CMaker

Patrick Krukenfellner

03.03.2018

Contents

1 Introduction

1.1 What is the purpose of this project?

Working with cmake is great. But sometimes, especially when you work without an IDE, it can be very annoying to create the folder structure, the CMakeLists.txt and so on. Also it is very time consuming when you try to use cmake with Java. So it would be great to have a tool that simplifies the work with cmake, right? And that's the point where CMaker comes in.

1.2 What is CMaker?

CMaker is a little tool which will help the user to create cmake based projects. It will minimise the effort to create a cmake project. The main purpose of CMaker is, to automatically create the standard cmake folder structure, as well as a pre defined CMakeLists.txt. The user will still be able to define all settings of the project, like the name, the entry point and even if he wants to use unit tests.

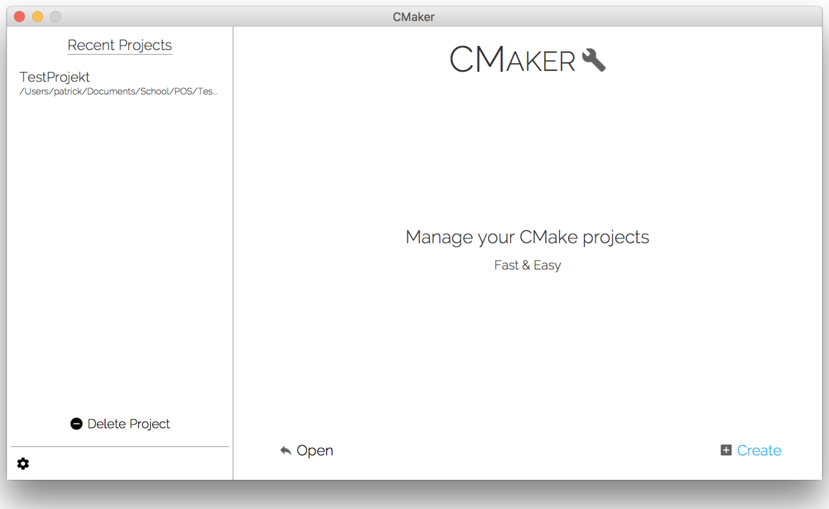
1.3 Further functions

The program will also offer features to work with unit tests. This means that the user (provided that he enabled the unit tests for that project) will be able to automatically create JUnit test files for his java files, as well as run the tests over the built-in terminal.

2 User Guide

2.1 Start page

Most functionality can be found on the starting page of the program:



2.1.1 Recents Bar

On the left side you can see a "Recent Projects" bar. All projects will be added to this bar automatically when they are created.

There is also on option to delete the project.

2.1.2 Open Button

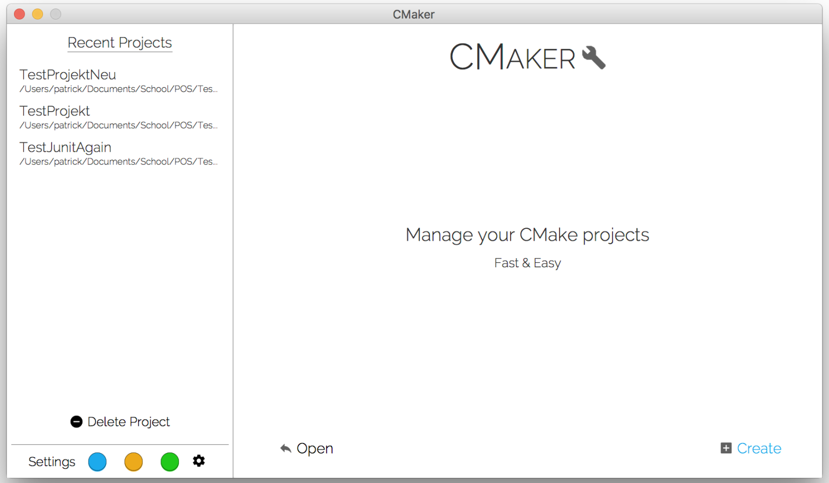
This button is used to open an existing project. When pressed, it will open a directory chooser window.

2.1.3 Create Button

This button is used to create a new project. It will open a new window where you can specify the project settings.

