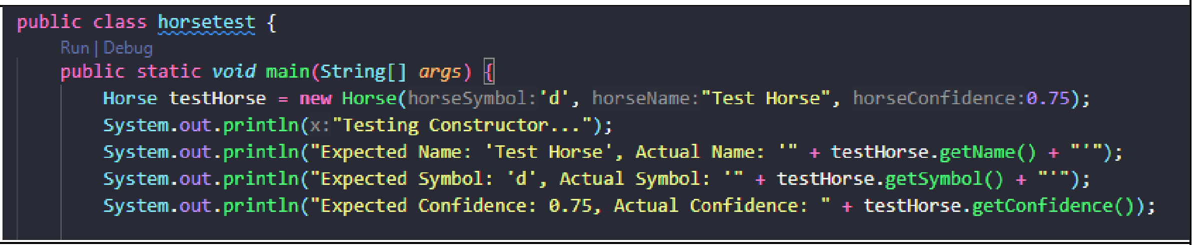
ENCAPSULATION:

Encapsulation is the practise of restricting access to a classes' internal fields and the only way to access the fields is by using specific methods such as accessor methods. I have successfully used encapsulation in my horse class to help secure data by restricting levels of access to certain methods and attributes. I have done this by declaring all my instance variables in the horse object as private for example, “name”, “symbol”, “hasFallen” and “confidence” as this makes sures that these fields cannot be access or altered outside the class without using specific methods such as accessor methods (getter methods) and mutator methods (setter methods). Accessor methods helps retrieve and access data in my horse class without modifying it. The accessor methods in my horse class include “getName()”, getDistanceTravelled()”, “hasFallen()”, and “getConfidence()”. These methods help other parts of the program to read the data stored in the class. I have also used mutator methods to help me modify internal data stored in the horse class. These methods include, fall(), moveForward(), goBackToStart() and setConfidence(double). These methods helps me change and modify the private fields of data stored in the horse class.

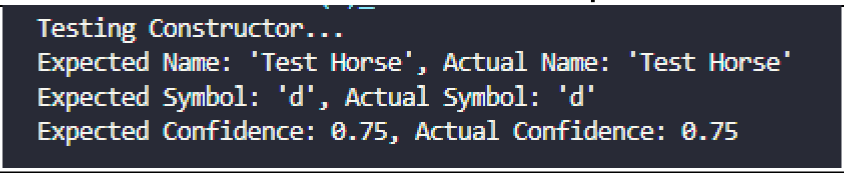
Task 1 – Horse Class

Test Code:



```
public class horsetest {
    Run | Debug
    public static void main(String[] args) {
        Horse testHorse = new Horse(horseSymbol:'d', horseName:"Test Horse", horseConfidence:0.75);
        System.out.println(x:"Testing Constructor...");
        System.out.println("Expected Name: 'Test Horse', Actual Name: '" + testHorse.getName() + "'");
        System.out.println("Expected Symbol: 'd', Actual Symbol: '" + testHorse.getSymbol() + "'");
        System.out.println("Expected Confidence: 0.75, Actual Confidence: " + testHorse.getConfidence());
```

Output:



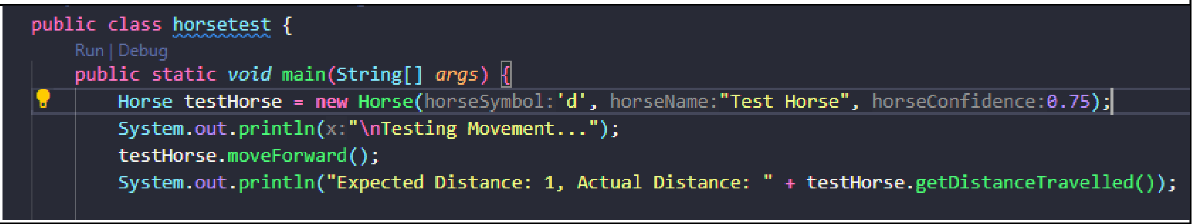
```
Testing Constructor...
Expected Name: 'Test Horse', Actual Name: 'Test Horse'
Expected Symbol: 'd', Actual Symbol: 'd'
Expected Confidence: 0.75, Actual Confidence: 0.75
```

Explanation:

Here I am testing the horse class to see if the constructor is correct. I have used getter methods to see if the values I set in “testHorse” can be retrieved and if it contains the same values I set in testhorse. For example, I have used getName() method to see if it will return the name “Test Horse” and it was successful as it did retrieve the correct name. I have tested the getConfidence() method which should return 0.75 and it did return the same value resulting in a successful test.

