MET cS665 Summer 2 2018

Abel Gancsos

Abel Gancsos productions | Hawthorne, NJ

AMGPetAgency

2018

Contents

[Introduction 2](#_Toc521198336)

[1 Application Description 3](#_Toc521198337)

[1.1 Implementation Details 3](#_Toc521198338)

[1.2 Assumptions 3](#_Toc521198339)

[1.3 Prerequisites 3](#_Toc521198340)

[2 Assignment Tasks 4](#_Toc521198341)

[2.1 Implementation Description 4](#_Toc521198342)

[2.2 UML Class Diagram 5](#_Toc521198343)

[2.3 UML Sequence Diagram 6](#_Toc521198344)

[Bibliography 7](#_Toc521198345)

[Appendices 8](#_Toc521198346)

[Appendix A – Setup 8](#_Toc521198347)

[Appendix A – Building project 8](#_Toc521198348)

[Appendix B – Running executable 9](#_Toc521198349)

# Introduction

With more and more pets being abandoned or removed from an abusive household, leading to more homeless animals, there is a higher need find adoptive parents for these pets. AMGPetAgency thus so happens to be one of these organizations that strives to place animals with the most appropriate adoptive parents based on personality traits, work schedule, family size, and overall lifestyle. However, because of the recent increase in demand, AMGPetAgency’s paper-copy approach isn’t working too well, leading to data loss, ranging from the very minor details to losing documentation. In all honesty, this issue only started recently and the president is looking for an immediate solution.

The president met up with a software developer friend and they came up with a suggestion to build a record management system in order to know which animal is placed in which home as well as which animals are still available for adoption. The end result would be a custom application that can be accessed only on the internal network from any machine and would be scalable to handle the large number of animals needed to be adopted. In order to get a better understanding of the system details, the president requested the development team to build a prototype that demonstrates the overall features that will be available in the initial release.

# 1 Application Description

## 1.1 Implementation Details

*The following details are also available in the README.md file*

This utility, AMGPetAgency, is a prototype system controller that helps keep pet adoption records safe and organized in one central location. The current prototype uses an in-memory database (collections) to store the records and is able to demonstrate the bulk of the functionality of the final product. The controller uses a command-line input mechanism to prompt the user to required details to construct a new person to add to the system or a new pet that is available for adoption. The controller then uses the array indices for the records when prompting the user to adopt a new pet or to return a pet that can no longer be kept in the current home.

During the prompt to adopt, the controller will display a list of available pets to adopt, will validate the input from the user, and will also display a list of all people that are able to adopt a pet. During the return process, the controller will ask for a confirmation if the user really wants to return the pet and will then display a list of adopted pets. If the user input is not valid, either because it’s not in a numeric form or is out of scope of the collection, a message will be displayed and/or the error will be corrected by reapplying a default value.

In the event that the prototype does become a CLI (command-line interface) for the final product, additional feature may be added, such as searching for a specific pet or a specific person. The key aspect of the controller is being able to smoothly build a complete object such as a dog or person since the rest of the operations are just input/output.

## 1.2 Assumptions

* The animal breeds have already been breaded and are available for appropriate animals.
* Animal types are predefined and no new animal can bee created via combinations.
* A person can only be either a male, female, or not specified.
* The measurement used for height is in inches.
* The measurement used for weight is pounds.
* Additional nationalities will be added in the future.
* Additional races will be added in the future.
* Additional animal breeds will be added in the future.
* Additional animal types will be added in the future.
* Additional properties will be added and be configurable to people in the future.
* Additional properties will be added and be configurable to animals in the future.

## 1.3 Prerequisites

* C++ compiler with C++11 capability
* CMake version 3.11 or higher
* XCode command-line tools (MacOS only)
* Visual Stuido 2017 or higher (Windows only)

# 2 Assignment Tasks

## 2.1 Implementation Description

*The following details are also available in the README.md file*

What are the design goals in your project?