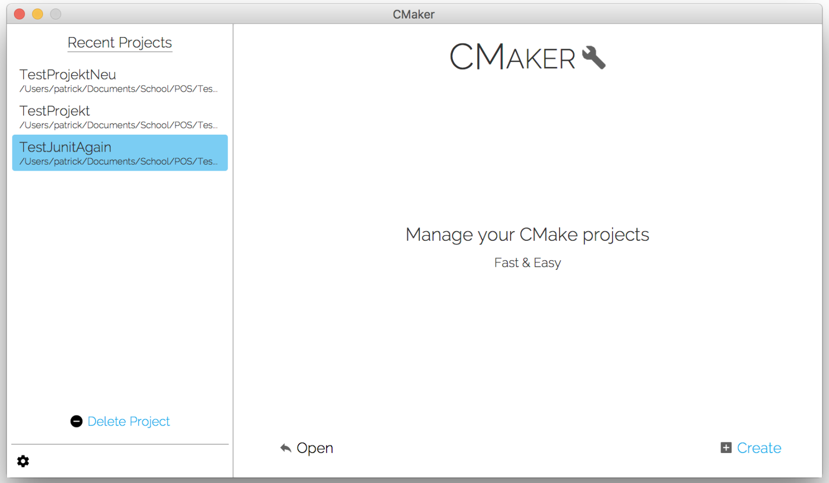
2.1.4 Settings Wheel

The settings wheel will show a bar where you can choose an accent color or view the settings window.


CMaker
Recent Projects
TestProjektNeu
/Users/patrick/Documents/School/POS/Tes..
TestProjekt
/Users/patrick/Documents/School/POS/Tes..
TestJunitAgain
/Users/patrick/Documents/School/POS/Tes..
Delete Project
Settings
CMAKER
Manage your CMake projects
Fast & Easy
Open
Create

2.1.5 Delete Button

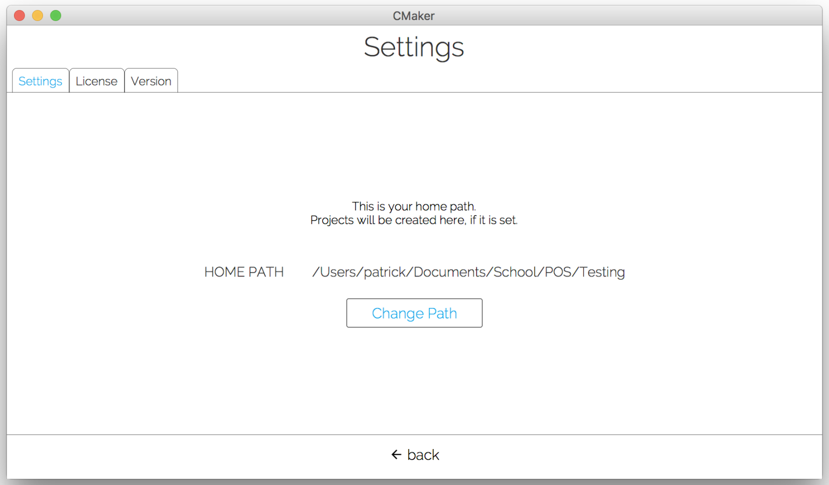
If the user wants to delete a project, he has to select it in the recents bar and press the delete button.


CMaker
Recent Projects
TestProjektNeu
/Users/patrick/Documents/School/POS/Tes..
TestProjekt
/Users/patrick/Documents/School/POS/Tes..
TestJunitAgain
/Users/patrick/Documents/School/POS/Tes..
Delete Project
CMAKER
Manage your CMake projects
Fast & Easy
Open
Create

2.2 Settings

2.2.1 Home Path

In this window, the user is able to set a home path where all projects will be created by default.

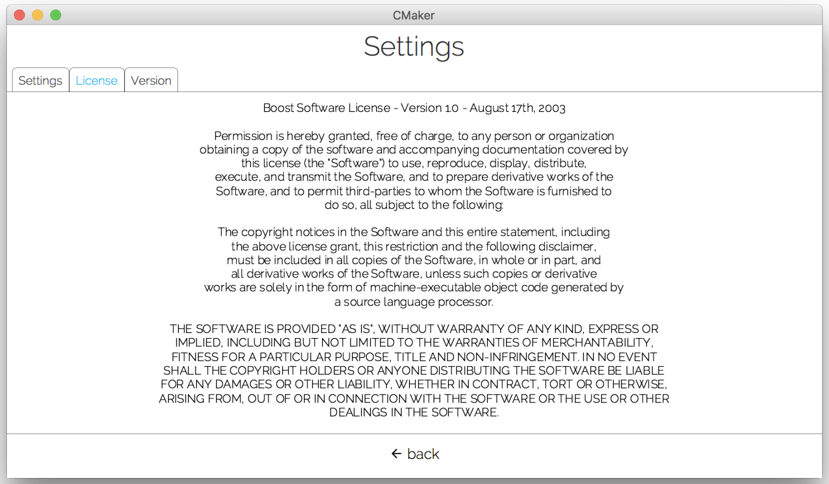


This is your home path.
Projects will be created here, if it is set.

