The Object of Horses (A4)

2/3/2023

79/100 Points

Attempt 1

Review Feedback
2/3/2023

Attempt 1 Score: **79/100**

View Feedback

Anonymous Grading: **no**

Unlimited Attempts Allowed

Details

Overview

Use the principles of Object Oriented Design (OOD) and Object Oriented Programming (OOP) to re-build the horseRace assignment using object-oriented programming. Each horse will be an object, and the race will be another object that contains a list of horses and manages the race. Along the way, you will experiment with UML, create classes, review access modifiers, build member variables, add access methods, and create constructors.

The Project

Build a program that simulates a horse race. The race works just like the last assignment, and the user may not ever see the difference. But this time, the underlying design will use objects. You will have two objects in the game. There will be an array of five horse objects. The horse will 'know' how to advance according to a random flip, and to return its position. The other primary entity in this game is the race (or the track, if you prefer.) This object will contain a series of horses. It will also have the ability to start the race. When the race is running, the race class will 'tell' each horse to advance, and will print out a track diagram showing the relative positions of the horses.

In this case we may do automated testing against your program. In order to accommodate this you will prompt (ask for input) for a random seed and use it to seed your random number generator. This will allow you to test your program against several seeds. And we may test your program with a series of these random seeds. While this will not be your entire grade, but your ability to pass these tests could be a consideration in your grade. This is a common industry practice and one we will try to introduce you.

Sample Run

You can find an HTML5 version here:

http://cs.iupui.edu/~ajharris/240/horseRace.html (http://cs.iupui.edu/~aharris/240/horseRace.html)

Note that you will not need the automation feature nor a GUI. This just gives you an idea of the race concept.

Here is a sample run of the program:

```
Please enter a random seed: 700
.0............
.1............
.2............
3.............
4.............

..0...........
..1...........
..2...........
```

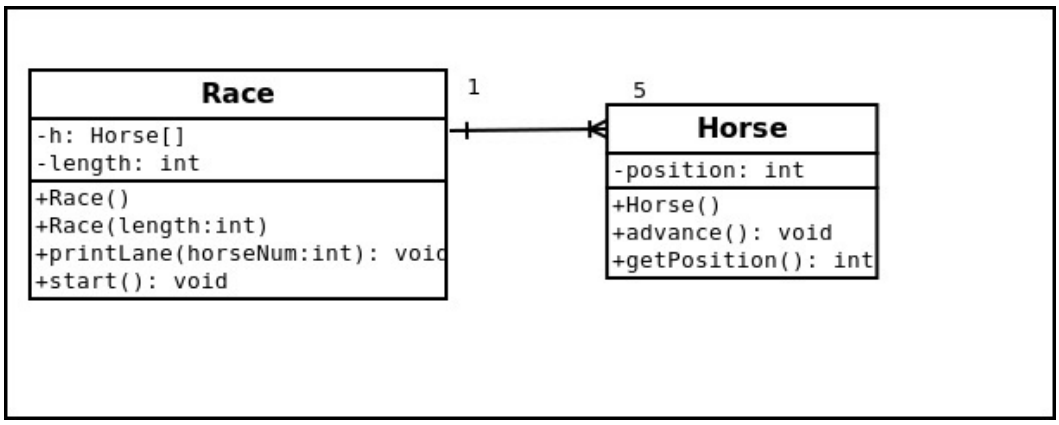
```
.3............
.4............


...0..........
..1...........
...2..........
.3............
.4............


...0..........
..1...........
....2.........
.3............
.4............


...0..........
..1...........
....2.........
..3...........
..4...........


...0..........
...1..........
....2.........
...3..........
...4..........


...0..........
...1..........
....2.........
....3.........
...4..........


...0..........
...1..........
.....2........
.....3........
...4..........


...0..........
...1..........
.....2........
......3.......
....4.........


...0..........
...1..........
......2.......
......3.......
....4.........


....0.........
...1..........
.......2......
.......3......
.....4........


....0.........
...1..........
.......2......
........3.....
.....4........


.....0........
...1..........
.......2......
........3.....
......4.......
```

```
.....0.........
....1..........
.......2.......
.........3.....
......4........


.....0.........
....1..........
........2......
..........3....
......4........


.....0.........
.....1.........
........2......
...........3...
.......4.......


.....0.........
......1........
.........2.....
............3..
.......4.......


.....0.........
.......1.......
.........2.....
............3..
.......4.......


.....0.........
........1......
..........2....
............3.
.......4.......


......0........
.........1.....
...........2...
............3.
.......4.......


.......0.......
.........1.....
...........2...
.............3
.......4.......


........0......
..........1....
............2..
.............3
........4......


.........0.....
..........1....
............2..
.............3
........4......
```

```
.........0.....
..........1....
............2.
..............
........4......

Horse 3 wins!
```

Code Organization

While you have already written this program in a procedural fashion, this program is a perfect candidate for the object-oriented paradigm. Please use the UML diagram I created as a starting point:



Programming Methodology

This program must compile on Tesla. You will need to create a header file and a cpp file for each object, as well as a main.cpp. You will also need to create a make file to compile and run your program. You will be turning this program in using the IU github system.