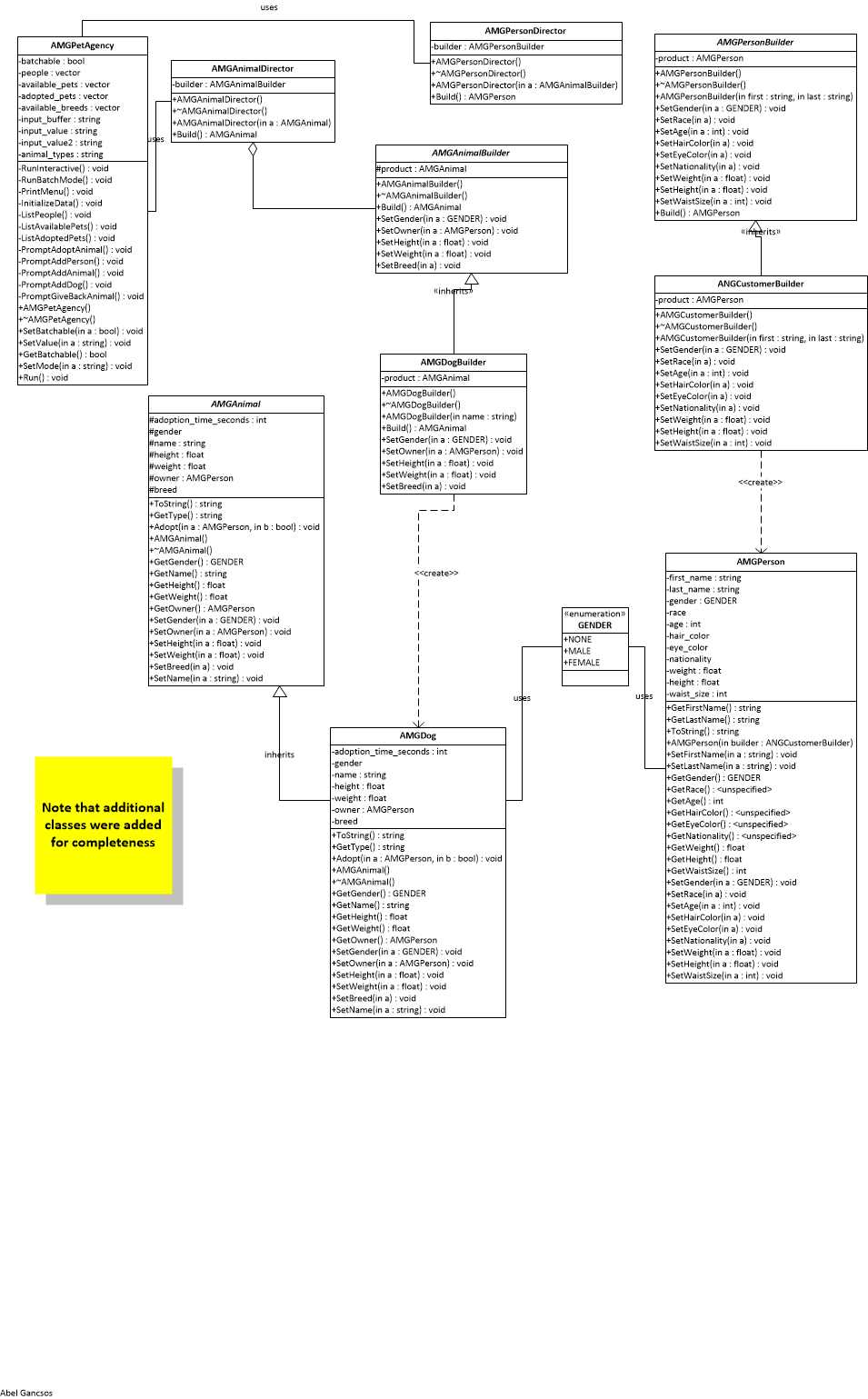
• How is the flexibility, of your implementation, e.g., how you add or remove in future new

types?