HOME PATH /Users/patrick/Documents/School/POS/Testing

2.2.2 License

This window will display the projects 'boost license'.



Boost Software License - Version 1.0 - August 17th, 2003

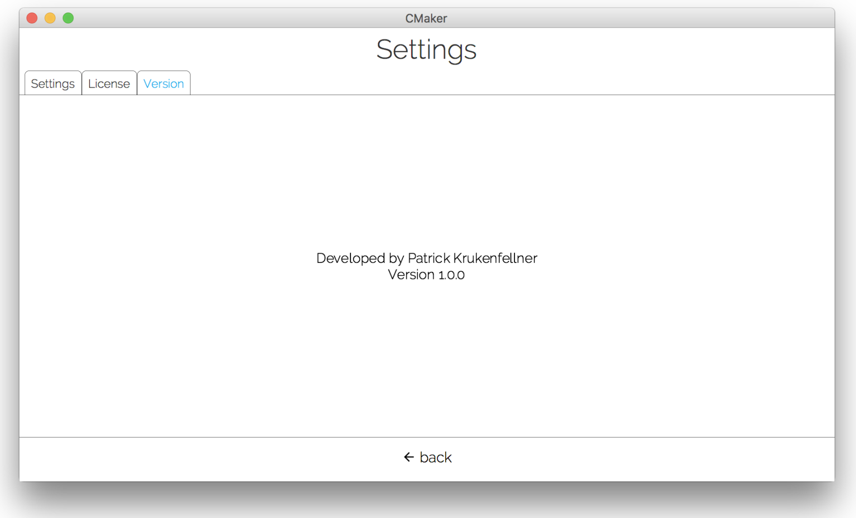
Permission is hereby granted, free of charge, to any person or organization obtaining a copy of the software and accompanying documentation covered by this license (the "Software") to use, reproduce, display, distribute, execute, and transmit the Software, and to prepare derivative works of the Software, and to permit third-parties to whom the Software is furnished to do so, all subject to the following:

The copyright notices in the Software and this entire statement, including the above license grant, this restriction and the following disclaimer, must be included in all copies of the Software, in whole or in part, and all derivative works of the Software, unless such copies or derivative works are solely in the form of machine-executable object code generated by a source language processor.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE SOFTWARE BE LIABLE FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER IN CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

2.2.3 Version

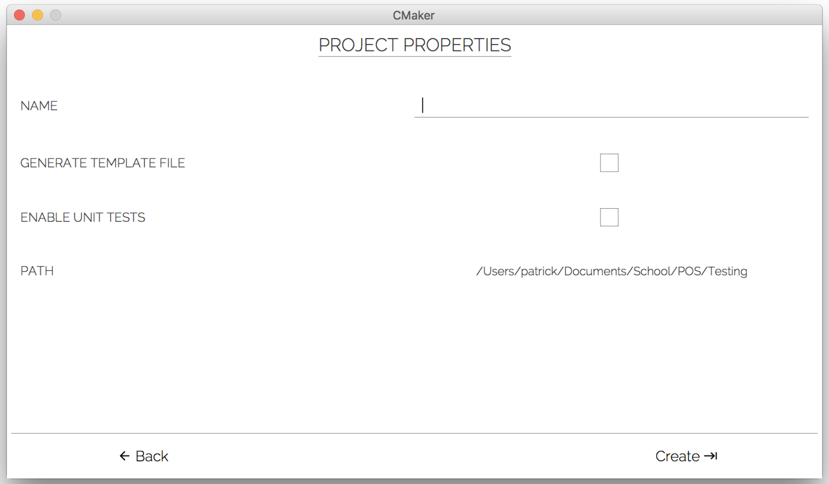
Here, the program provides information about the actual version.



Developed by Patrick Krukenfellner
Version 1.0.0

2.3 Create Project

The create button will lead you either to a directory chooser, where you can choose the path where the project will be created, or it will lead you directly to the project properties window, where you can specify the projects settings, depending on if the home path is set or not.


CMaker

PROJECT PROPERTIES

NAME

GENERATE TEMPLATE FILE

ENABLE UNIT TESTS

PATH /Users/patrick/Documents/School/POS/Testing

2.3.1 Name

Here the user can specify the projects name.

2.3.2 Generate Template File

If the user picks this option, a 'Main.java' template file will be added to the project.