It is not necessary to create a GUI for this project. Since this is an exercise in OOP programming, you are expected to create at least two classes, as indicated in the class diagram provided above. All the methods and member variables necessary are listed in the UML diagram, but you might need to create other local variables inside your methods. There are other legitimate ways to organize this data, but this is one of the simpler techniques.

Tips

- You will need two classes: each will have a header and a cpp file.
- You will also need a main.cpp file.
- do not include namespace std
- Your make file should have a target for each object as well as the main target to build the final executable.
- Your make file should also include clean and run targets.
- The horse class will need to utilize random number generation. I included an example of this with the last homework exercise, but you may still need to look up how to generate a random number in C++. Don't forget the srand() function to ensure you get a different random sequence on each run of the program.
- Each class will need a simple constructor. (My version of Race actually has two constructors.) Remember that every class should have a null parameter constructor. Also, remember that the main point of a constructor is to initialize the member variables. You might additionally do some other initial work that happens only once in the constructor. (That was a hint.)

- Each method should represent something the class can DO. Each member variable should be information that the data has.
- This program uses an aggregate data structure. One of the classes includes an array of instances of the other class.
- The Race class' start() method (at least in my version of the program) controls most of the race action. It not only starts the race, but it controls the entire race.
- The Race::printLane() method is used to print out a visual depiction of the lane for a particular horse. This method is easy to visualize, but it can be tricky to implement, because the horse ID will probably be an integer, and the rest of the track will likely be characters. Think about how clever use of std::cout and the << operator might simplify this situation.

Algorithm

As always, begin with an algorithm in markdown this time. This document should begin by describing the main purpose of the program using the Goals - Input - Output - Steps technique we've been using. For each function you have described in the main program, you must go through the same process. Remember that your algorithm should be complete enough that by the time you are ready to start writing code, you pretty much know what you are going to do. You should get mostly through the algorithm step together in the recitation center, but definitely try to write this on your own as well. Your algorithm file should be named algorithm.md.

Points will be taken off if it is not named this exactly, or if it is in another file type (including but not limited to; .rtf, .docx, .doc, .pdf, ... etc.)

Turning in the project

- This assignment will be turned in through iu github. Please name your repo CSCI24000_spring23_A4
- If your repo has another name, it will not be graded.
- After you've submitted your assignment in Github, come back to this assignment page and submit the full Github URL (**https://github.iu.edu/username/reponame** **(https://github.iu.edu/username/reponame)**) for your repo here.
- Your repo must be private. We will not grade projects in public repos, and we will require you to take down any homework assignments in a public repo.
- Please ensure to add all the following collaborators:
- **Git Collaborators (https://iu.instructure.com/courses/2131288/pages/assignments-submission-and-grading)**

Note that your code will be tested with an online plagiarism tool. Please DO NOT turn in work that is not yours. We will know, and we will be displeased.

Black Belt Extensions

The black belt potential for this project is enormous. As always, begin with the basic expectations. Once you get the core program working, there are many ways to extend it. Here are a few options:

- Create a new type of horse that advances on a percentage rather than a simple coin flip. This could be an extension of the basic Horse class. For the constructor, include an int parameter that indicates the percentage of time the horse will move forward. You will need to modify the advance() method to handle the more elaborate movement mechanism. You may want to then have horses with differing abilities in the same race.
- Implement a betting mechanism. Allow the user to bet on a race before it begins. Keep track of the user's cash through several runs of the program.
- Give horses with different abilities different odds. Reward a long shot (40% chance of advancing each round) much more than a favorite (50% chance of advancing.)

- Multi-player - Allow more than one player to place bets, keeping track of each player's performance
- Bribery. Allow user to 'purchase' performance modifications. This will increase or decrease a particular horse's performance during a race, but there is a chance the player will be caught and immediately lose all his money.
- Cosmetic changes. Add a menu system, GUI, or other features to make the game more aesthetically appealing. Try writing the game in another language.
- For submission, put all files on the master branch. Give the folder with the base assignment the name *base*. For any blackbelt, create a folder with the name *blackbelt*. Any sub-folders in there will be pulled, organize it to your heart's desire. For any program that requires special instructions, such as using some esoteric language not on Tesla, include a README which explains how to run your program.

Supporting Materials

- **oophorse.jpg** (https://iu.instructure.com/courses/2131288/files/150891090/preview?hidden=1)

---

https://github.iu.edu/parmsing/CSCI24000_spring23_A4

(https://iu.instructure.com/courses/2131288/modules/items/28790594)

(https://iu.instructure.com/courses/2131288/modules

You are unable to submit to this assignment as your enrollment in this course has been concluded.