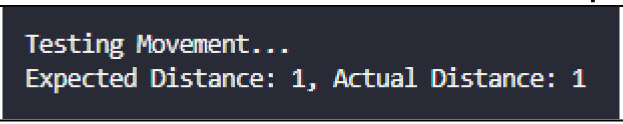
Task 1 – Horse Class

Test Code:



```
public class horsetest {
    Run | Debug
    public static void main(String[] args) {
        Horse testHorse = new Horse(horseSymbol:'d', horseName:"Test Horse", horseConfidence:0.75);
        System.out.println(x:"\nTesting Movement...");
        testHorse.moveForward();
        System.out.println("Expected Distance: 1, Actual Distance: " + testHorse.getDistanceTravelled());
```

Output:



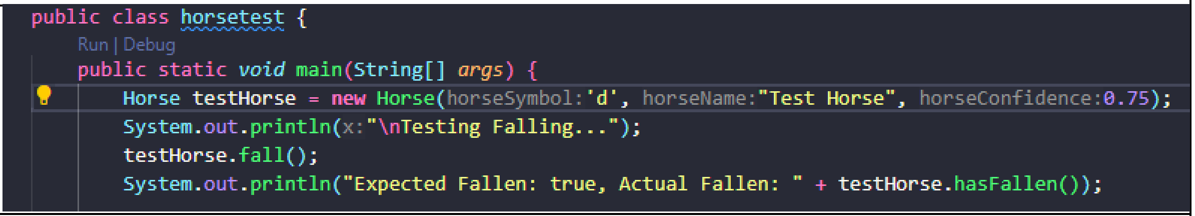
```
Testing Movement...
Expected Distance: 1, Actual Distance: 1
```

Explanation:

Here I am testing the moveForward() method to see of the distance increments by 1. And the test was successful as it increments the distance by 1. The moveFoward() method contains an increment counter which takes the initial value which is 0 here as the horse has not moved yet and increments it by 1 every time the horse moves so that's why the method printed the distance as 1.

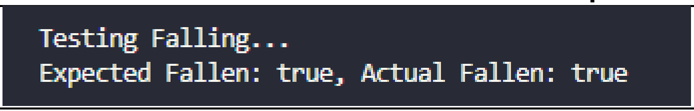
Task 1 – Horse Class

Test Code:



```
public class horsetest {
    Run | Debug
    public static void main(String[] args) {
        Horse testHorse = new Horse(horseSymbol:'d', horseName:"Test Horse", horseConfidence:0.75);
        System.out.println(x:"\nTesting Falling...");
        testHorse.fall();
        System.out.println("Expected Fallen: true, Actual Fallen: " + testHorse.hasFallen());
```

Output:



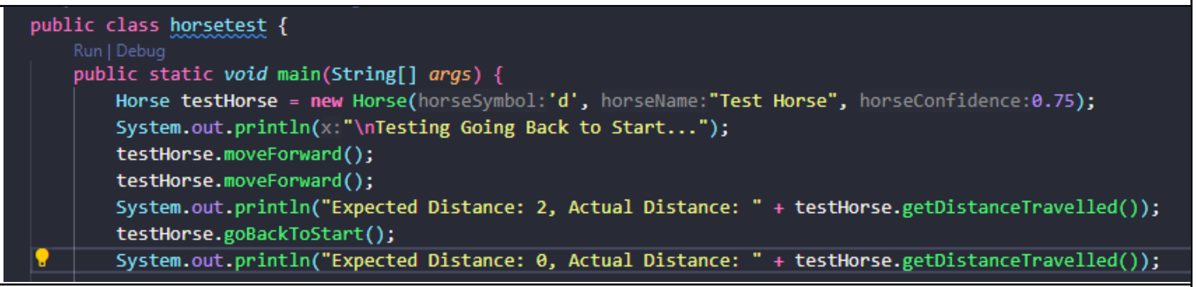
```
Testing Falling...
Expected Fallen: true, Actual Fallen: true
```

Explanation:

Here I am testing the fall() method to see when the method is called does the horse fall. From calling the fall() method and then calling the hasFallen() method we can see the Boolean variable for the hasFallen in the horse class is set to true which mean the test is successful as the horse has fell.

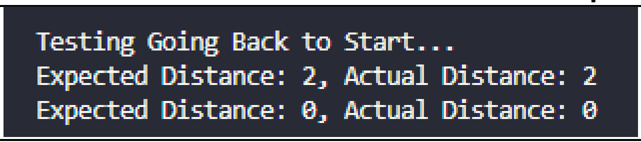
Task 1 – Horse Class

Test Code:



```java
public class horsetest {
    public static void main(String[] args) {
        Horse testHorse = new Horse(horseSymbol:'d', horseName:"Test Horse", horseConfidence:0.75);
        System.out.println(x:"\nTesting Going Back to Start...");
        testHorse.moveForward();
        testHorse.moveForward();
        System.out.println("Expected Distance: 2, Actual Distance: " + testHorse.getDistanceTravelled());
        testHorse.goBackToStart();
        System.out.println("Expected Distance: 0, Actual Distance: " + testHorse.getDistanceTravelled());
```

Output:



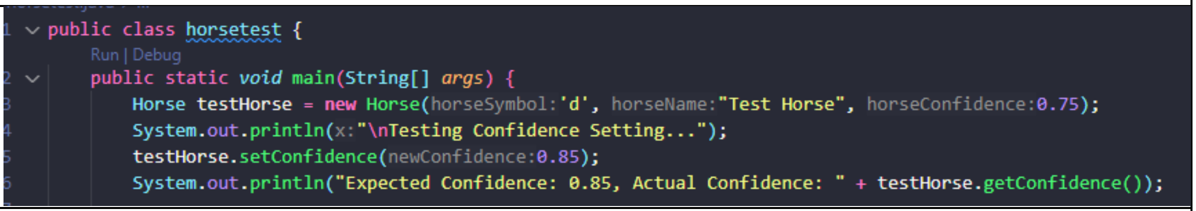
```
Testing Going Back to Start...
Expected Distance: 2, Actual Distance: 2
Expected Distance: 0, Actual Distance: 0
```

Explanation:

Here I am testing the gobacktostart() method which should make the horse reset its distance and make the horse start the race again. I have called the moveFoward() method twice and displayed the distance as 2 and then when the goBackToStart() method is called it resets the distance to 0 meaning the test was successful.