2.3.3 Enable Unit Tests

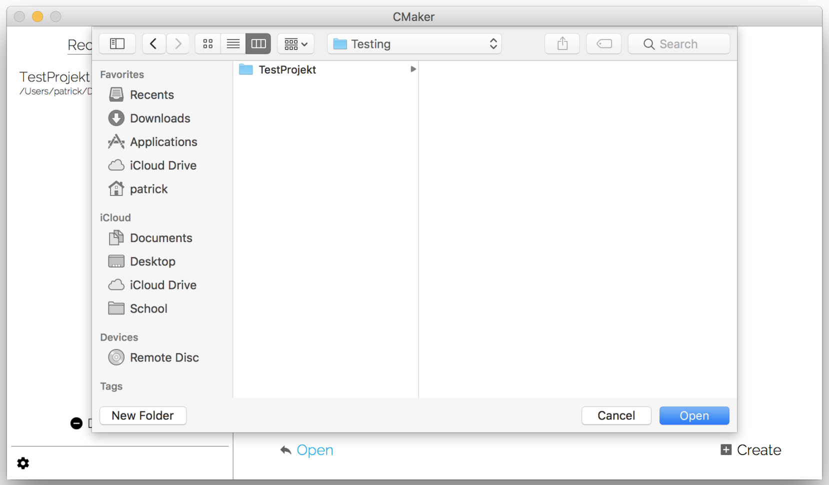
This option will decide, if the project will contain settings to create unit tests.

2.3.4 Path

This is the path where the project will be created. If set, this will automatically be the home path.

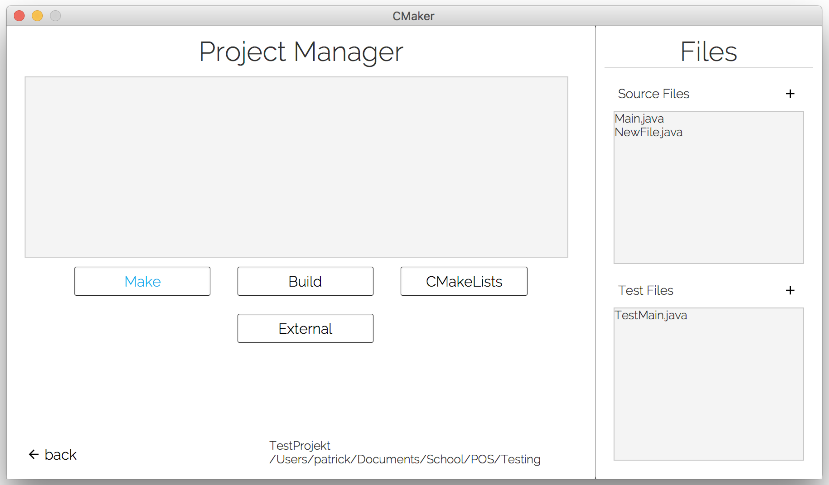
2.4 Open Project

If you click the open button, it will show a directory chooser. If you pick a project which was created by CMaker you will get to the Project Manager.



2.5 Project Manager

Here the user is able to manage his project, which can be compiled via the built in console. Also the user is able to add source files, or generate test files.



2.5.1 Make Button

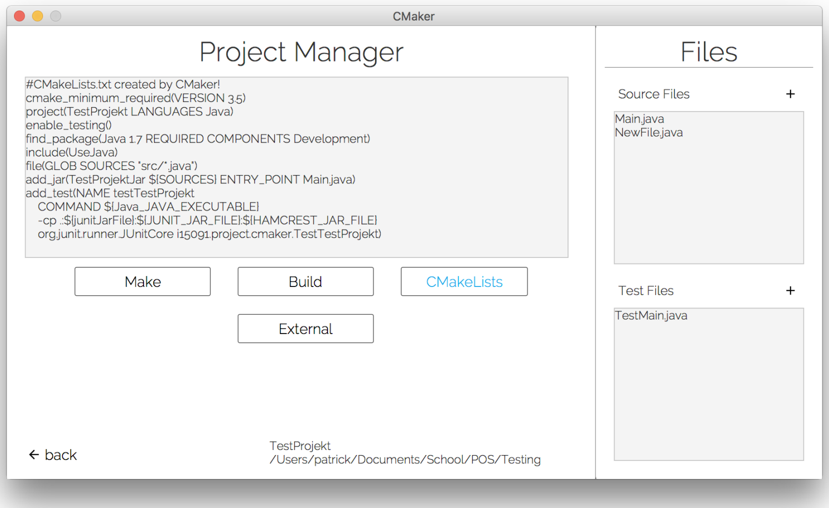
This button is used to perform a 'make' command in the projects build folder.

2.5.2 Build Button

This button is used to perform a 'cmake ..' command in the projects build folder.

