Project 2: Data Structure with User Defined Data

Andrew G., Cody N., Caleb R.

Professor Mehri

December 14, 2019

Capital Technology University

CS-200-D01

Abstract

Introduction

Procedure

1. To start, we had a meeting to discuss the topic of the project. We decided to do a zoo app as the banking app has a lot of messy code and just a lot of code in general. It was general consensus that it would be much more efficient and faster to make a new program that we can set the requirements for. From there, we discussed the features of the program and came up with a feeding schedule for animals as well as providing information on the animals. To add even more functionality, we decided to add in an option to add an animal.
2. Next was the AGILE planning. To start with, we came up with a list of requirements we would have to complete in order to complete this project.
3. Other stuff
4. Once the planning was complete, we began working on the code itself. We split off and began working based on what we had determined through AGILE estimation to be the best places to start. We began with the class files since, despite their low BFTB value, they are required to effectively test the rest of the program; as well as the basic structure of the main method.
5. Cody puts the other stuff for main since idk what went on while I finished classes
6. The process of selecting an animal to perform actions on requires the user to input the name of the animal they desire. From there the program looks for a file that matches that name and reads the data to recreate the Animal object. Since the program won’t know what type of Animal the user selected until runtime we used an Animal pointer that points to a reference of whatever child class the user’s animal is. After the file is parsed and the child reference is created and pointed to by the Animal pointer the user is able to view the Animal’s print function and perform actions such as add it to the feeding queue or change its weight. If the user changes the Animal’s weight the Animal’s file is overwritten so that the change is persistent.
7. To keep the code organized the process of making a new animal was divided into separate methods (Fig X). The methods return either an Avian or Reptile, based on the user input, which is set to an Animal pointer, so the Animal’s print just has to be output to a file for the Animal to become persistent.

Analysis

Conclusion

References

Figures

A screenshot of a social media post

Description automatically generated

*Figure X*. Method for Adding New Reptile