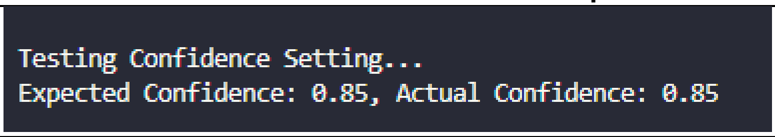
Task 1 – Horse Class

Test Code:



```java
public class horsetest {
    public static void main(String[] args) {
        Horse testHorse = new Horse(horseSymbol:'d', horseName:"Test Horse", horseConfidence:0.75);
        System.out.println(x:"\nTesting Confidence Setting...");
        testHorse.setConfidence(newConfidence:0.85);
        System.out.println("Expected Confidence: 0.85, Actual Confidence: " + testHorse.getConfidence());
```

Output:



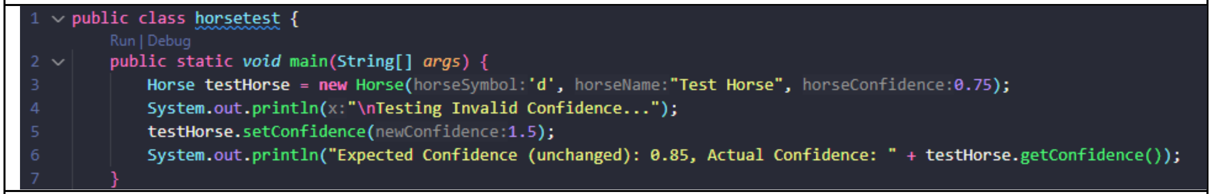
```
Testing Confidence Setting...
Expected Confidence: 0.85, Actual Confidence: 0.85
```

Explanation:

Here I am testing the setConfidence() method. The original confidence rating was already set as 0.75 and I set a new confidence rating as 0.85 and I printed out getConfidence() method to see if it will print out the new confidence rating set and it did meaning the test was successful.

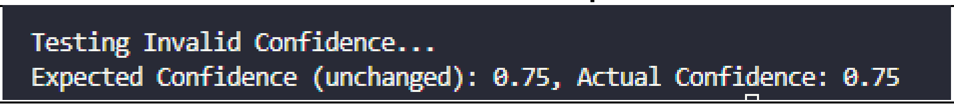
Task 1 – Horse Class

Test Code:



```java
public class horsetest {
    Run | Debug
    public static void main(String[] args) {
        Horse testHorse = new Horse(horseSymbol:'d', horseName:"Test Horse", horseConfidence:0.75);
        System.out.println(x:"\nTesting Invalid Confidence...");
        testHorse.setConfidence(newConfidence:1.5);
        System.out.println("Expected Confidence (unchanged): 0.85, Actual Confidence: " + testHorse.getConfidence());
    }
```

Output:



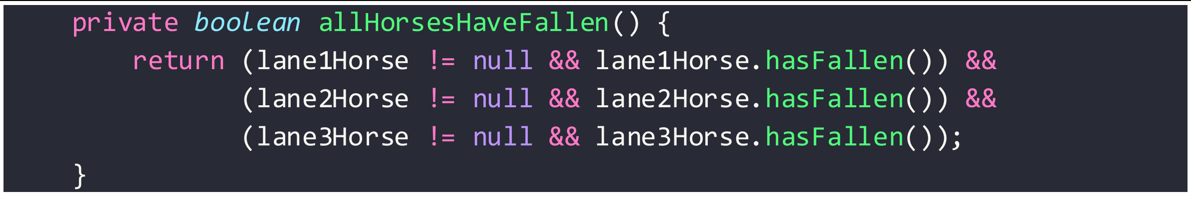
```
Testing Invalid Confidence...
Expected Confidence (unchanged): 0.75, Actual Confidence: 0.75
```

Explanation:

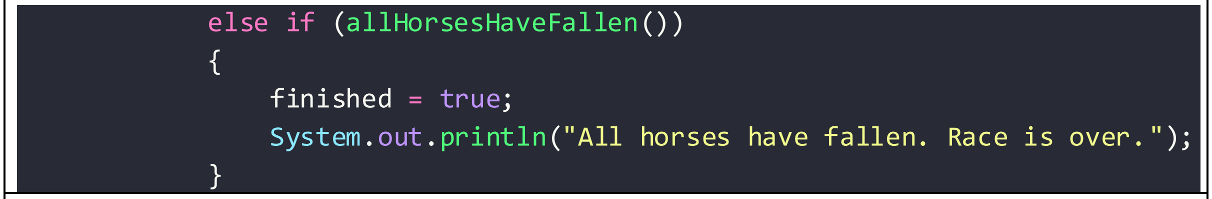
Here I am testing the setConfidence() method again but to see if it will accept and invalid confidence rating which is above the maximum threshold which is 1. I put 1.5 as the testing code and when I returned the getconfidence() method it returned the original horse confidence which was set as 0.75 meaning the test was successful.