2.5.3 CMakeLists Button

This button is used to view the CMakeLists.txt file. It will be shown over the built-in console.



```
#CMakeLists.txt created by CMaker!
cmake_minimum_required(VERSION 3.5)
project(TestProjekt LANGUAGES Java)
enable_testing()
find_package(Java 1.7 REQUIRED COMPONENTS Development)
include(UseJava)
file(GLOB SOURCES "src/*.java")
add_jar(TestProjektJar ${SOURCES} ENTRY_POINT Main.java)
add_test(NAME testTestProjekt
   COMMAND ${Java_JAVA_EXECUTABLE}
   -cp .:${junitJarFile}:${JUNIT_JAR_FILE}:${HAMCREST_JAR_FILE}
   org.junit.runner.JUnitCore i15091.project.cmaker.TestTestProjekt)
```

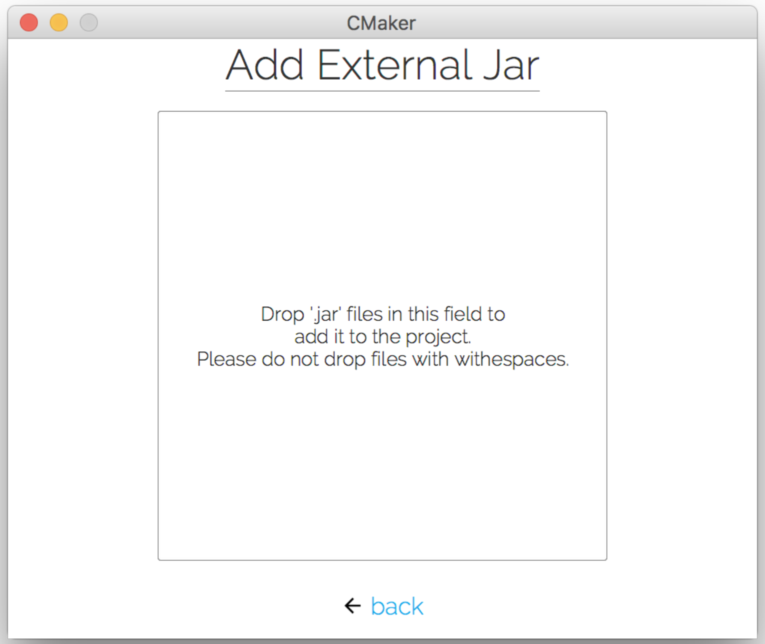
Main.java
NewFile.java

TestMain.java

TestProjekt
/Users/patrick/Documents/School/POS/Testing

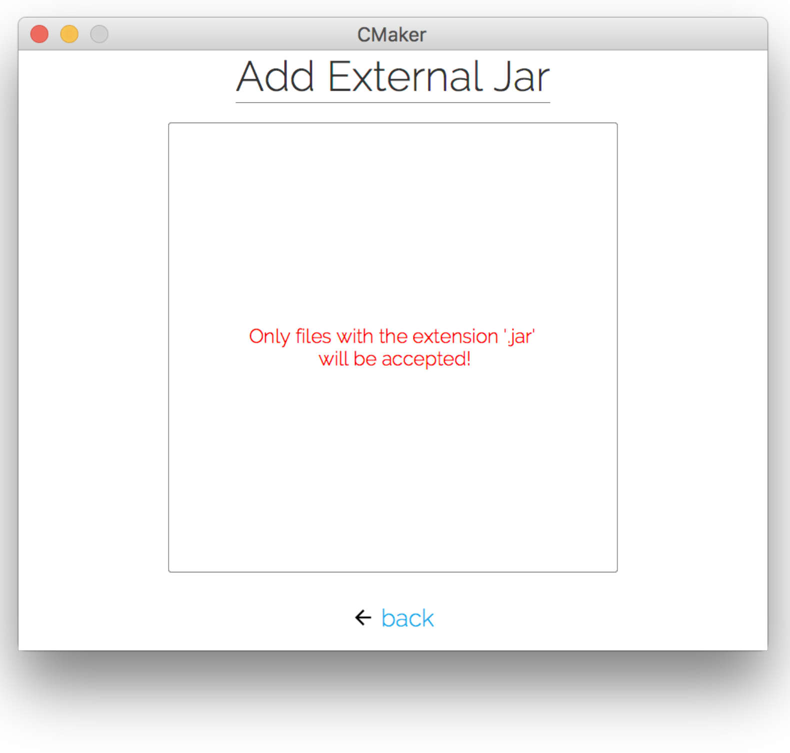
2.5.4 External Button

The External Button will open a window, where the user can drop .jar files, which will be added to the project. Therefor a new folder 'external' will be created (or the file will be added to the external folder if it already exists) and the jar will be added to the 'CMakeLists.txt'. As the information says, this will only work with files which do not include a whitespace in their name.



