Name: Lim Jie Ying
Student ID: 230417942

Report Part I - Horse Class

In the Horse class, encapsulation is used so data is not accidentally manipulated. For example, the field name cannot be accessed outside the class. If object.name is called in another class such as Race, the compiler will raise an error. This is achieved by setting all fields to private, which means that these fields cannot be accessed outside of the class. To access these fields to retrieve and modify data in these fields from other classes, public methods must be included within the Horse class. These methods will access the fields directly.

Encapsulation is important because it prevents other classes from accidentally changing values in the horse class. This will be useful when developing larger code (Such as in Part II), as if a value is accidentally changed, it will be time consuming to debug. However, if fields are set to private, the compiler will raise an error, identifying it and the program will not run (Illustrated in the second test below). This will make it easier to make changes to the code.

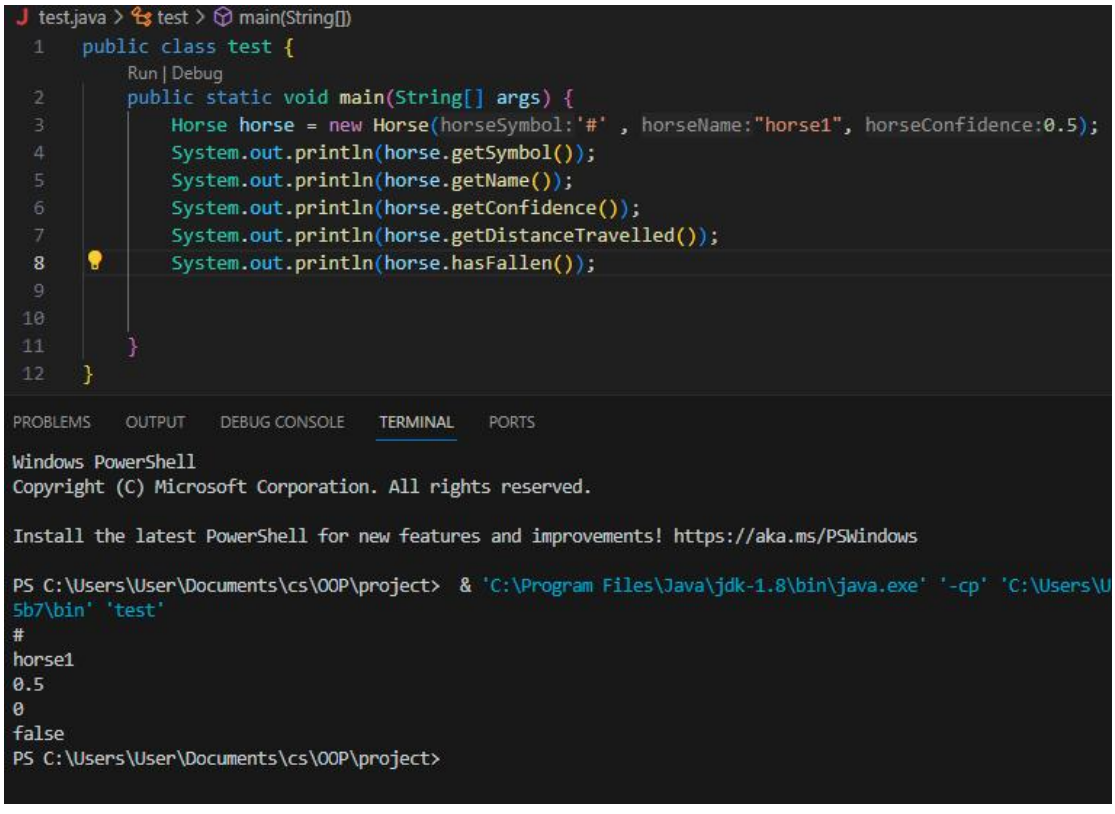
The methods that modifies data (mutator methods) are fall, goBackToStart, moveForward, setConfidence, setSymbol