Task 1 – Race Class

Test Code:

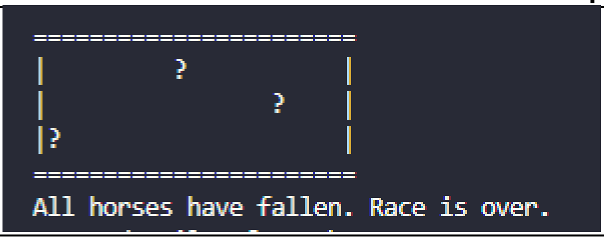


```java
    private boolean allHorsesHaveFallen() {
        return (lane1Horse != null && lane1Horse.hasFallen()) &&
               (lane2Horse != null && lane2Horse.hasFallen()) &&
               (lane3Horse != null && lane3Horse.hasFallen());
    }
```



```java
            else if (allHorsesHaveFallen())
            {
                finished = true;
                System.out.println("All horses have fallen. Race is over.");
            }
```

Output:



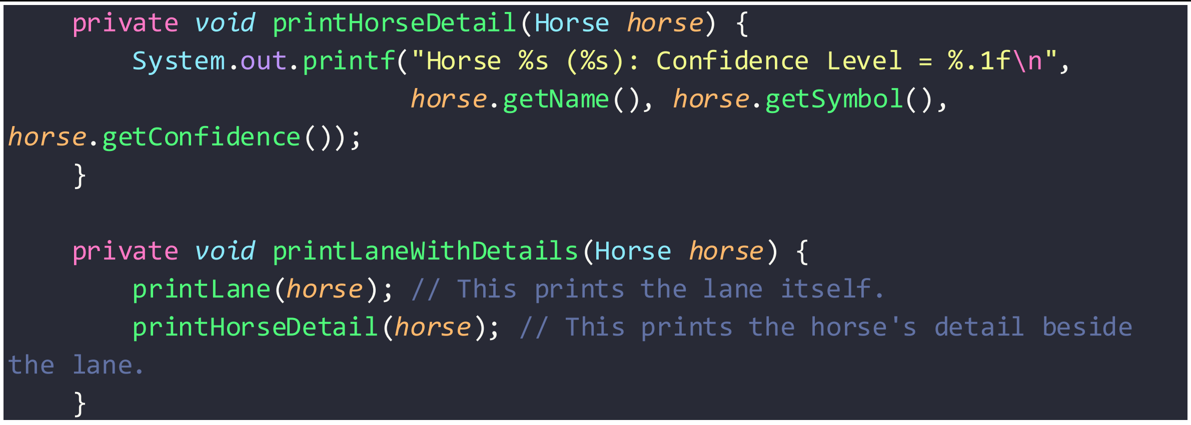
```
=======================
|        ?            |
|              ?      |
|?                    |
=======================
All horses have fallen. Race is over.
```

Explanation:

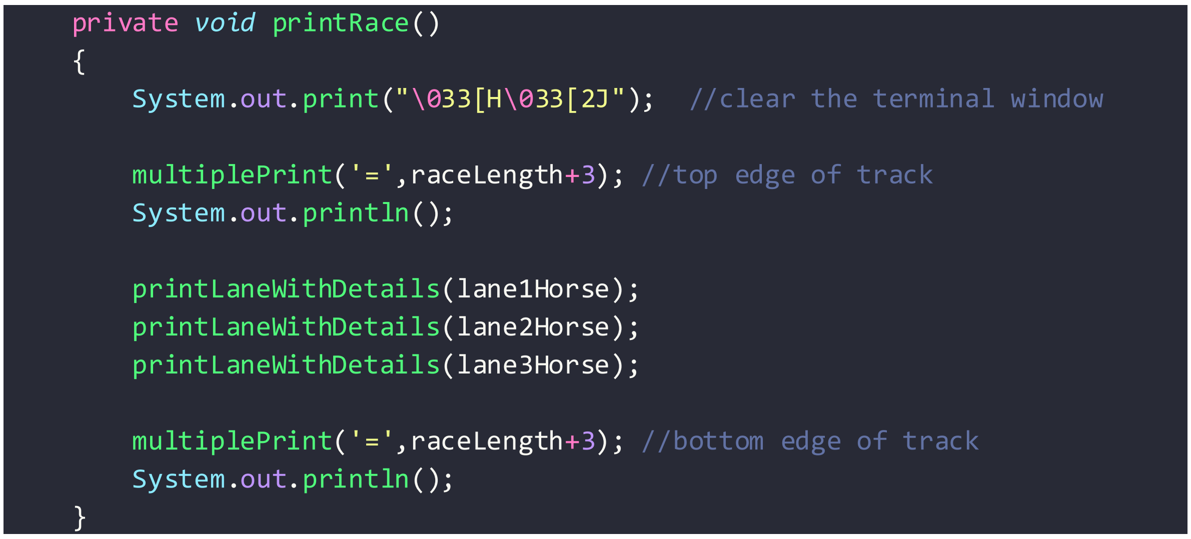
I have had one problem with my code and that was when testing the code and all the horses have fallen the code will still carry on an infinite loop as none of the horses have reached the finish line. I have coded a allHorsesHaveFallen() method which returns all the horses as fallen And I implanted this in the startrace() method as an else if statement checking if all the horses have fallen by running the allHorsesHaveFallen() method. If it is true then finished will equal true which will stop running the race and it will print the statement “All horses have fallen. Race is over.”.