Drop 'jar' files in this field to
add it to the project.
Please do not drop files with withespaces.

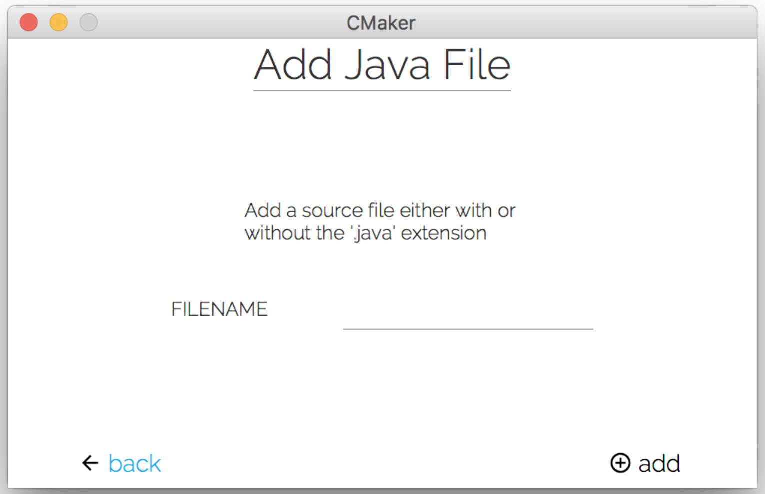
The user can only add files with the '.jar' extension, otherwise this error message will be displayed:



Only files with the extension 'jar' will be accepted!

2.5.5 Add Source File

Here the user can add a new source file to the project. The '.java' extension can, but does not need to be added. If not, it will be added automatically.

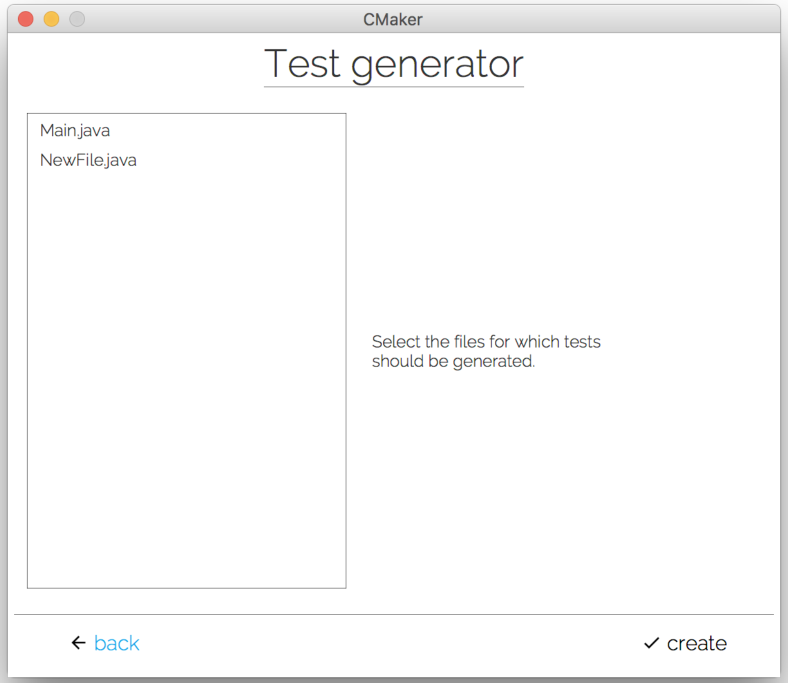


Add a source file either with or without the '.java' extension

FILENAME

2.5.6 Add Test File

In the test generator window, the user can choose source files for which test files will be generated.



Select the files for which tests should be generated.

2.6 ShortCuts

2.6.1 Start Page

1. **CTRL + Q** Quit application
2. **CTRL + O** Open Project
3. **CTRL + C** Create Project
4. **CTRL + S** Settings

2.7 Project Manager

1. **CTRL + M** Make Project
2. **CTRL + B** Build Project
3. **CTRL + P** Preview
4. **CTRL + E** Add external jar file
5. **CTRL + T** Add test file(s)
6. **CTRL + A** Add source file

2.8 License

The project is distributed under boost software license.

3 API

This section is for everyone who is interested in the background and thoughts behind this project. It will explain the ideas and solutions that were used to encounter several problems that occurred during the development process.

3.1 Classes

The classes section will give you a general overview over the backend classes. There will be a short description as well as an explanation of the purpose of each class.

3.1.1 Project

The project class should represent a CMaker project. It will create the folder structure as well as several files such as 'CMakeLists.txt' and 'CMaker.properties'. Also it holds the projects information e.g. name and path.