The methods for retrieving data(accessor methods) getConfidence, getDistanceTravelled, getName, getSymbol, hasFallen

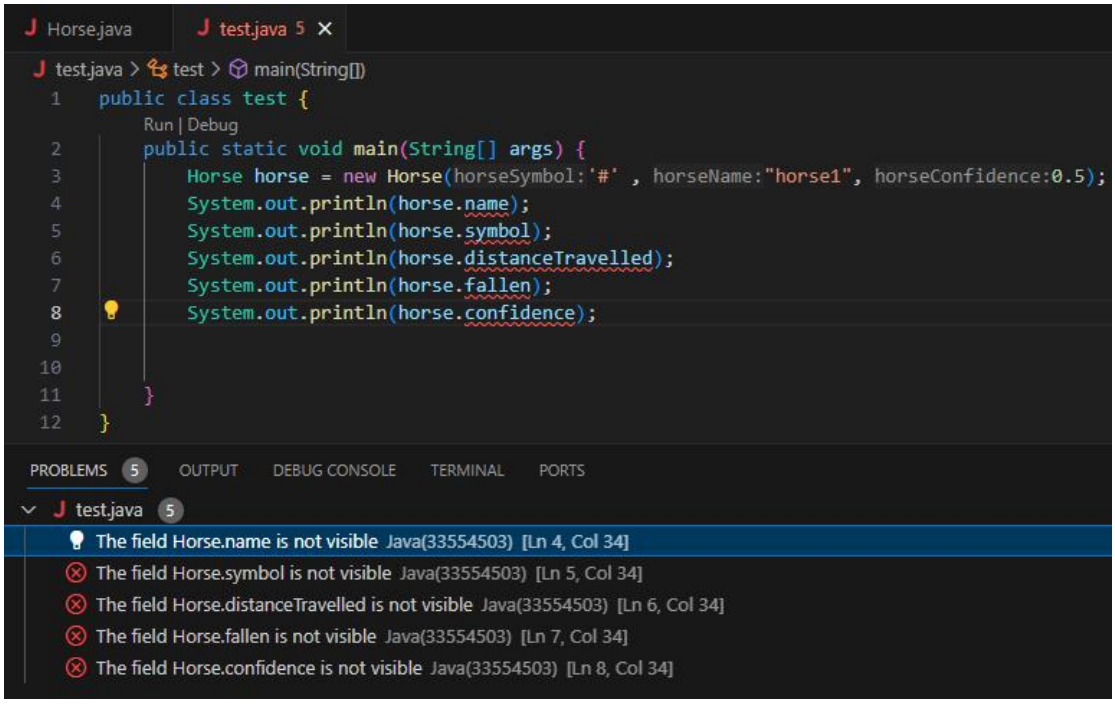
The accessor methods returns the values stored in each field and the mutator methods will modify/ change the values which are currently stored in the fields. To illustrate this, I have provided a series of test codes.