Task 1 – Race Class

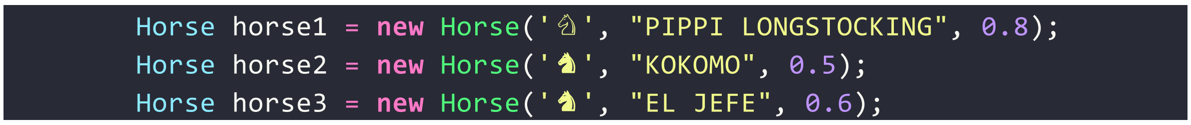
Test Code:



```
    private void printHorseDetail(Horse horse) {
        System.out.printf("Horse %s (%s): Confidence Level = %.1f\n",
                          horse.getName(), horse.getSymbol(),
horse.getConfidence());
    }

    private void printLaneWithDetails(Horse horse) {
        printLane(horse); // This prints the lane itself.
        printHorseDetail(horse); // This prints the horse's detail beside
the lane.
    }
```

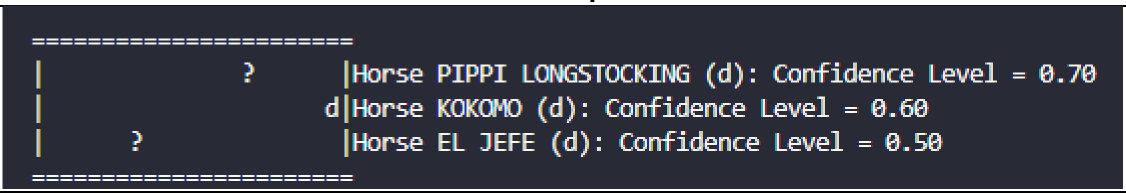


```
    private void printRace()
    {
        System.out.print("\033[H\033[2J");  //clear the terminal window

        multiplePrint('=',raceLength+3); //top edge of track
        System.out.println();

        printLaneWithDetails(lane1Horse);
        printLaneWithDetails(lane2Horse);
        printLaneWithDetails(lane3Horse);

        multiplePrint('=',raceLength+3); //bottom edge of track
        System.out.println();
    }
```



```
        Horse horse1 = new Horse('♘', "PIPPI LONGSTOCKING", 0.8);
        Horse horse2 = new Horse('♞', "KOKOMO", 0.5);
        Horse horse3 = new Horse('♞', "EL JEFE", 0.6);
```

Output:



```
=======================
|              ?      |Horse PIPPI LONGSTOCKING (d): Confidence Level = 0.70
|                    d|Horse KOKOMO (d): Confidence Level = 0.60
|      ?              |Horse EL JEFE (d): Confidence Level = 0.50
=======================
```

Explanation:

Here I have added the information of each horse including their name, symbol and their confidence level on the right side of the race so when the horse falls or wins the user can see the horses confidence level increase or decrease. I have created a printHorseDetail() method which prints the name, symbol and confidence of each horse. I have also created a printLaneWithDetails() method which displays the printHorseDetail method besides the printLane() method. I then changed the printRace() method by calling the printLaneWithDetails() for each horse. And now the horse details can be seen on the side of the race.

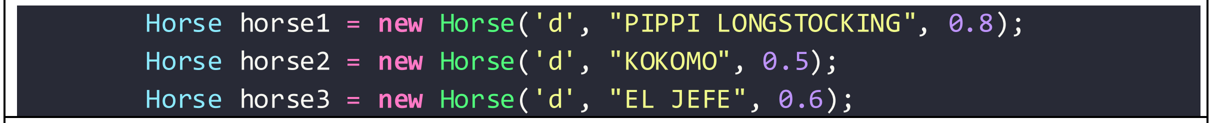
Task 1 – Race Class

Test Code:

```java
    private void increaseConfidence(Horse horse) {
        double newConfidence = horse.getConfidence() + 0.1;
        if (newConfidence > 1.0) {
            newConfidence = 1.0; // Cap the confidence at 1.0
        }
        horse.setConfidence(newConfidence); // Update the horse's confidence
    }
```

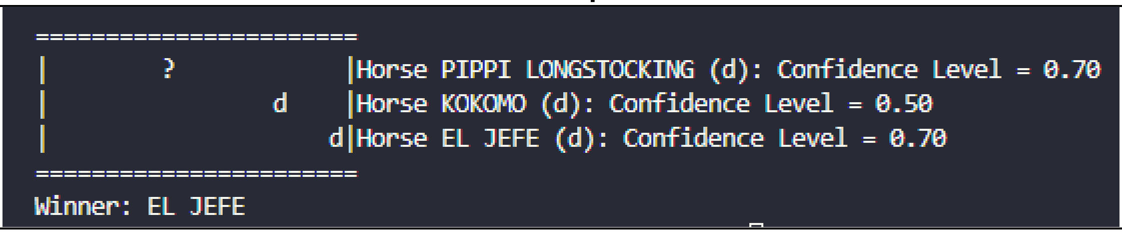
```java
            if ( raceWonBy(lane1Horse))
            {
                increaseConfidence(lane1Horse);
                printRace();
                finished = true;
            }
            else if (raceWonBy(lane2Horse))
            {
                increaseConfidence(lane2Horse);
                printRace();
                finished = true;
            }
            else if (raceWonBy(lane3Horse))
            {
                increaseConfidence(lane3Horse);
                printRace();
                finished = true;
            }
            else if (allHorsesHaveFallen())
            {
                finished = true;
                System.out.println("All horses have fallen. Race is over.");
            }
```

```java
    private boolean raceWonBy(Horse theHorse)
    {
        if (theHorse.getDistanceTravelled() == raceLength)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
```



```java
        Horse horse1 = new Horse('d', "PIPPI LONGSTOCKING", 0.8);
        Horse horse2 = new Horse('d', "KOKOMO", 0.5);
        Horse horse3 = new Horse('d', "EL JEFE", 0.6);
```

