|  |
| --- |
| CPSC 3720 Spring 2018 |
| Code:Blocks Plugin:  Implementation File Generator |
| [Team Logo]  The Remaining Jerry’s  Project Group E |
|  |
| **Jace Riehl**  **Nathan Tipper**  **Vincent Cote** |
| **February 2nd, 2018** |

Table of Contents

[Introduction 3](#_Toc505335679)

[Proposal 3](#_Toc505335680)

[Project Management 3](#_Toc505335681)

[Team Organization 3](#_Toc505335682)

[Risk Management 3](#_Toc505335683)

[Devevlopment Process 4](#_Toc505335684)

[Coding Conventions 4](#_Toc505335685)

[Code Review Process 6](#_Toc505335686)

[Communication Tools 6](#_Toc505335687)

[Change Management 7](#_Toc505335688)

[Appendices 7](#_Toc505335689)

[Appendix A: Figures and Tables 7](#_Toc505335690)

# Introduction

# Proposal

The idea behind the Implementation File Generator plugin is that we would like to simplify the class creation process. When a developer is creating a class, we start by making the interface for the class i.e. the header file. Once a header file is made, we have to go through the tedious process of rewriting all the signatures of our prototypes and then implementing the functionality of those functions. What the Implementation File Generator would do for us is the following:

* Create a .cc file with the same name as the header file and save it to your /src folder or a designated folder given by the user.
* Include the header file from which the code was generated.
* Create all signatures within the header file with braces for the implementation and, if there is a return type or no variables names provided, create a return statement with a default value and default argument names.

Now the user no longer has to create an implementation file again. They simply need to create the implementation and configure their arguments and return statements to match their implementation.

# Project Management

## Team Organization

All none code related documentation is to be done using LibreOffice Writer in order to keep a set standard for all team members. As this is a school project and is used as a learning tool for all team member we understand that we are all designers, programmers, leaders and documentation gurus. With that being said the group is still structured with each member having a specific role. These roles are a guideline and will help the team know who to approach with specific issues concerning the different aspects of the project. The specific roles are as follows:

* **Team Lead:** Nathan Tipper
* **Design Lead:** Vincent Cote
* **Quality Assurance Lead:** Jace Riehl

## Risk Management

As with any project, uncertainty is always present and can create chaos if the team is not prepared. The team has come up with a few different problems that could occur on this project as well as some solution on how to deal with said problems if they occur. Being aware that any of these scenarios could occur at any phase during the project is a good method to help prevent and/or handle these issues. The most important issues discussed are as follows:

* **Unforeseen Major Life Event.** This includes, but not limited to, death in the family, major illness rendering the team member unable to work for extended period of time, unforeseen financial setback (lose of house, claiming bankruptcy). As this is the most extreme case and (hopefully) the least likely, we have agreed as a group on what to do in these events. If a team member undergoes one of these unfortunate situations, the remaining team members will take on the responsibilities and divide them equally between each other until the team member is ready to come back.
* **Loss of Team Member.** The loss of a team member could be due to the events listed above or simply by having a team member drop the class. In any case, we feel we have a strong enough team to divide the work into three and still meet the deliverables and deadlines. If two are more team members leave the group a meeting with Dr. Anvik will be arranged to discuss options.
* **Unproductive Team Member.** An unproductive team member does not always mean the individual is lazy (although sometimes it is the case). We value a productive, healthy and open team management style, this means the team members support each other in all aspects inside and outside of the project. We encourage each other at all milestones no matter how small or large, this can lead to higher productivity through positive encouragement. If these prevention methods do not help and a team member is still being unproductive a meeting between all team members is to be called to discuss the issue.
* **Inexperienced Team Member.** It is understandable that sometimes egos can get in the way and lead to a team member taking on more than he/she can handle. Based on the teams values listed above, the group will gain a good understanding of each team members strength and weaknesses. The idea is to help each other guide us into tasks at which we are best at. If a team member falls behind due to a lack of experience, the other team members will intervene in order to help the one who is falling behind. If needed tasks may be reassigned to better suite the team members strength.

# Devevlopment Process

## Coding Conventions

Throughout the lifetime of this project, our team has decided to follow the following coding conventions:

* **Starting blocks on the next line of any scope.**
  + Ie :

Class Class1

{

...

};

* **Using camelCase for both variables (attributes) and functions (members)**
  + Ie:

int thisIsAVariable = 1;

void thisIsAFunction()

{

...

}

* **Class names are to be capatilized and then use camelCase**
  + Ie:

Class ThisIsAClassThatDoesNotFollowExistingConventions { ... } ;

* **Variables, functions, and class names should be clear and consise and be named with respect to their function.**
  + Example:

int numberOfJerrys = 4;

* **Comments should be short and give a broad overview, leaving the variable, function, and class names to speak to the functionality of the code.**
* **All nested scopes should be indented a level in for a readibility.**
* **Each header file should start with a ifndef, def statement to ensure header files are not duplicated throughout the code. The defined statement should be read as the name of the class in all capital letter followed by “\_H.”**
  + Example:

#ifndef SOMECLASS\_H

#define SOMECLASS\_H

class SomeClass { ... };

#endif // SOMECLASS\_H

* **Use accessor methods for users to use instead of public attributes.**
* **Documentation of all members in header files is recorded and is required to have all of the following:**

 What the function does.

 Description of parameters and what the return value is.

 How the function modifies the object.

 The pre-condition and post-conditions of the function.

Any restrictions the function may have.

Procedures for Configuration Management

As agreed upon by our team, Pull requests are to be handled in a ‘pull requests’ fashion, where if there is a branch off it will require a pull request to merge back into the trunk which must then be approved by the design lead. For more information on merging back into the main trunk, refer to the code review process section.

## Code Review Process

## Communication Tools

The primary methods of communication for the duration of the project will be face to face meeting as well as using slack for text based communication regarding general information, random information, design documentations, game design, game structure and repository documentation; all of which are subdivided into their own channels. If face to face meetings are not feasible due to scheduling conflicts the team is to arrange a video conference call on Google Hangout at the earliest opportunity.

## Change Management

Any group member will be able to issue bug reports that they encounter from the program through bitbucket. This ensures that bugs are not forgotten about and ensures that the team lead is involved in the whole process. When a bug is found, the reviewer who identified the bug(s) will assign them to the developer who wrote the function/class. Only the reviewer who identified the specific bug is allowed to close it. Once the bug has been resolved, the developer is to report back to the reviewer who will test it again until it is agreed that it is fixed or that it is more of a feature than a bug; at which point the issue will be closed through bitbucket.

# Appendices

## Appendix A: Figures and Tables