Testing

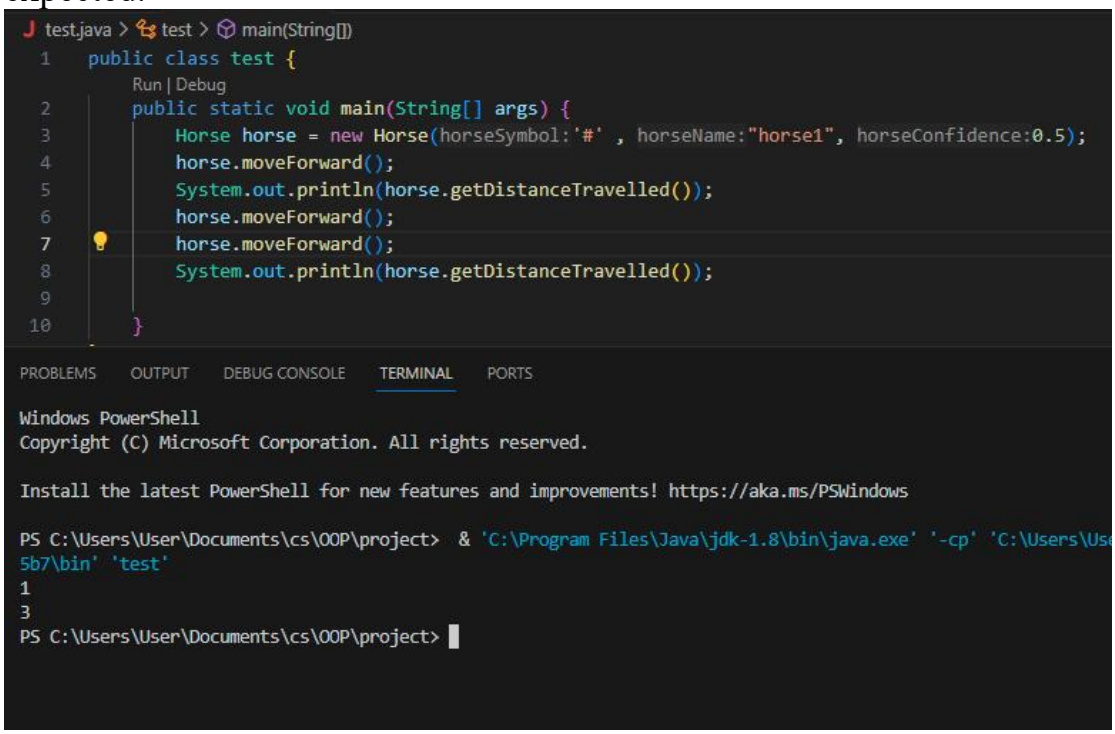
1) The following code tests the constructor and the 5 accessor methods getSymbol, getName, getConfidence getDistanceTravelled, hasFallen . A new object called horse is created and its given 3 arguments, a character(which represents a horse symbol), a String (which represents a name) and a double(which represents the confidence). Then in the test code, all the fields are printed out. This shows that the constructor is working as expected as it takes the arguments and assigns them to its respective fields. Although an argument for distanceTravelled and fallen is not passed which the object is created, by default it should be set to 0 and false respectively. Each of the accessor methods are working normally, as illustrated by the test code, when each is called, it returns the data stored in its respective fields. The test code then prints them out.



2) The following test code checks if each of the fields are set to private. This is shown by trying to access each field directly. As expected, the compiler raises an error telling me that the fields are not visible, which mean that these fields are not accessible from the method test, and only accessible from the method Horse. However, the previous test code shows that the data can still be retrieved from the accessor methods. From this, I can conclude that encapsulation has been successfully implemented. The code is working as normal.



3) The following code tests the methods moveForward. When moveForward is called, it increments by 1. The result is then printed out when moveForward is called once then another 2 times (which totals to it being called 3 times.). Get DistanceTravelled needs to be called when printing out the total distance travelled as the instance variable distanceTravelled is set to private. The test code prints out 1 then 3 as expected.



4) Next, this is the test for the goBackToStart method. The horse is moved forward twice, then the distance is printed out (which is 2). Next, goBackToStart is called and this should reset the distance back to 0. As illustrated by the test code, it prints out 2 then 0, which indicates the distance is reset. The method is working properly.

```java
public class test {
    Run | Debug
    public static void main(String[] args) {
        Horse horse = new Horse(horseSymbol:'#' , horseName:"horse1", horseConfidence:0.5);
        horse.moveForward();
        horse.moveForward();
        System.out.println(horse.getDistanceTravelled());
        horse.goBackToStart();
        System.out.println(horse.getDistanceTravelled());


    }
}