Output:



```
=======================
|        ?            |Horse PIPPI LONGSTOCKING (d): Confidence Level = 0.70
|                d    |Horse KOKOMO (d): Confidence Level = 0.50
|                    d|Horse EL JEFE (d): Confidence Level = 0.70
=======================
Winner: EL JEFE
```

Explanation:

Here I have added a increaseConfidence() method which increases the horses confidence by 0.1 and also makes sures that the horses confidence is capped at 1 and does not go above 1. The method also sets the newconfidence using the setconfidence() method. I have also changed the if statement by adding increaseConfidence() under each horse so if one of the horses win it increases their confidence as this was one of the criteria’s to add.

Task 1 – Race Class

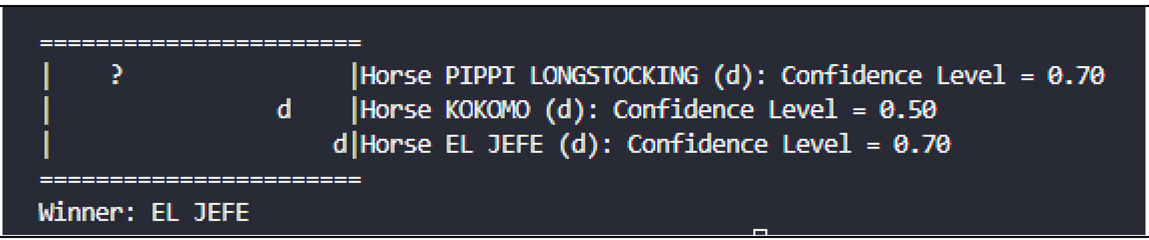
Test Code:

```java
            //if any of the three horses has won the race is finished
            if ( raceWonBy(lane1Horse))
            {
                increaseConfidence(lane1Horse);
                printRace();
                System.out.println("Winner: " + lane1Horse.getName());
                finished = true;
            }
            else if (raceWonBy(lane2Horse))
            {
                increaseConfidence(lane2Horse);
                printRace();
                System.out.println("Winner: " + lane2Horse.getName());
                finished = true;
            }
            else if (raceWonBy(lane3Horse))
            {
                increaseConfidence(lane3Horse);
                printRace();
                System.out.println("Winner: " + lane3Horse.getName());
                finished = true;
            }
```

```java
    private boolean raceWonBy(Horse theHorse)
    {
        if (theHorse.getDistanceTravelled() == raceLength)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
```

```java
public class main {
    public static void main(String[] args) {
        // Create a new Race with a length of 100 meters.
        Race race = new Race(30);

        // Create Horse objects
        Horse horse1 = new Horse('d', "PIPPI LONGSTOCKING", 0.8);
        Horse horse2 = new Horse('d', "KOKOMO", 0.5);
        Horse horse3 = new Horse('d', "EL JEFE", 0.6);
```

Output:



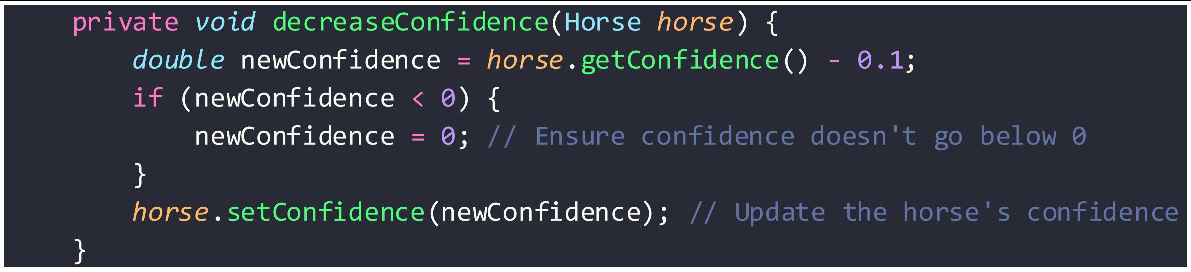
```
=======================
|    ?               |Horse PIPPI LONGSTOCKING (d): Confidence Level = 0.70
|               d    |Horse KOKOMO (d): Confidence Level = 0.50
|                   d|Horse EL JEFE (d): Confidence Level = 0.70
=======================
Winner: EL JEFE
```

Explanation:

