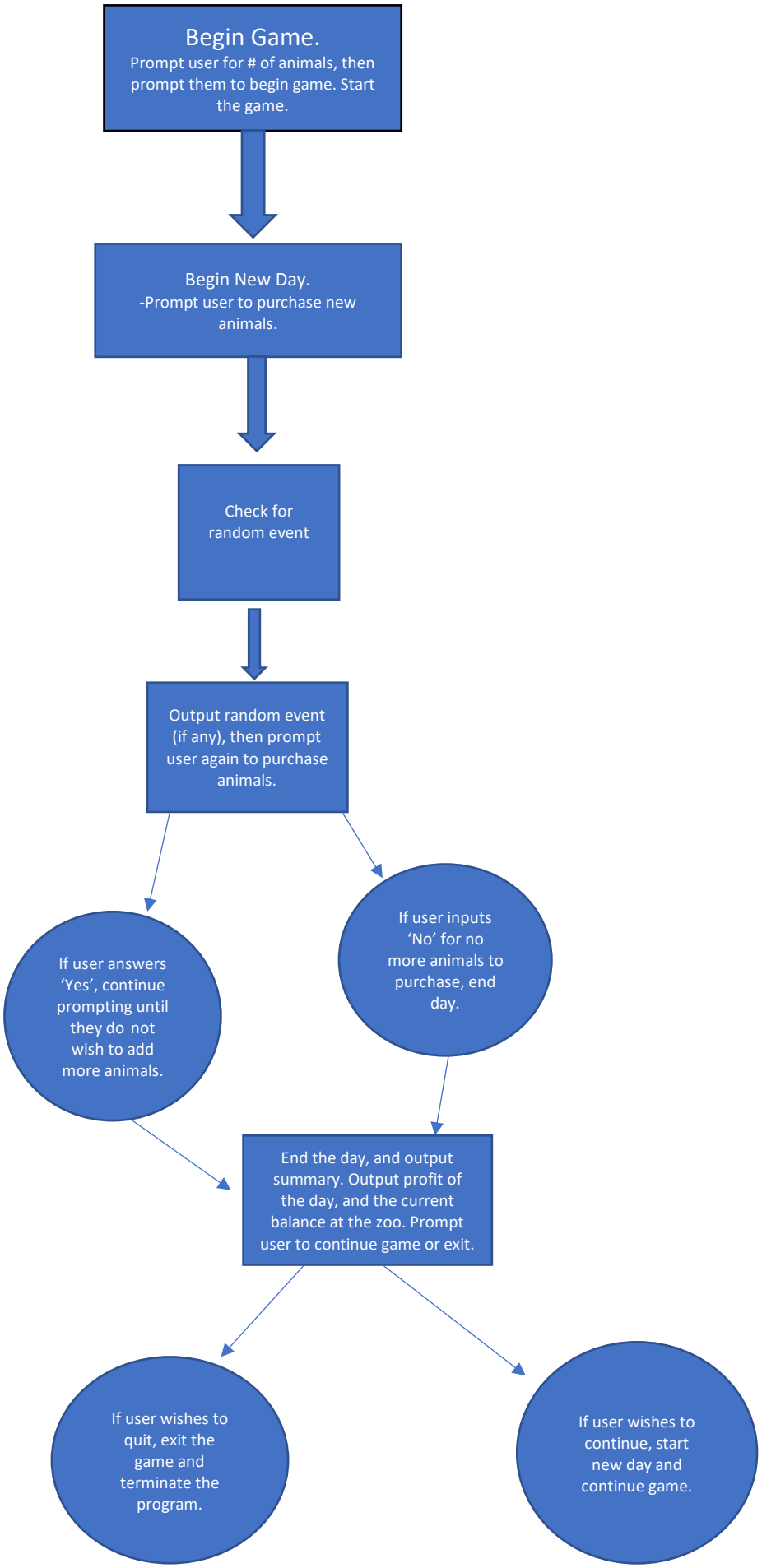
Jamie Loebe
10-21-2018
CS 162
Project 2 Reflection

Starting with Zoo Tycoon was much less intimidating than Langton's Ant was, but the introduction of inheritance did throw me for a loop. I struggled for a while trying to learn the ins and outs and get my classes to be able to inherit others. However, once I had the general idea of it down and finally got one class working to inherit the other, they seemed to fall in place after that. I wanted to design this project as simply as possible, as the game itself at first sounded incredibly complex. However, I began simply by just following the directions and setting up as much of the program as I could on a very basic level. My design was kept simple throughout and I didn't want t change much nor did I have to. I basically just wanted to be able to begin the game, prompt the user throughout, and show the user their current stats at the end of each day. A very basic game but I didn't want to overcomplicate such a large program. The tiger, penguin and turtle classes were the simplest classes to create and required a very bare minimum to start off with. Where I ran into trouble was with keeping the functions in line as to what went where. At times my program became so cluttered that by the time I went to compile I would get a myriad of errors and I had to go back one by one to reveal and then resolve each error. What helped was inputting in the description boxes above each function, so I was able to separate them evenly. Once I did this, I found it was much easier to decipher the code and any problems relating to functionality or just the fluidity of the program. I struggled also with the idea of each baby being born, and with figuring out a reliable way of checking if any animal was an adult, and if so the action of the baby being born. It took me some time to get through each function, but each time I resolved an error and got a line of code to work correctly, I tried to leave myself a comment explaining what it did, and thus providing myself a breadcrumb of what to look at and try and understand. The dynamic arrays of each animal also threw me off for a little while. I found myself combing through stack overflow and the textbook to reread and understand the concept of a pointer to a pointer, and pointers decaying to arrays. However, once I was able to piece together those ideas, I felt I had a pretty good handle on where the project was going. Doubling the array size to allow for more animals also caused me some issues as well, but the resolution of it was kind of fun for me to learn. The idea of doubling the capacity, then moving the array contents to a different array, then delete the old array and double the size of the new array was definitely a learning experience. After doing that 3 times for each animal I felt like I understood it but still find myself reading through each line when I go back through it. This program took me a considerable amount of

time to resolve and work through and tested my patience at numerous times throughout. However, with the said I feel like the learning experience I got out of it was well worth it and has helped to prepare me for more large programs in the future.

Zoo Tycoon Test Table By: Jamie Loebe

Test Case	Input Values	Driver Functions	Expected Outcomes	Observed Outcomes
Input invalid number	Input <= minimum Input >= maximum	Main() While()……..	Output “Input is Invalid!” and re-prompt user	Outputs “Input is invalid!” and re-prompts user
Input ≠num	Input ≠num	Main() while() …….	Loop back and prompt user again	Prompts user again


Begin Game.
Prompt user for # of animals, then prompt them to begin game. Start the game.
Begin New Day.
-Prompt user to purchase new animals.
Check for random event
Output random event (if any), then prompt user again to purchase animals.
If user answers 'Yes', continue prompting until they do not wish to add more animals.
If user inputs 'No' for no more animals to purchase, end day.
End the day, and output summary. Output profit of the day, and the current balance at the zoo. Prompt user to continue game or exit.
If user wishes to quit, exit the game and terminate the program.
If user wishes to continue, start new day and continue game.