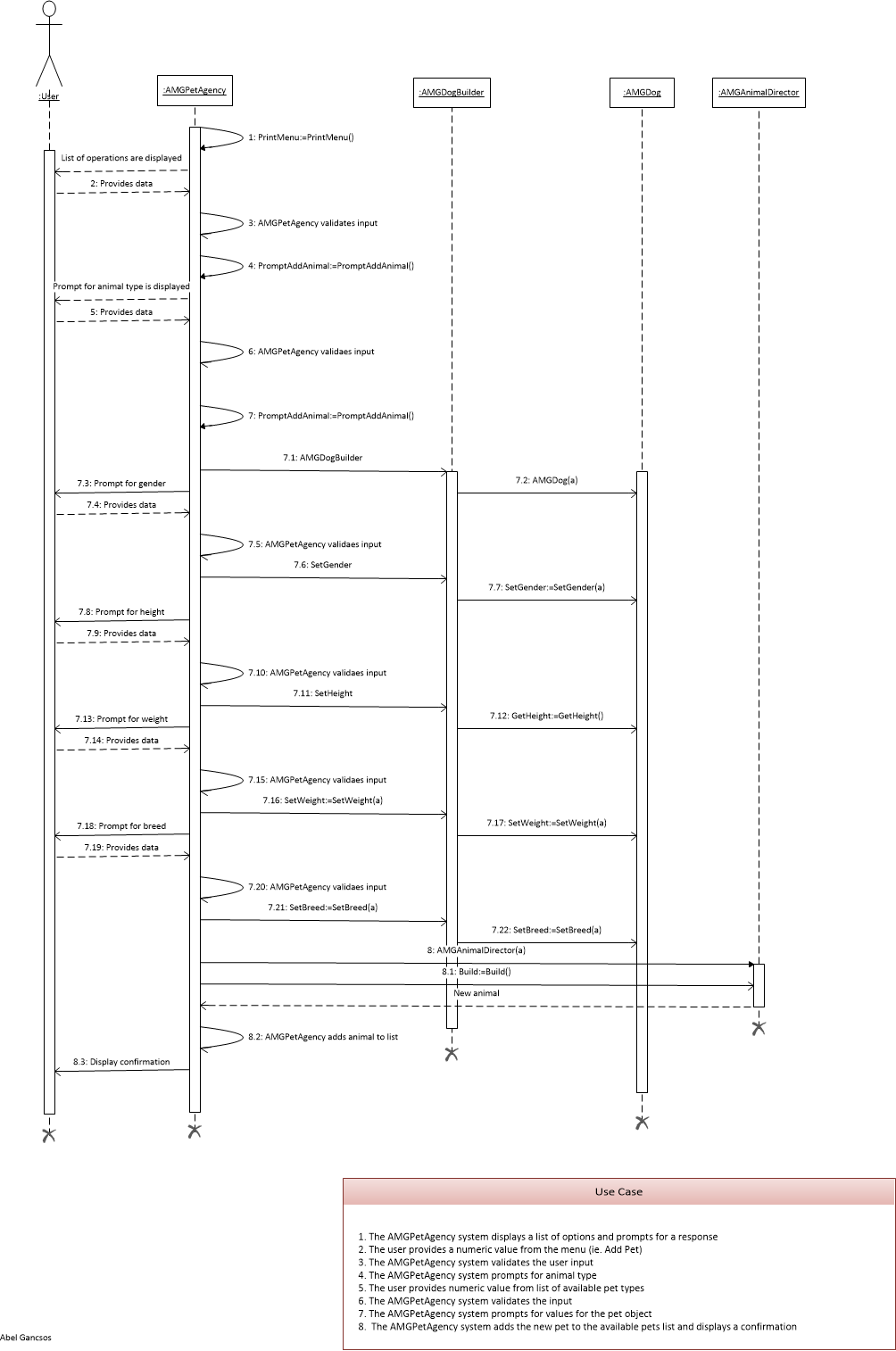
• How is the simplicity and understandability of your implementation?

• How you avoided duplicated code?

## 2.2 UML Class Diagram



## 2.3 UML Sequence Diagram



# Bibliography

Döring, A. (2016, 11 07). *minimal\_cmake\_example*. Retrieved 07 03, 2018, from GitHub Web site: https://github.com/krux02/minimal\_cmake\_example

Geeks for Geeks Organization. (n.d.). *Builder Design Pattern* . Retrieved from Geeks for Geeks Web site: https://www.geeksforgeeks.org/builder-design-pattern/

Google Corporation. (2018, 06 19). *Google C++ Style Guide*. Retrieved 07 03, 2018, from Google GitHub styles: https://google.github.io/styleguide/cppguide.html

Kitware Corporation. (n.d.). *Download*. Retrieved 07 03, 2018, from CMake Web site: https://cmake.org/download/

Lucid Software Corporation. (2017). *Class Diagram Tutorial*. Retrieved 7 25, 2017, from Lucid Software Corporation Web site: https://www.lucidchart.com/pages/uml/class-diagram

Microsoft Corporation. (2010). Visio. (2010). Redmond, WA, USA.

Moser, M. (2018, 06 28). *Patterns*. Retrieved 07 04, 2018, from Java Patterns Web site: http://java-design-patterns.com/patterns/builder/

Oracle Corporation. (2017). VirtualBox. (5.1.18). Redwood City, CA, USA.

