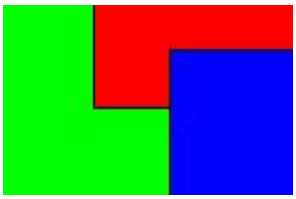
3Blockz



3Blockz Development

Team F

Cody Crawford, Tyler Loewen, Troy Paul

February 02, 2018

Table of Contents

Introduction

When writing code, developers often repeat a lot of similar code, or trouble themselves of writing new code for common design patterns such as builders, factories, etc. This often leads to wasted resources such as time and money, or even simple mistakes causing bugs to be created.

We aim to write a C++ plugin for code blocks to address these problems. The plugin will automate the creation of common code and structures while giving developers flexibility to choose specifics in the generation of code through a simple user interface (UI).

Some example code automation tools which could be written for our plugin include “builder pattern generator,” “factory pattern generator,” “equality function generator,” “constructor generator,” “getters and setters generators,” “cout printer generator,” “hash code function generator,” etc. The list goes on for the possibilities.

By using a code generation tool like the ones which will be included in our proposed plugin, developers will be able to: save time and resources; reduce boredom when creating very common and simple code; reduce scope of human error; encourage the use of better, pre-defined build patterns instead of sloppy code created under pressure or with limited knowledge, etc.

This may not be a new idea, but we plan to make our plugin more simple to use and easier to edit and extend. We will ensure high code quality including vast code testing and documentation.

Proposal

Outline

We will be designing a simple menu item in the plugins menu. Within our menu will be the tools we are creating. See Appendix A for drop down menu diagram.

These tools will be as follows:

1. Empty hashcode and equality function generators

   These tools will allow the users to generate an empty hashcode function for use in their program as well as generating an equality function. This will be accomplished by prompting the user for the data type which the function should be used for. We may then further explore the option of allowing the user to specify certain functions to call or parameters to use from the object to automatically generate the hashcode. The same goes for the equality function generator.
2. Getter and setter generator

   The getter and setter functions are useful and necessary to many objects created. The getter/setter generator will prompt the user variable names and types and will generate the appropriate getter/setter functions. When selecting this option a prompt will appear for the class name and location, as well as variable names and types. With this info the plugin will be able to build any getter and setter the user could need.
3. Builder class generator

   The builder pattern makes it a lot easier for users to create new objects without the need to have a ton of different constructors. This tool will prompt the user for required variable names and types, as well as optional variable names and types. It will then generate a class following the builder pattern using those specified parameters. The user will also be able to specify domain constraints for integer data types and possibly others if possible. These constraints will then be validated in the appropriate function of the generated builder class.

Additionally, at the bottom of our menu we will add an about link which opens up a interface explaining who we are, the aim of the project and a link to our project page (such as GitLab). There will also include a user manual to explain all the uses of our plugin as well as examples and diagrams. The code generated will produce something similar to that shown in the link provided in Appendix B.

Timeline

We plan to complete this project in three two-week sprints. The first will be composed of item 1 from above. The second will be composed of item 2. The third will be composed of item 3 and the help/about interface.

Essential / Desired / Optional Requirements

Essential

- Items 1-3 from above

Desired

- Help/about interface

Optional

- Non-empty hashcode and equality function generators

Project Management

Team Organization

People

- Cody Crawford: Team Lead, Software Developer, Software Tester
- Troy Paul: Documentation Lead, Software Developer, Software Tester
- Tyler Loewen: Quality Assurance Lead, Software Developer, Software Tester

Roles

- Software Developer
    - Responsibilities
        - Design software including software structure, following the SOLID principles
        - Write implementations of software design
        - Write documentation to clearly define the behavior of functions, classes, etc.
- Software Tester
    - Responsibilities
        - Write all applicable test cases using black, grey, and white box testing techniques
        - Fix bugs found in the testing process
- Team Lead
    - Responsibilities
        - Coordinate with team to organize group meetings
        - Ensure team members are on track
        - Take input from team
- Documentation Lead
    - Responsibilities
        - Record thorough answers to SCRUM questions and make them accessible to group-mates throughout project
        - Ensure all documentation in implementation is legible and useful information
        - Provide a helpful and thorough user manual for the project
- Quality Assurance Lead
    - Responsibilities

- Ensure project testing framework is properly working
- Set project standards for code quality
- Ensure test first development
- Ensure SOLID principles are being implemented
- Ensure white/grey/black box testing techniques are used in all possible cases
- Ensure high code coverage
- Ensure a high rate of test cases pass
- Ensure the code is free of memory leaks
- Regularly review reports such as static analysis reports, code coverage reports, testing reports, etc.

