

All the variables except fallen is set as private. This prevents them from being accessed by the other classes.

A screen shot of a computer program

Description automatically generatedA screen shot of a computer program

Description automatically generated

The methods getConfidence, getDistanceTravelled, getName and getSymbol are the getter methods. They are used to retrieve the associated value of the variables.

The methods setConfidence and setSymbol are the mutator methods. We need setConfidence to ensure that the confidence level is always between 0 and 1. This is why we use encapsulation to hide that implementation that would be repetitive if we did not use it. It also ensures that the program runs smoothly. A negative confidence level could lead to unintended scenarios.

Testing:

A screen shot of a computer screen

Description automatically generated

I used this to test the Horse Class. The printHorseInformation method is used for abstraction purposes.

When attempting to test, I ran into a few errors.

A screen shot of a computer

Description automatically generated

The symbols only show up as “?” while the setConfidence method works incorrectly.

The issue with the setConfidence method is that I implemented it incorrectly. It does not change the confidence level when it’s not between 0 and 1.

A computer screen shot of code

Description automatically generatedA computer screen with text and numbers

Description automatically generated

This was an easy fix. All I needed to do was remove the last else and take the assignment out of the brackets so no matter what it assigns the parameter to

### Race Class

A screen shot of a computer

Description automatically generatedA screen shot of a computer code

Description automatically generated

I changed the if statement to the second to ensure that it displays who wins the race at the end of the race.

There was an issue where it would not show when a force had fallen so I tried this:

A computer screen shot of text

Description automatically generated

However, this did not work.

A computer screen shot of text

Description automatically generatedA screen shot of a computer code

Description automatically generated

I realised this might have been the issue. Therefore, I changed \u2322 to just X as shown on the left.

A screenshot of a computer

Description automatically generated

This led to this. As you can see the falling now is working as intended. However, there are still some issues. Certain Unicode symbols do not work for some reason. This will be fixed later. For now, I will change them to just letters. Furthermore, the confidence level and names are not displayed. The winner at the end is at least displayed.

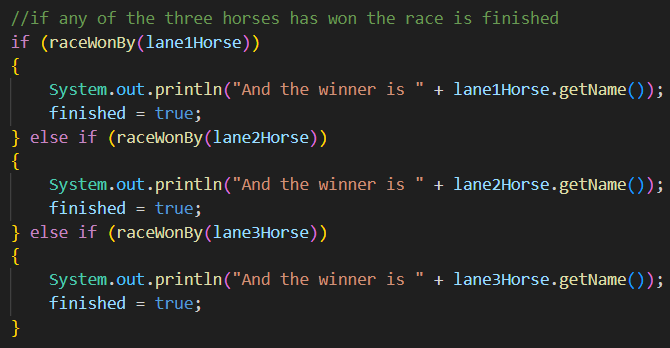
A black background with white text

Description automatically generated

I also found this minor issue where it would always say the first horse (called Glitterhoof here represented by G) won. This was an easy fix.

A screen shot of a computer code

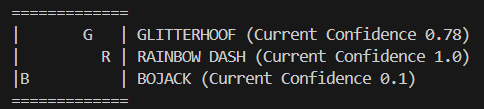
Description automatically generated



All I needed to do was change the variable I was printing.

A screen shot of a computer program

Description automatically generated



Adding a print statement at the end of the printLane() method allowed me to show the information I needed to. Progress was being made.

Another issue I witnessed was that when the horses all fell the game would continue when it should stop.

A screen shot of a computer screen

Description automatically generated

I added this to the startRace method to deal with those kinds of situations.

A screen shot of a computer

Description automatically generated

It led to this result.

A computer screen with text

Description automatically generated

I added this to make sure that changing confidence is always appropriate.

With the error of the horse symbols being “?”s and the terminal not clearing, the fix was simply changing to a different IDE.