Encapsulation Explanation

Strong encapsulation in the Horse class is achieved through several specific mechanisms.

Private Fields:

The Horse class uses private declaration for all its instance variables so they remain accessible only within the class.

horseSymbol()

horseName()

horseConfidence()

distanceTravelled()

isFallen()

isWinner()

Accessor Methods (Getters):

The accessor methods provide read-only access to the private fields of the class.

getConfidence() - Returns the horse's confidence level

getDistanceTravelled() – returns the total distance the horse has moved.

getName() - Returns the horse's name

getSymbol() - Returns the horse's symbol

getIsWinner() Returns if the horse is a winner (isWinner private var)

hasFallen() - Returns whether the horse has fallen

Mutator Methods (Setters):

The following mutator methods provide controlled means to update the private fields.

setConfidence(double newConfidence) - Updates the horse's confidence level

setSymbol(char newSymbol) - Updates the horse's symbol

setIsWinner(boolean value) - Updates the winner status

moveForward() - Increments the distance travelled

fall() changes the horse's fallen status to true

goBackToStart() repositions the horse while clearing its fallen status.

Data Protection Mechanisms

Controlled Access:

Each field remains inaccessible outside the class because they are private.

Data accessibility requires usage of predefined public methods.

State Management:

The moveForward() method enables a horse to move forward by one unit of distance.

The fall() method exclusively sets the fallen status to true.

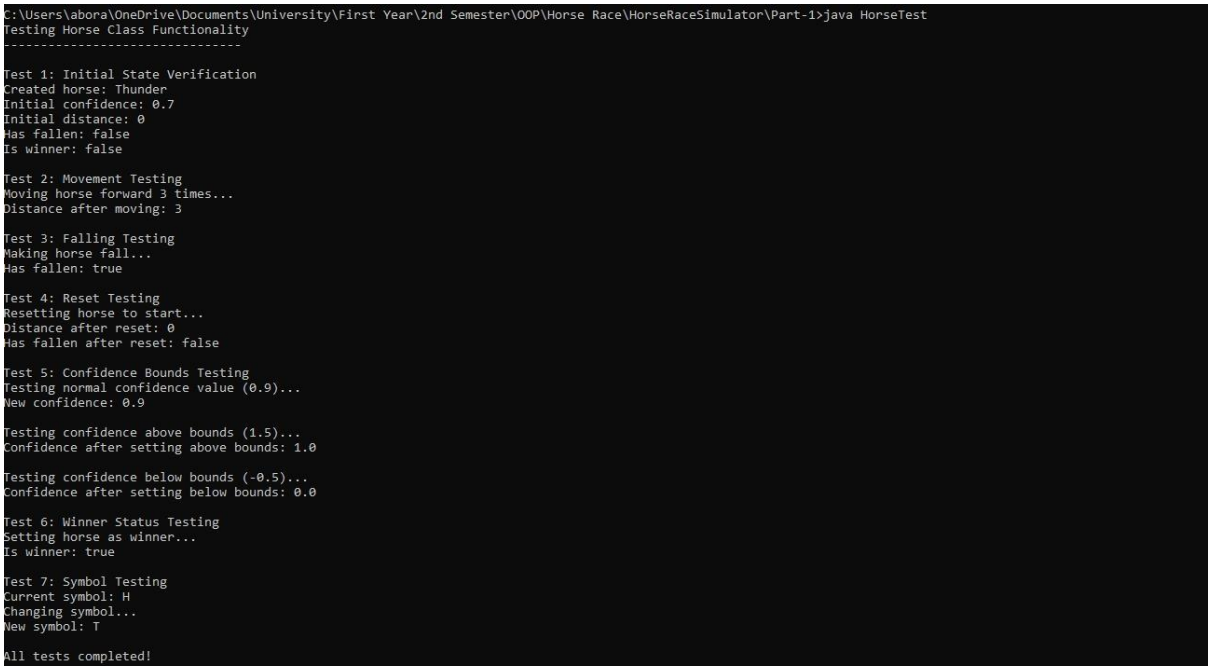
The goBackToStart() method allows for a controlled process to reset the horse's state.

