```java
//Fields of class Horse
private String name;
private char symbol;
private int distance;
boolean fallen;
private double confidence;


//Constructor of class Horse
/**
 * Constructor for objects of class Horse
 */
public Horse(char horseSymbol, String horseName, double horseConfidence)
{
    symbol = horseSymbol;
    name = horseName;
    confidence = horseConfidence;
    fallen = false;
}
```

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

```java
//Other methods of class Horse
public void fall()
{
    fallen = true;
    return;
}

public double getConfidence()
{
    return this.confidence;
}

public int getDistanceTravelled()
{
    return this.distance;
}

public String getName()
{
    return this.name;
}

public char getSymbol()
{
    return this.symbol;
}

public void goBackToStart()
{
    distance = 0;
    return;
}
```

```java
public boolean hasFallen()
{
    return this.fallen;
}

public void moveForward()
{
    distance ++;
    return;
}

public void setConfidence(double newConfidence)
{
    if (newConfidence < 0) {
        newConfidence = 0;
    } else if (newConfidence > 1) {
        newConfidence = 1;
    } else {
        confidence = newConfidence;
    }
    return;
}

public void setSymbol(char newSymbol)
{
    symbol = newSymbol;
    return;
}
```

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:

```java
class HorseRaceSimulator {
    Run | Debug
    public static void main(String[] args) {
        Horse horse1 = new Horse(horseSymbol:'🏇', horseName:"GLITTERHOOF", horseConfidence:0.78);
        Horse horse2 = new Horse(horseSymbol:'🐴', horseName:"RAINBOW DASH", horseConfidence:3);
        Horse horse3 = new Horse(horseSymbol:'馬', horseName:"HORSE 3", -0.65);

        horse1.setConfidence(newConfidence:0.78);
        horse2.setConfidence(newConfidence:3);
        horse3.setConfidence(-0.65);

        printHorseInformation(horse1);
        printHorseInformation(horse2);
        printHorseInformation(horse3);
    }

    public static void printHorseInformation (Horse horse) {
        System.out.println("HORSE NAME: " + horse.getName());
        System.out.println("HORSE SYMBOL: " + horse.getSymbol());
        System.out.println("HORSE CONFIDENCE: " + horse.getConfidence());
    }
}
```

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.

```
HORSE NAME: GLITTERHOOF
HORSE SYMBOL: ?
HORSE CONFIDENCE: 0.78
HORSE NAME: RAINBOW DASH
HORSE SYMBOL: ?
HORSE CONFIDENCE: 3.0
HORSE NAME: HORSE 3
HORSE SYMBOL: ?
HORSE CONFIDENCE: -0.65
```

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.

```
public void setConfidence(double newConfidence)
{
    if (newConfidence < 0) {
        newConfidence = 0;
    } else if (newConfidence > 1) {
        newConfidence = 1;
    } else {
        this.confidence = newConfidence;
    }
    return;
}
```

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

```
public void setConfidence(double newConfidence)
{
    if (newConfidence < 0) {
        newConfidence = 0;
    } else if (newConfidence > 1) {
        newConfidence = 1;
    }
    this.confidence = newConfidence;
    return;
}
```

Race Class

```
//if any of the three horses has won the race is finished
if ( raceWonBy(lane1Horse) || raceWonBy(lane2Horse) || raceWonBy(lane3Horse) )
{
    finished = true;
}
```

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

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:

```java
private void moveHorse(Horse theHorse)
{
    //if the horse has fallen it cannot move,
    //so only run if it has not fallen

    if  (!theHorse.hasFallen())
    {
        //the probability that the horse will move forward depends on the confidence;
        if (Math.random() < theHorse.getConfidence())
        {
           theHorse.moveForward();
        }

        //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();
            theHorse.setSymbol(newSymbol:'X');
        }
    }
```

However, this did not work.

```java
//if the horse has fallen then print dead
//else print the horse's symbol
if(theHorse.hasFallen())
{
    System.out.print(c:'\u2322');
}
else
{
    System.out.print(theHorse.getSymbol());
}
```

```java
//if the horse has fallen then print dead
//else print the horse's symbol
if(theHorse.hasFallen())
{
    System.out.print(c:'X');
}
else
{
    System.out.print(theHorse.getSymbol());
}
```

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

```
==============
|          ?  |
|          X  |
|?            |
==============


==============
|           ? |
|          X  |
|?            |
==============


==============
|            ?|
|          X  |
|?            |
==============
And the winner is GLITTERHOOF
```

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.

```
==============
|         G  |
|           R|
| B          |
==============
And the winner is GLITTERHOOF
```

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.

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

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

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

```
    System.out.print(" " + theHorse.getName() + " (Current Confidence " + theHorse.getConfidence() + ")");
  }
```

```
==============
|       G    | GLITTERHOOF (Current Confidence 0.78)
|         R  | RAINBOW DASH (Current Confidence 1.0)
|B           | BOJACK (Current Confidence 0.1)
==============
```

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.

```
if (lane1Horse.hasFallen() && lane2Horse.hasFallen() && lane3Horse.hasFallen()) {
    System.out.println(x:"All horses have fallen. No winner");
    finished = true;
}
```

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

```
=============
|X           | GLITTERHOOF (Current Confidence 0.78)
|      X     | RAINBOW DASH (Current Confidence 0.3)
|  X         | BOJACK (Current Confidence 0.65)
=============
All horses have fallen. No winner
```

It led to this result.

```
public void setConfidence(double newConfidence)
{
    Scanner s = new Scanner(System.in);
    while (newConfidence < 0 || newConfidence > 1) {
        System.out.println("Invalid number for " + this.getName() + ". Must be between 0 and 1");
        newConfidence = Double.parseDouble(s.nextLine());
    }
    this.confidence = newConfidence;
    return;
}
```

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.