3.1.2 Maker

This class is used to manage project objects. It not only contains the project settings but also the project itself and holds several methods to work with the project e.g. delete the project, clear the build folder (for compiling purposes), check if it is a valid CMaker project or get the files of the source folder.

3.2 Helper Classes

This are that classes will assist the programmer in the developing process and ease a lot of tasks.

3.2.1 CSVHandler

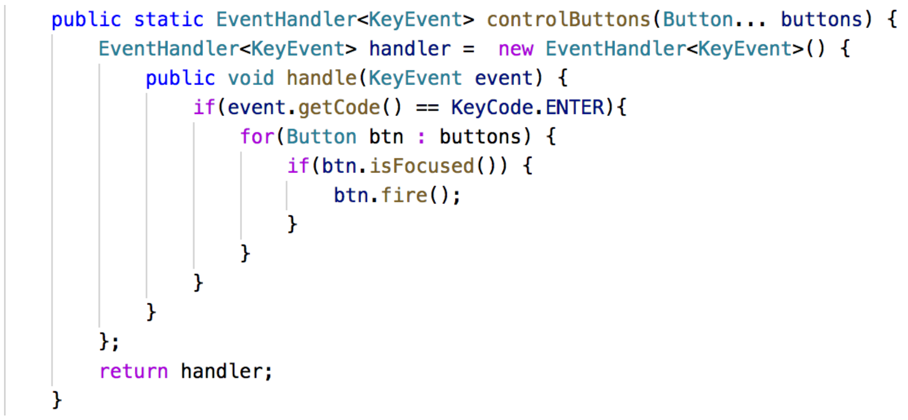
The CSVHandler class is a helper class whose purpose is to handle and ease all actions with CSV files, such as writing in and reading out of a csv file, which is the file type used in this project to contain persistent data.

3.2.2 FXHelper

Another class to assist in the developing process is the FXHelper class which provides several methods to handle JavaFX operations such as creating objects in a more comfortable way.

3.2.3 CMKeys

The 'CMKeys' class contains functions to write code that effects keyboard actions, such as 'controlButtons()' which gets a variable amount of buttons, and returns and EventHandler that checks if a Button is focused and fires this button when the 'Enter' button is pressed.



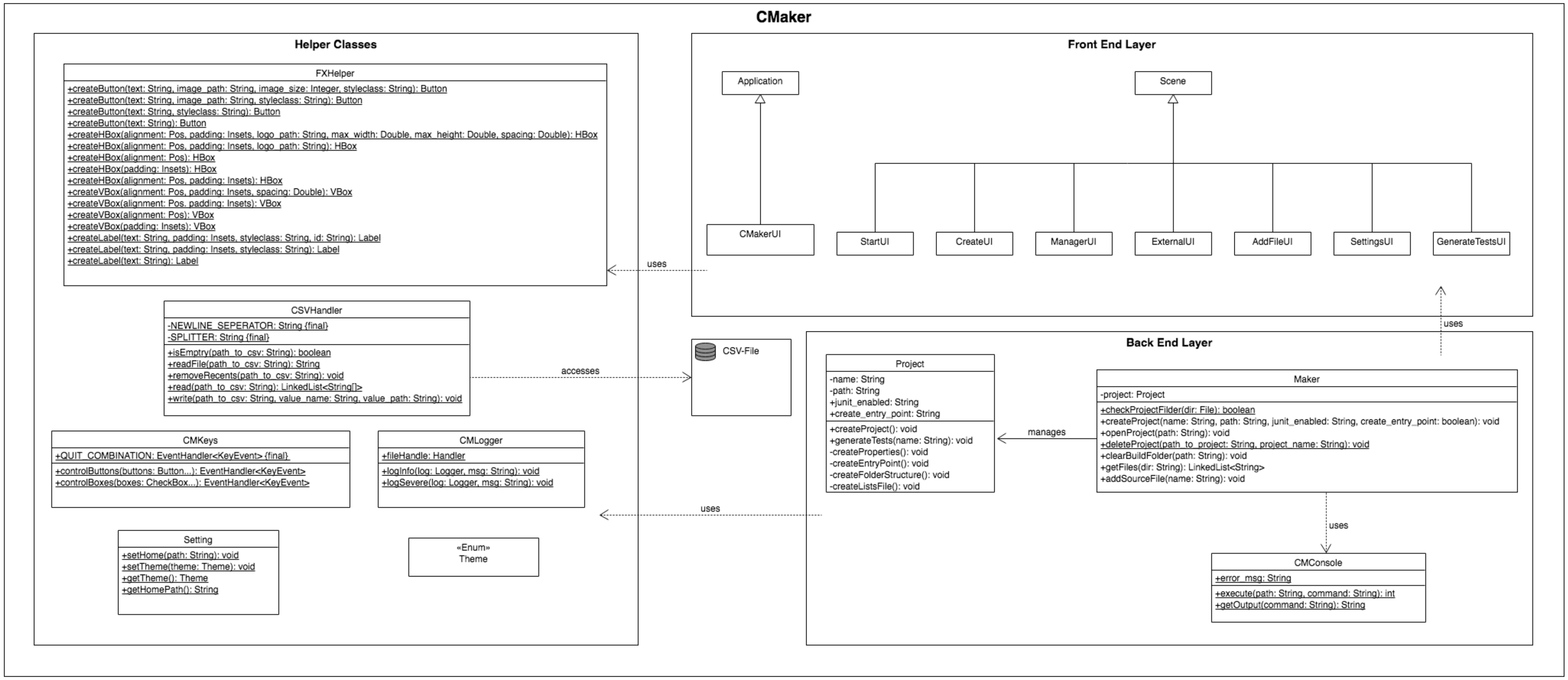
```
public static EventHandler<KeyEvent> controlButtons(Button... buttons) {
    EventHandler<KeyEvent> handler =  new EventHandler<KeyEvent>() {
        public void handle(KeyEvent event) {
            if(event.getCode() == KeyCode.ENTER){
                for(Button btn : buttons) {
                    if(btn.isFocused()) {
                        btn.fire();
                    }
                }
            }
        }
    };
    return handler;
}
```