Pazdera, R. (2011). *Example of `builder' design pattern in C++*. Retrieved from GitHub Web site: https://gist.github.com/pazdera/1121152

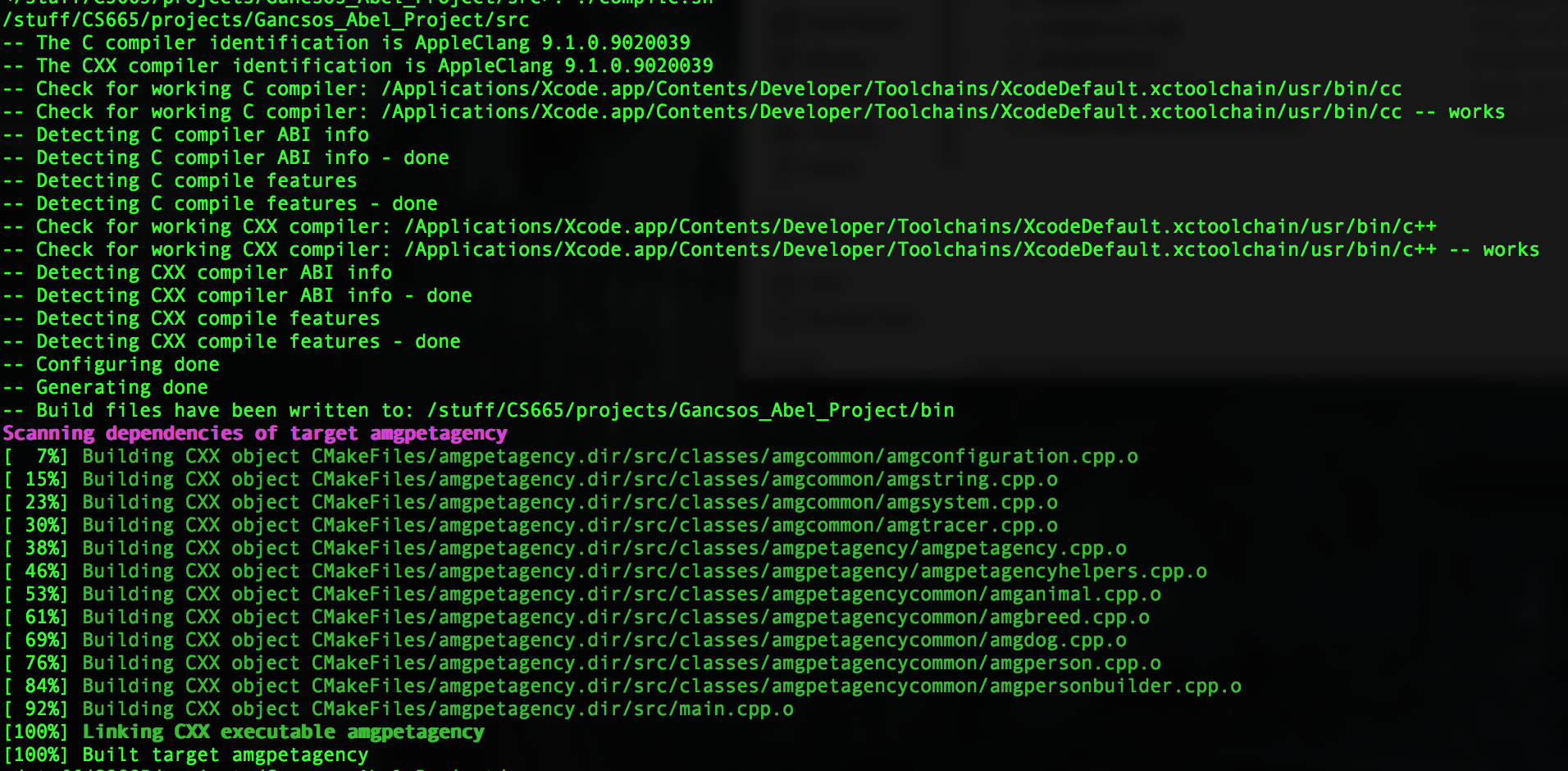
Tutorials Point. (n.d.). *Design Patterns Builder Pattern*. Retrieved 08 01, 2018, from Tutorials Point Web site: https://www.tutorialspoint.com/design\_pattern/builder\_pattern.htm

# Appendices

## Appendix A – Setup

The project can be build using the make.sh or make.bat scripts, depending on the platform. Both of these scripts will run cmake to generate the CMakeFiles and then run the make command in the bin directory. Alternatively, the project can be built manually using the CMakeLists.txt file that comes with the package.

## Appendix A – Building project



## Appendix B – Running executable

