Lucas Allen, 5004607031, 1004

CS 202 assignment 7: Abstract Classes and Interfaces

For this assignment I created a zoo directory. In the zoo there are four animals which each have a specific keeper(with keeper ID#), name, feeding schedule, and cage. There are three header files and one .cpp file that make up this program. These are, animals.h, hungry.h, keepers.h, and Cs202hw7.cpp.

I will walk through the function of each of these files. Animals.h contains the pure virtual class animal(){}. The subsequent derived classes are, Tiger, Penguin, Giraffe and Elephant. Each of these derived classes works off of the inherited virtual functions to return what type of animal it is and its name.

Hungry.h is a simply a file that contains the Food class. The food class contains the public noFood and feedEm functions. These functions output status messages to the screen. In the main .cpp program these functions work to tell the user whether or not the animal in question needs to be fed. I will cover the implementation of these functions when I discuss Cs202as7.cpp.

Keepers.h was by far the most difficult to properly implement and did not end up having all the functionality I originally intended due to time constraints. This being said, keepers.h contains the abstract class keeper. Within keeper is the virtual void id() function and the void printCages() function. By default, the virtual class id outputs to the screen, "DATA NEEDED: keeper name and ID # not on file." Subsequent derived classes are keeper_1, keeper_2, keeper_3, and keeper_4, and grandChild. Each keeper_# class contains a constructor of the same name, a printCages function, and a void id function. The constructor initializes a classes' specific cage variable to its matching number (i.e the cage variable for keeper_1 is cage1, and its integer value is 1). The printCages function in each derived class outputs that keepers unique name and ID number. The only derived class which does not contain this function is keeper_2. Because of this, when called, the default id function is output which informs the user there is no keeper name or ID associated with that cage. Last but not least, the grandChild class inherits from keeper_1, keeper_2, keeper_3, and keeper_4 (which each inherit from keeper). GrandChild contains the functions, getx(), gety(), setx(), sety(), and showCages().

Side-note, originally the plan was for this program to be able to swap any of the animals cages however we wanted. However, this seems like it would almost only work with pointers(if not 16 if statements) and I did not have time to make that happen. But hey, next time! Given all this, the main() function in the .cpp asks the user if they would like to switch the animals in the 3rd and 4th cages because they are both herbivores and therefore can switch. If the user answers yes then the getx(), setx(), gety(), and sety() functions are called. Getx() asks the user to input the number three to indicate their decision to switch the animal in cage three. That input is set to the variable first using setx(). Similarly, the gety() function asks the user to input the number 4 to indicate their decision to switch the animal in cage 4. If these numbers are not entered correctly an error message will be output to the screen and the program will exit. Finally, the showCages function just outputs the cage# variables derived from each of the keeper_# functions.

Here is an example of the expected output for Cs202hw7.cpp:

```
Lucass-MacBook-Air:documents lucasallen$ g++ Cs202hw7.cpp
Lucass-MacBook-Air:documents lucasallen$ ./a.out

********** ZOO DIRECTORY **********
 Current zoo population: 4
 Total Cages: 1, 2, 3, 4

********** POPULATION DATA **********
Animal #1
 Type of animal: Tiger
 Animal name: Francisco
 ** FEEDING: **
 Hours since fed: 10
 This animal doesn't need to be fed.
 ** HABITAT: **
 This animal is in cage: 1
 Keeper Name: Steve Irwin  ID #: 05472

Animal #2
 Type of animal: Penguin
 Animal name: Bernard
 ** FEEDING: **
 Hours since fed: 22
 ATTENTION: This animal needs to be fed!
 ** HABITAT: **
 This animal is in cage: 2
 DATA NEEDED: Keeper name and ID # not on file.

Animal #3
 Type of animal: Giraffe
 Animal name: Britney
 ** FEEDING: **
 Hours since fed: 11
 This animal doesn't need to be fed.
 ** HABITAT: **
 This animal is in cage: 3
 Keeper Name: Michael Scott  ID #: 42069

Animal #4
 Type of animal: Elephant
 Animal name: Peanut
 ** FEEDING: **
 Hours since fed: 20
 ATTENTION: This animal needs to be fed!
 ** HABITAT: **
 This animal is in cage: 4
 Keeper Name: Bill Gates  ID #: 10000

 ***RECORDS INDICATE ANIMALS 3 AND 4 ARE BOTH HERBIVORES AND THUS CAN SWITCH CAGES***
 Would you like to switch the cages of two animals? (y/n): y
 PLEASE CONFIRM DECISION BY TYPING "3" TO INDICATE CAGE 3: 3
 PLEASE CONFIRM DECISION BY TYPE "4" TO INDICATE CAGE 4: 4
 Cages switched.

********** ZOO DIRECTORY **********
 Current zoo population: 4
 Total Cages: 1, 2, 3, 4

********** POPULATION DATA **********
Animal #1
 Type of animal: Tiger
 Animal name: Francisco
 ** FEEDING: **
 Hours since fed: 10
 This animal doesn't need to be fed.
 ** HABITAT: **
 This animal is in cage: 1
 Keeper Name: Steve Irwin  ID #: 05472

Animal #2
 Type of animal: Penguin
 Animal name: Bernard
 ** FEEDING: **
 Hours since fed: 22
 ATTENTION: This animal needs to be fed!
 ** HABITAT: **
 This animal is in cage: 2
 DATA NEEDED: Keeper name and ID # not on file.

Animal #3
 Type of animal: Giraffe
 Animal name: Britney
 ** FEEDING: **
 Hours since fed: 11
 This animal doesn't need to be fed.
 ** HABITAT: **
 This animal is in cage: 4
 Keeper Name: Michael Scott  ID #: 42069

Animal #4
 Type of animal: Elephant
 Animal name: Peanut
 ** FEEDING: **
 Hours since fed: 20
 ATTENTION: This animal needs to be fed!
 ** HABITAT: **
 This animal is in cage: 3
 Keeper Name: Bill Gates  ID #: 10000


End of Zoo Directory. Have a nice day!
```

Everything you can see here is pretty self-explanatory. Lots of cout statements to create the directory and function calls to output data that lives inside the different header files. One interesting thing to note however, is that the hours since fed under each animal is random and will be different every time the program is run. If it has been more than 12 hours since the animal is fed then the feedEm function is called to let you know the animal is probably hungry. Otherwise, the noFood function is called. There is also some special loop magic to ensure that the program runs until the user either says no they don't want to switch an animal or the animals have been switched. The directory prints again so that the user can see that the cages have switched. After that the directory just tells you to have a nice day! Thanks for checking up on Francisco, Bernard, Britney, and Peanut!