3.2.4 Setting

This class holds all information about the projects environment e.g. Theme or Home path.

3.3 UML

The UML diagram can also be found in the projects 'Docs' folder as an external png.


CMaker
Helper Classes
FXHelper
+createButton(text: String, image_path: String, image_size: Integer, styleclass: String): Button
+createButton(text: String, image_path: String, styleclass: String): Button
+createButton(text: String, styleclass: String): Button
+createButton(text: String): Button
+createHBox(alignment: Pos, padding: Insets, logo_path: String, max_width: Double, max_height: Double, spacing: Double): HBox
+createHBox(alignment: Pos, padding: Insets, logo_path: String): HBox
+createHBox(alignment: Pos): HBox
+createHBox(padding: Insets): HBox
+createHBox(alignment: Pos, padding: Insets): HBox
+createVBox(alignment: Pos, padding: Insets, spacing: Double): VBox
+createVBox(alignment: Pos, padding: Insets): VBox
+createVBox(alignment: Pos): VBox
+createVBox(padding: Insets): VBox
+createLabel(text: String, padding: Insets, styleclass: String, id: String): Label
+createLabel(text: String, padding: Insets, styleclass: String): Label
+createLabel(text: String): Label
CSVHandler
-NEWLINE_SEPERATOR: String {final}
-SPLITTER: String {final}
+isEmptry(path_to_csv: String): boolean
+readFile(path_to_csv: String): String
+removeRecents(path_to_csv: String): void
+read(path_to_csv: String): LinkedList<String[]>
+write(path_to_csv: String, value_name: String, value_path: String): void
CMKeys
+QUIT_COMBINATION: EventHandler<KeyEvent> {final}
+controlButtons(buttons: Button...): EventHandler<KeyEvent>
+controlBoxes(boxes: CheckBox...): EventHandler<KeyEvent>
CMLogger
+fileHandle: Handler
+logInfo(log: Logger, msg: String): void
+logSevere(log: Logger, msg: String): void
Setting
+setHome(path: String): void
+setTheme(theme: Theme): void
+getTheme(): Theme
+getHomePath(): String
«Enum»
Theme
uses
accesses
uses
CSV-File
Front End Layer
Application
Scene
CMakerUI
StartUI
CreateUI
ManagerUI
ExternalUI
AddFileUI
SettingsUI
GenerateTestsUI
uses
Back End Layer
Project
-name: String
-path: String
+junit_enabled: String
+create_entry_point: String
+createProject(): void
+generateTests(name: String): void
-createProperties(): void
-createEntryPoint(): void
-createFolderStructure(): void
-createListsFile(): void
manages
Maker
-project: Project
+checkProjectFilder(dir: File): boolean
+createProject(name: String, path: String, junit_enabled: String, create_entry_point: boolean): void
+openProject(path: String): void
+deleteProject(path_to_project: String, project_name: String): void
+clearBuildFolder(path: String): void
+getFiles(dir: String): LinkedList<String>
+addSourceFile(name: String): void
uses
CMConsole
+error_msg: String
+execute(path: String, command: String): int
+getOutput(command: String): String