# Design Document

## Modification History (of the design document)

First created 27/12/2014

Activity and Use case updated 30/01/2015

Draft Class Diagram added 7/02/2015

## Introduction

This documents purpose is to outline the design phase of the projects software lifecycle, in this case the design and implementation phase is being done in a prototype agile format.

Each change will be recorded under the modification history section.

## Environment

In this section the environment of the product will be outlined, both in terms of specific hardware and peripherals.

### Hardware

At this point the need for any specific hardware have not been outlined by the client or within the analysis and design phase of the project.

### Peripherals

At this point the need for any specific hardware have not been outlined by the client or within the analysis and design phase of the project.

## Fixed Interfaces

In this section any fixed interfaces that will be developed or required will be outlined and discussed.

### Operating System

For the eclipse plugin to work the machine will need to have eclipse installed which can run on both windows and linux.

### DBMS

The need for any database management systems have not been identified by the client or within the analysis and design phase.

### Interfaces to other systems

At this stage the need for interfaces to other systems have not been identified.

## Methodology

In this section the chosen methodology for the implementation of the product will be outlined and discussed.

Many methodologies and their spin offs were considered and some of these can be read in the research document.

However the iterative prototype method with elements of scrum was chosen for this group project. This was decided from the risk analysis in the risk document and by the research performed in the research document [Reference to those documents?]

This approach is ideal to this group as the member size is only 2 and therefore adopting a full agile methodology seemed unnecessary as these are tailored for small to medium teams which usually consist of at least 5 to 10 members [Get reference here]. The chosen approach allows for the two members of the group to work on both the implementation and documentation side of the product iteratively, each requirement will be completed and tested before the next requirement is addressed. As there is only 2 people this approach is efficient and easy to maintain [Reference here]. The scrum elements that will be present are a weekly meet up with the client to report on progress and discuss on any topics such as requirements validation.

The nature of the product also requires this methodology, as the knowledge of how to implement a plug in for eclipse is unknown. The iterative prototype approach lets us tackle each requirement separately until it is implemented and tested. As each requirement will involve different functionality that will have to be first researched and learned.

Another reason for this approach is that it allows us to receive feedback during implementation from the client or test sessions held with students. As the product is intended for students it would be wise and useful to use feedback from them and the client after each prototype. With the feedback gained and used after each working prototype it ensures a final product which the client truly wanted.

## System Structure

## System Models

In this section UML diagrams will be shown in relation to the requirements set out in the requirements analysis document.

### Activity Diagrams

This diagram is an activity diagram. It shows the process of the user using the system in a step by step fashion, the diagram begins with a start node and ends with an end node. In between these nodes, actions and decisions are outlined.

This version of the activity diagram shows the primary process that the user would take when using the plugin.

After opening the object bench toolkit perspective the user can decide to either create a new java project or select an already existing one. From there the class diagram view is opened, this allows the user to navigate packages and / or classes. The user can either create a package or class or select an already existing class.

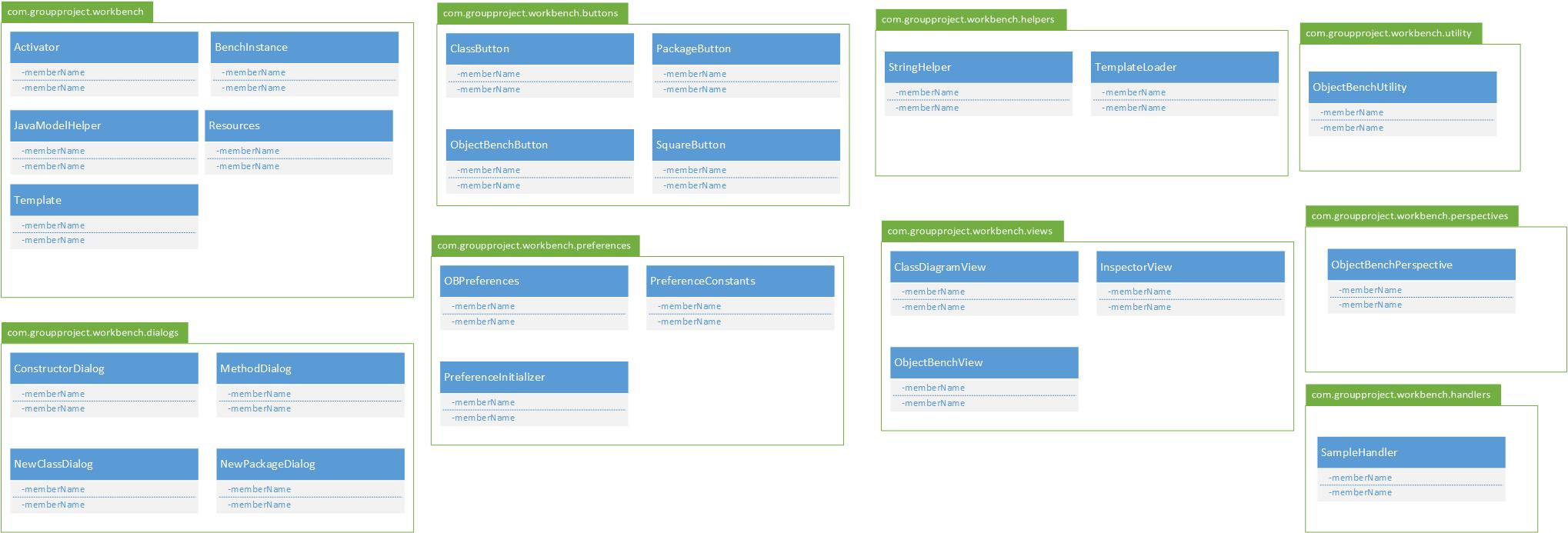
From there the user can both open the code editor and add code to that class or instantiate that class. Instantiating the class places the class on the object bench. Selecting this instantiation allows to either inspect it or run a method.