Risk Management

1. Requirements/Design/Estimation
    a. Using scrum management techniques, we will be able to detect if and when we “bit off more than we can chew”
    b. Review what is completed after first sprint and re-estimate and re-evaluate our time
    c. Plans will be designed in advance and specific enough to understand what is needed for each phase. A detailed plan of the first sprint and generalized plans for the next two sprints will be created. In doing so, we will be able to catch major changes needed sooner, rather than later.
2. People
    a. Diversify existing work based on the responsibility documented for the lost group member
    b. Team leader Cody will be monitoring each member to ensure that they are holding up responsibilities. Group meetings will be held if necessary to discuss members not pulling their own weight.
    c. Work may be reassigned if necessary, or group members will assist the lacking member in learning necessary background
    d. For unanticipated life events work may be shifted to ease the load of team members, but they will be expected to pick up work as they return back to normal life
3. Learning & Tools
    a. Group members will be able to contact and assist others with the use of new tools. The aim is to not let a group member be stuck on an issue for too long.
    b. When the learning curve is too steep, group members will gather to assess the value of using the new tools and decide whether to continue using the tool and learn it together, or to stop using it at all
    c. When tools don’t work together in an integrated way, the team will locate new tools and test them in a more isolated environment to ensure they integrate together

Development Process

Coding Conventions

Class Names

- Upper camel case
- Example: ClassOne

Function Names

- Lower camel case
- Example: myFunction

Constant Variables

- All uppercase with underscores acting as spaces
- Example: CONSTANT_ONE

JavaDocs Documentation Style

- See: http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html
- Example:
  /** General description of what the function does, how it operates on the input, how it calculates the output, etc.
  @param x Description for what x represents along with any constraints.
  @return The specifics of what is returned, the conditions for which certain values are returned, the range if known and definite, etc.
  @throws Exception When and why this specific exception type is thrown, how to recover from it, etc.
  @see Link to any resources to further explain any complicated understandings. **/
  int func1(int x) throws Exception;

Google C++ Style

- Other various conventions as listed in Google's C++ Style Guide (https://google.github.io/styleguide/cppguide.html) which do not contradict our explicitly defined coding conventions are encouraged

Code Review Process

The team will review each others work for overlooked mistakes, or any other improvements that can be made to stick with the coding conventions and style. The quality assurance lead will then further use the code quality assurance methods listed under their responsibilities above to ensure high code quality.

Collaboration Tools

Gitlab will be main tool used for file management. Google Drive will be a secondary tool. GitLab is connected to a continuous integration (CI) server to automate testing and report/documentation generation.

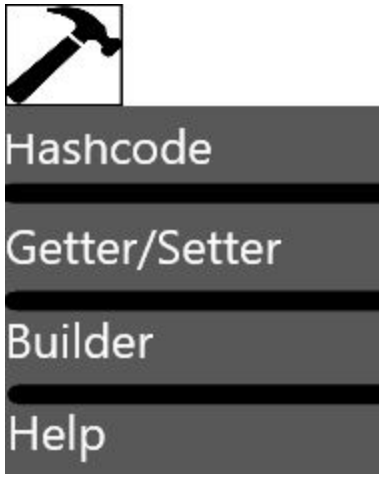
Communication Tools

The team will be using Slack to communicate. With in Slack, gitlab has been integrated to make it easier to notice when changes are pushed, builds fail, etc. Email will also be use as a secondary communication tool.

Change Management

If and when major changes are to be made, the team will meet and collaborate on how to best handle the situation. Assignments will be discussed and sorted in this meeting. A mix of waterfall and agile planning will be done to prevent major changes from being required.

Appendix

A: 

B. http://en.cppreference.com/w/cpp/utility/hash