code in such a way that the UMLView is what the user sees, and is completely separate and independent of UMLController, which handles all of the interactions with the GUI.

**Usability**

Within this current iteration, everything is properly labeled, and relatively self explanatory in the hands of an experienced user with a background in UML. Usability is not necessarily achieved in this current iteration because a few of the features do not have the expected functionality of their functional counterparts in traditional programs.

**Portability**

With many different systems running around in the world of software engineering and development, we aim to create a program that will run seamlessly on the three big operating systems that dominate today’s market (Mac, Windows and Linux). Utilizing Java, a very well known programming language, and a java environment to run in, the ability to run our software on any of the three big Operating Systems in use today will be a very straightforward and easy task.

**Future Enhancements**

**In the future, we plan on implementing an inspector tool, adding a delete option for the objects, and painting relationships by source object and destination object instead of default location.**