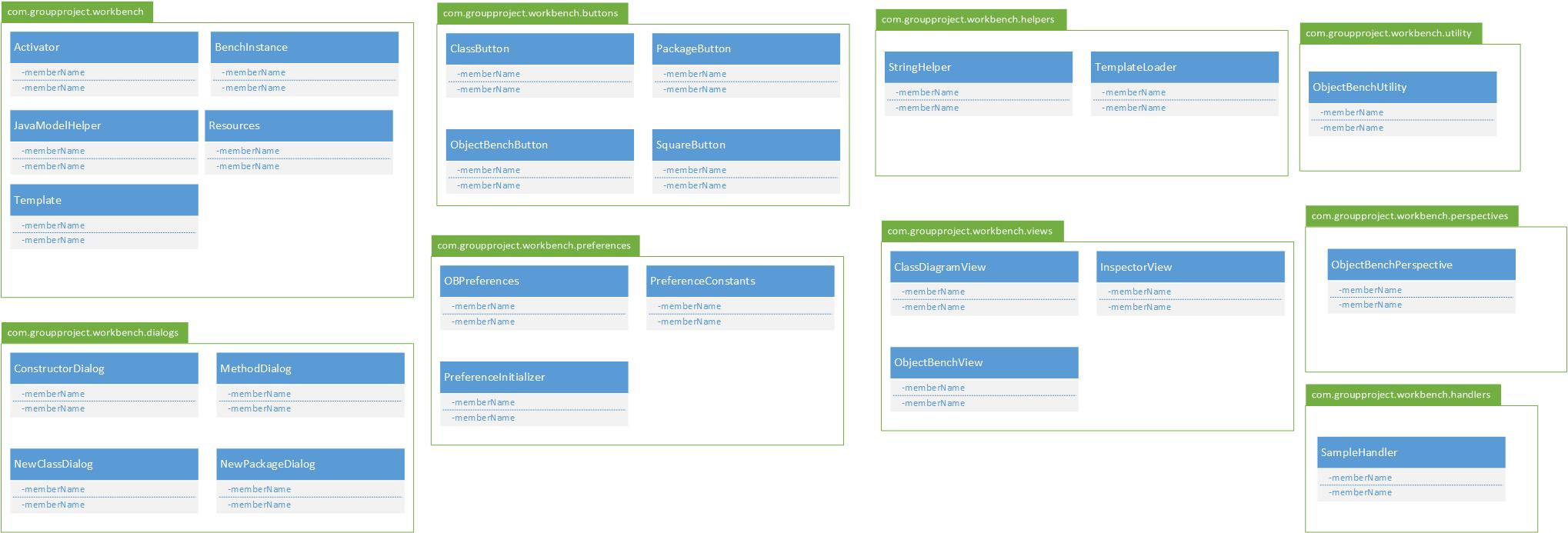
### Use Cases

This diagram is a use case, it shows all the interactions the actor (the user of the plugin in this case) can perform to the system (The Eclipse IDE with the plugin). Here the actor can do the following things:

* Create a new java project
* Select a java project
* Open the perspective of the plugin
* Add a class
* Add a package
* Select a class
* Open the code editor
* Add a method through the code editor
* Instantiate a class
* Selecting the instantiation
* Inspecting the instantiation
* Running a method of the instantiated class
* Running the compiler

### Class Diagram





This diagram shows the structure of the packages and classes of the plugin.

The current version is an early draft as packages and classes may get added and removed.

## System specifications

### Interface

### Function (and / or)

### Detailed Algorithm

### Performance Requirements

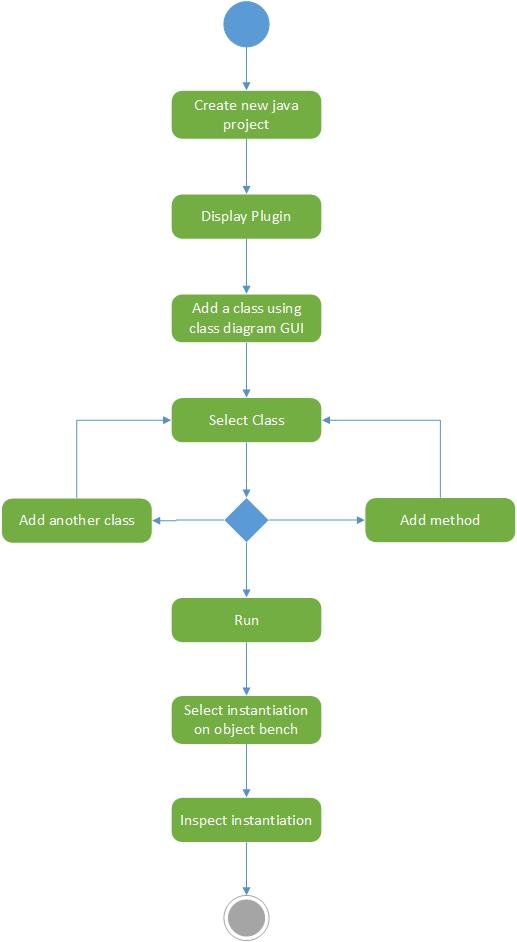
### Exception Conditions

## Implementation Plan

## Glossary

## Appendix

Activity diagram v1



Use Case diagram v1