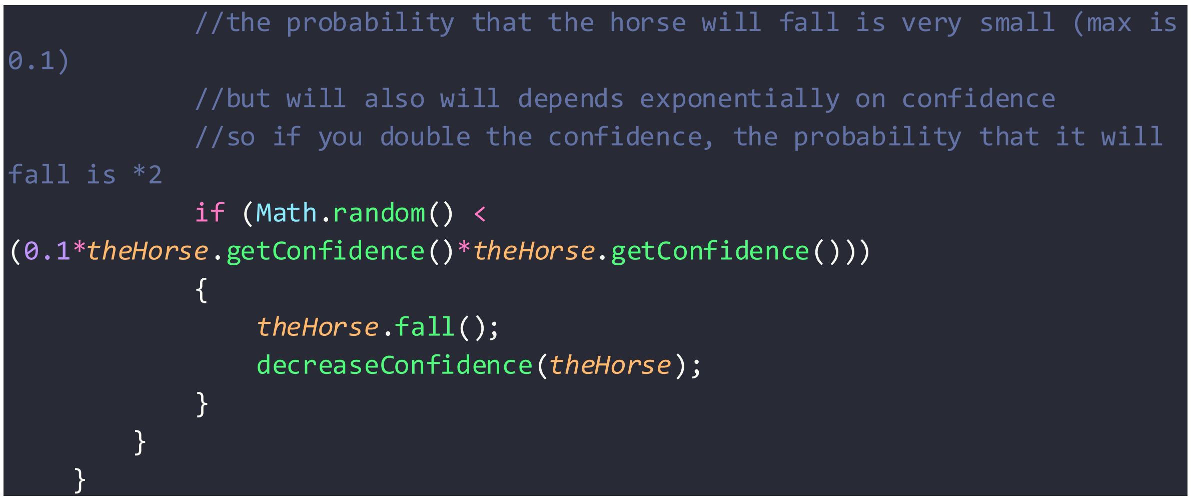
Here I have updated the if statement in the startrace() method so it prints the winners name as well as updating the printrace() method by calling it again so it shows the newly updated increase confidence. I have also added a print statetmnet so when the horse win it dispalys "Winner:" and the horses name that has won.

Task 1 – Race Class

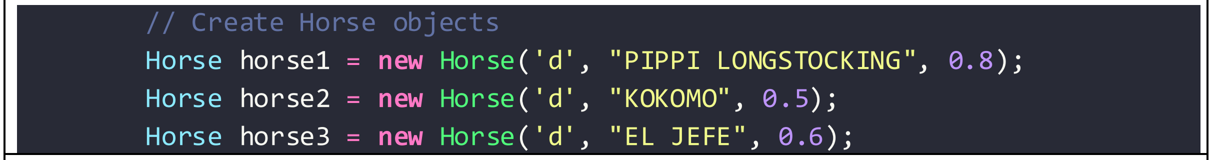
Test Code:



```
    private void decreaseConfidence(Horse horse) {
        double newConfidence = horse.getConfidence() - 0.1;
        if (newConfidence < 0) {
            newConfidence = 0; // Ensure confidence doesn't go below 0
        }
        horse.setConfidence(newConfidence); // Update the horse's confidence
    }
```

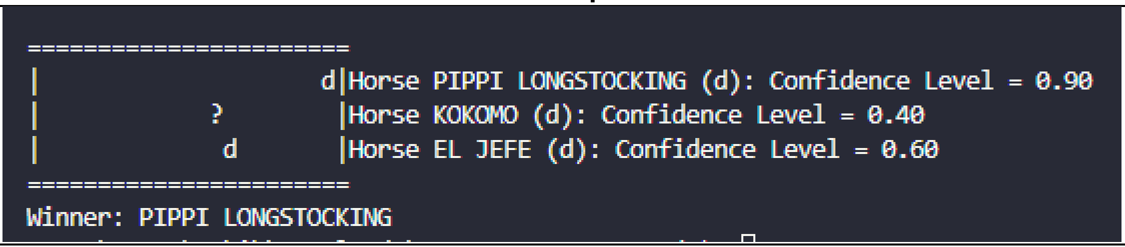


```
            //the probability that the horse will fall is very small (max is
0.1)
            //but will also will depends exponentially on confidence
            //so if you double the confidence, the probability that it will
fall is *2
            if (Math.random() <
(0.1*theHorse.getConfidence()*theHorse.getConfidence()))
            {
                theHorse.fall();
                decreaseConfidence(theHorse);
            }
        }
    }
```



```
        // Create Horse objects
        Horse horse1 = new Horse('d', "PIPPI LONGSTOCKING", 0.8);
        Horse horse2 = new Horse('d', "KOKOMO", 0.5);
        Horse horse3 = new Horse('d', "EL JEFE", 0.6);
```

Output:



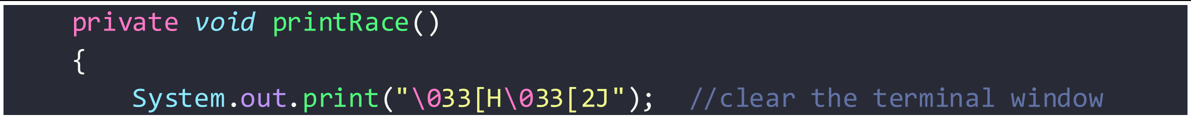
```
=======================
|                    d|Horse PIPPI LONGSTOCKING (d): Confidence Level = 0.90
|            ?        |Horse KOKOMO (d): Confidence Level = 0.40
|             d       |Horse EL JEFE (d): Confidence Level = 0.60
=======================
Winner: PIPPI LONGSTOCKING
```

Explanation:

Here I have added a decreaseConfidence() method which decreases the horses confidence by 0.1 and also makes sures that the horses confidence is capped at 0 and does not go below 0. The method also sets the newconfidence using the setconfidence() method. I have added this method in the moveHorse() method under when the horse.fall() method is called so when the horse falls the horse confidence decreases by 0.1. This was one of the criteria's to add when the horse falls the horse confidence decreases.