Data Validation:

I've added validation to the setter **setConfidence(double newConfidence)** method by checking if the provided double is within the acceptable bounds (0-1) if it is allow setting the value, otherwise if over 1 set to one if below 1 set to 0

Testing

```java
public class HorseTest {
    public static void main(String[] args) {
        System.out.println("Testing Horse Class Functionality");
        System.out.println("--------------------------------");

        // Test 1: Create a new horse and verify initial state
        System.out.println("\nTest 1: Initial State Verification");
        Horse testHorse = new Horse('H', "Thunder", 0.7);
        System.out.println("Created horse: " + testHorse.getName());
        System.out.println("Initial confidence: " + testHorse.getConfidence());
        System.out.println("Initial distance: " + testHorse.getDistanceTravelled());
        System.out.println("Has fallen: " + testHorse.hasFallen());
        System.out.println("Is winner: " + testHorse.getIsWinner());

        // Test 2: Test movement functionality
        System.out.println("\nTest 2: Movement Testing");
        System.out.println("Moving horse forward 3 times...");
        testHorse.moveForward();
        testHorse.moveForward();
        testHorse.moveForward();
        System.out.println("Distance after moving: " + testHorse.getDistanceTravelled());

        // Test 3: Test falling functionality
        System.out.println("\nTest 3: Falling Testing");
        System.out.println("Making horse fall...");
        testHorse.fall();
        System.out.println("Has fallen: " + testHorse.hasFallen());

        // Test 4: Test reset functionality
        System.out.println("\nTest 4: Reset Testing");
        System.out.println("Resetting horse to start...");
        testHorse.goBackToStart();
        System.out.println("Distance after reset: " + testHorse.getDistanceTravelled());
        System.out.println("Has fallen after reset: " + testHorse.hasFallen());

        // Test 5: Test confidence modification and bounds checking
        System.out.println("\nTest 5: Confidence Bounds Testing");
        System.out.println("Testing normal confidence value (0.9)...");
        testHorse.setConfidence(0.9);
        System.out.println("New confidence: " + testHorse.getConfidence());

        System.out.println("\nTesting confidence above bounds (1.5)...");
        testHorse.setConfidence(1.5);
        System.out.println("Confidence after setting above bounds: " + testHorse.getConfidence());

        System.out.println("\nTesting confidence below bounds (-0.5)...");
        testHorse.setConfidence(-0.5);
        System.out.println("Confidence after setting below bounds: " + testHorse.getConfidence());

        // Test 6: Test winner status
        System.out.println("\nTest 6: Winner Status Testing");
        System.out.println("Setting horse as winner...");
        testHorse.setIsWinner(true);
        System.out.println("Is winner: " + testHorse.getIsWinner());

        // Test 7: Test symbol modification
        System.out.println("\nTest 7: Symbol Testing");
        System.out.println("Current symbol: " + testHorse.getSymbol());
        System.out.println("Changing symbol...");
        testHorse.setSymbol('T');
        System.out.println("New symbol: " + testHorse.getSymbol());

        System.out.println("\nAll tests completed!");
    }
}
```



Here, I started by creating an object Horse and setting the name to Thunder the symbol to H and confidence to 0.7, then I verified that all initial values were correctly set:

name was correctly stored, confidence was within bounds initial distance was 0 and horse was not fallen and not a winner

this test proves that the constructor works perfectly.

Then for movement, I tested the moveForward () method by calling it three times and verified that the distance increased by 1 each time, and I also confirmed the distance counter works correctly. This test validates the basic movement functionality and it works as expected.

For falling mechanism, I tested the fall() method and verified that hasFallen() returns true after calling fall(), I confirmed the horse's fallen state is peroperly tracked, so this test ensures falling mechanism works perfectly.

For reset functionality, I tested the goBackToStart() method and verified it by checking the distance to be 0 and fallen status to be reset and it confirmed that the horse can be properly reset to its initial state.

For confidence bounds testing, I tested three confidence scenarios 0.9 (normal) 1.5 (over the limit) and -0.5 (below limit) and in all three cases the behaviour was correct, in normal case it was kept in over it was capped at 1 and in lower it was capped at 0

For winner status I tested the functionality by verifying that setIsWinner() and getIsWinner() work correctly, and confirmed the winner status can be peropely set and retrieved which can be seen in the output

Finally for symbol testing I tested the symbol modification and verified that setSymbol and getSymbol work correctly, and confirmed that the horse's symbl can be changed and retrieved.

These tests results show that all methods work as intended, data validation is working, state changes are properly tracked and that all horses can be reset to initial state and finally all getters & setters work.