Final Project 162

Bryan Konshak

For my final project I am making a game that involves a rabbit trying to escape from a zoo. The game begins with a battle between the zookeeper and the user's player a rabbit. The rabbit loses and starts the game escaping into the park. He or she must then accomplilsh a set of tasks to escape the park, strengthen their character, and weaken the zookeeper so the rabbit will win the rematch.

The base class will be the Zoo class, and derived classes witll be Apes, Birds, Concessions, Elephants, FrontGate, Giraffes, and Monkeys. I will have a Player class that keeps track of the movements of the player throughout the park, as well as keep track of inventory that the player collects. The game will be played in the main function.

Class Zoo—parent class
Member variable: Zoo* Left Zoo* Right Zoo* Up Zoo* Down Pointer variables to connect the spaces Zoo* Same String item Vector<string> items Bool keeperWeak Bool winGame
Class public functions: Zoo() ~Zoo() Virtual void interact() Virtual void setLocation(Zoo left, right, up, down) Virtual string getName() Virtual Zoo* moveLeft() Virtual Zoo* moveRight() Virtual Zoo* moveUp() Virtual Zoo* moveDown() Virtual string getItem() Vector<string> updateItems(vector<string>& itemIn) Virtual void displayItems() Virtual void setKeeperWeak(bool) Virtual bool getWingame()

All derived classes have their own interact() function.

Zoo* Left, Right, Up, and Down are the pointers to the adjacent spaces. Zoo* Same will be used if the direction points to NULL, then the space stays the same. String item is for inventory collected throughout the park. The vector items is a string of items in player's backpack. The bool keeperWeak

will be false, and set to true when the Apes are released. The bool winGame is false and will set to true when the player uses the final key at the front gate.

The zoo's constructor will set keeperWeak to false and winGame to false. It will also set the vector items size to 5. The interact function in Zoo.cpp doesn't do anything, but I will include a cout statement for testing.

The zoo's setLocation function will take the parameters and set them to Left, Right, Up and Down. GetName() will return the name of the zoo class, moveLeft() and all the other move functions will move to the adjacent spaces unless the space is NULL. getItem() will return a string name for an item, updateItems() will convert the player's items and return a vector. DisplayItems will display inventory. setKeeperWeak(bool zookeep) changes the bool keeperWeak when the apes are released. GetWinGame returns the winGame bool.

Class Player
Protected member variables Zoo* Location Vector<string> items
Class public functions: Player(Zoo*) ~Player() Zoo* getLocation() Void setLocation(Zoo*) Void addItem(string) Vector<string> displayItems

The Player class has a Zoo pointer named Location and a vector of items. The constructor sets the items to "EMPTY" for inventory. Then there are getter and setter functions for the locations to move the player, then addItem function to add items to inventory, and a displayItems function that returns a vector of strings.

Each "space" has a unique interaction function that runs their part of the game. These are listed in order to solve the game. They also have a bool function that once the objective is achieved in the space, it changes the interact() text.

Elephants:

Bool gotBucket() set to false;

Interact(vector<string>& inventory) gives the user four options: take the bucket, use the bucket to water the elephants, use an item, or leave. The player has to use the bucket to water the elephants, then the player gets the bucket and gotBucket is set to false. When player goes back to space the elephants are happy.

Monkeys:

Bool filledBucket set to false in constructor.

Interact()

The monkeys are throwing their feces at the player. The player must use the bucket to catch it, setting the filledBucket variable to false and changing inventory item to filled bucket, and indicating there is nothing left to do there. Player can also leave or try to block the stuff.

Concessions:

Bool line = true in constructor.

Interact() the player must use the filled bucket to get rid of the long line. This sets line to false, and the line leaves, and allows to get the snack item.

Giraffes:

Bool hungry = true in constructor

Player must feed the giraffes with the Snack, then they get a key, and hungry = false.

Apes:

Bool caged = true.

Use the key to release the apes, who weaken the zookeeper when they leave the park and caged = false.

Birds

Bool training = false

In interact() if keeperWeak = true (apes escape) and training = false then the eagles train the player for the final fight, giving the player the final key and the flying eagle training certificate. If the player hasn't helped the apes, the birds just fly overhead.

FrontGate:

You access the FrontGate once you get the key from the birds. The final fight takes place, the rabbit wins, and the game is over with bool winGame= true.

In main() I will create all the pointers to the different rooms, then use setLocation to set up the park as well as a Player player.

I'll have a vector of strings for inventory, and bool gameOver = false, gameWin= false, and overCount= 0. The game runs while gameOver = false and gameWin = false. gameOver switches to true after 30 turns.

gameWin switches to true when the bool winGame is set to true.

The loop displays a menu on what direction the player wants to go and displays a map. The player automatically interacts when moving to a new space. So if player selects move left, and left is not NULL via player.getLocation()→moveLeft then player's setLocation ues moveLeft(). Then getLocation is used to run interact(). One is added to overcount each round.

Test Plan Passed Failed

Set up pointers in constructors		Can't do this, have to use setLocation
Create setLocation functions	Passed	
Test MoveLocation functions	Passed	
Test player.getlocation→move	Passed	
Test player.setLocation(move)	Passed	
Test overCount	Passed	
Create and delete all pointers	Passed	
Test interact function in Ape	Passed	

Test Birds interact()		Failed—had created the keeperWeak but it changed based on Apes, so I had to run a function in Apes that changed keeperWeak for Birds (based on location down)
Test Concessions	Passed	
Test Giraffes	Passed	
Test FrontGate		Had to create a key in birds because I didn't have an easier way to change the bool I had created for frontGate
Test Elephants	Passed	
Test Monkeys	Passed	
Test input validation in menu and selections in interact()	Passed	
Test the count to end the game	Passed	

Test inventory update after interact()	Passed	
Test getWinGame()	Passed	
Test vector updating	Passed	

Reflection:

I enjoyed this project. It took a long time to get my pointers working because I thought I could make them work in the constructors of the classes, but that didn't work, it created segmentation faults. The player class worked flawlessly, and the practice in returning Zoo pointers using the player class and using that to set the location and get the location drove home some key concepts I was struggling with. Once I got the pointers set up, and then I was able to use the player class to move around the world, it was time to set up the various interact functions. Everything worked great there except I probably should have used the player class to set up the bools for the weakened zookeeper and initially I had a bool to enter the front gate. I changed that to a key. And I also changed where my apes exhibit was so I could

use a down pointer to update the bool of weakened zookeeper. Then that triggered the interaction in Birds.

This project improved my proficiency in pointers, linking spaces, using bools to manipulate information, using a class's functions to return another class's objects, and also I finally created a good standalone menu function.