Task 1 – Race Class

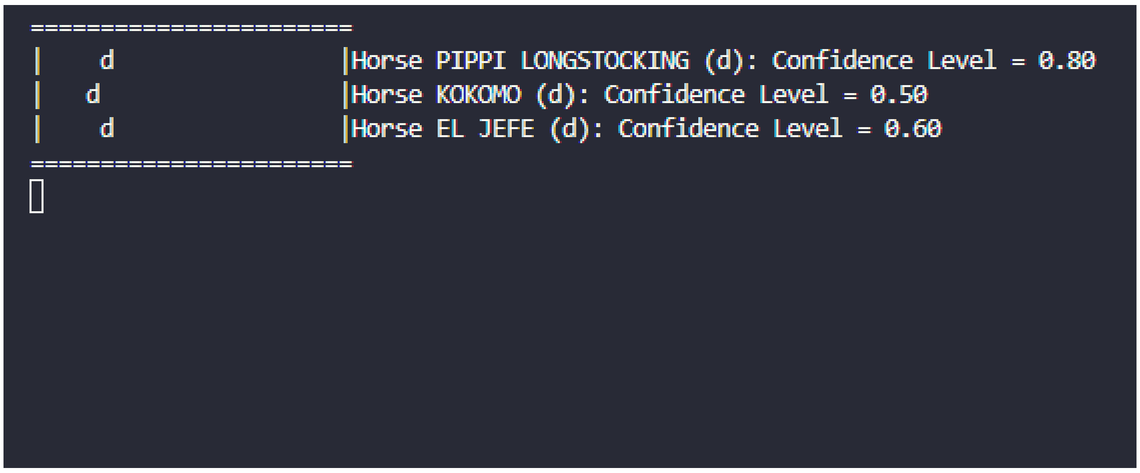
Test Code:



```
    private void printRace()
    {
        System.out.print("\033[H\033[2J");   //clear the terminal window
```

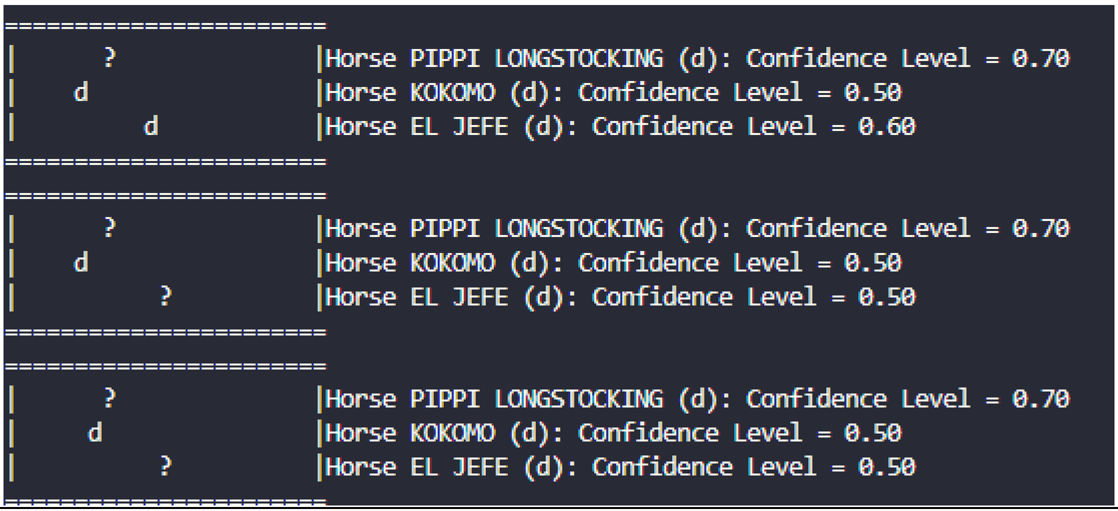
Output:

After:



```
=======================
|    d                |Horse PIPPI LONGSTOCKING (d): Confidence Level = 0.80
|   d                 |Horse KOKOMO (d): Confidence Level = 0.50
|    d                |Horse EL JEFE (d): Confidence Level = 0.60
=======================
```

Before:



```
=======================
|      ?              |Horse PIPPI LONGSTOCKING (d): Confidence Level = 0.70
|    d                |Horse KOKOMO (d): Confidence Level = 0.50
|         d           |Horse EL JEFE (d): Confidence Level = 0.60
=======================
=======================
|      ?              |Horse PIPPI LONGSTOCKING (d): Confidence Level = 0.70
|    d                |Horse KOKOMO (d): Confidence Level = 0.50
|          ?          |Horse EL JEFE (d): Confidence Level = 0.50
=======================
=======================
|      ?              |Horse PIPPI LONGSTOCKING (d): Confidence Level = 0.70
|     d               |Horse KOKOMO (d): Confidence Level = 0.50
|          ?          |Horse EL JEFE (d): Confidence Level = 0.50
=======================
```

Explanation:

Here I fixed the clear terminal window function as before it didn't clear the terminal window properly and the horse race kept repeating and after fixing the terminal window now it shows the most recent frame so it looks more clean and more like an animation compared to before.