```

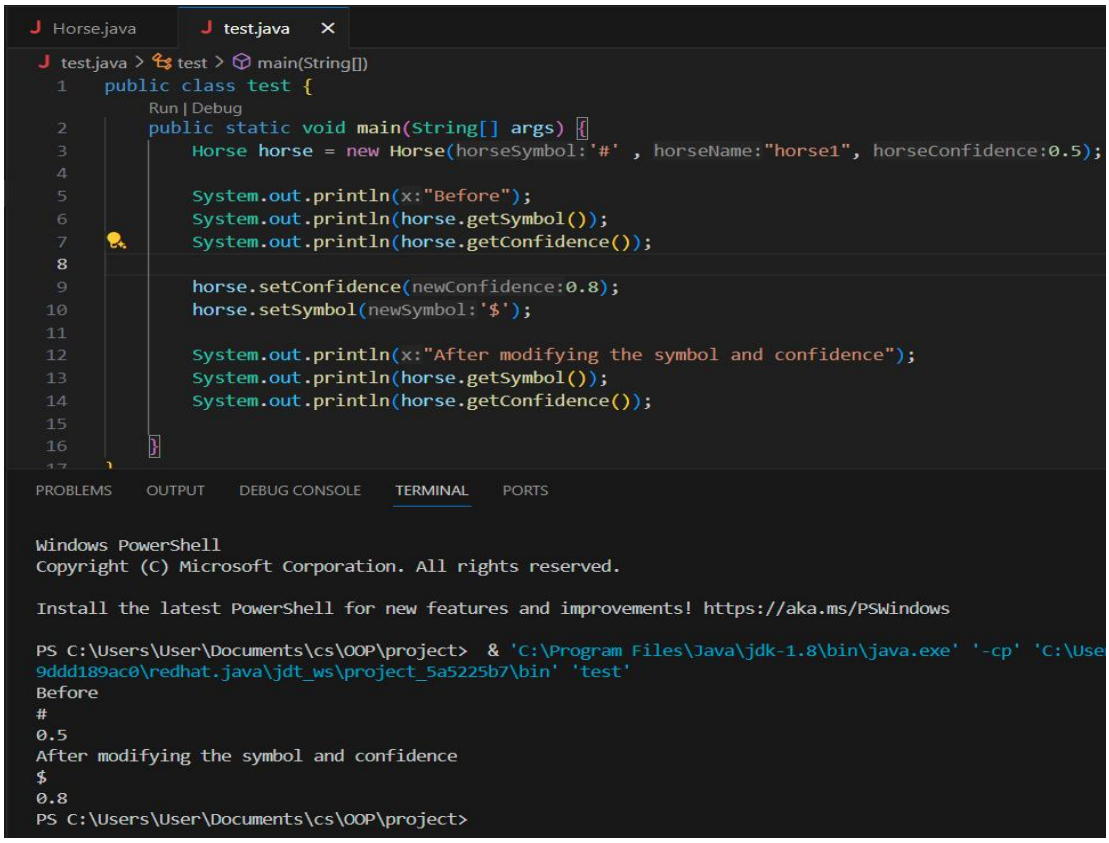
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\User\Documents\cs\OOP\project>  & 'C:\Program Files\Java\jdk-1.8\bin\java.exe' '-cp' 'C:\User
9ddd189ac0\redhat.java\jdt_ws\project_5a5225b7\bin' 'test'
2
0
PS C:\Users\User\Documents\cs\OOP\project>
```

5) The following test code tests the method fall. By default, the field fallen should be set to false when the object is first created. This is printed out first. Next, the method fall is called and this should set the field fallen to be true. The code shows that the method is working as normal when the result is printed out. This also shows that the if statement in the hasFallen method is working properly.

```java
public class test {
    Run | Debug
    public static void main(String[] args) {
        Horse horse = new Horse(horseSymbol:'#' , horseName:"horse1", horseConfidence:0.5);

        System.out.println(horse.hasFallen());
        horse.fall();
        System.out.println(horse.hasFallen());


    }
}

```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\User\Documents\cs\OOP\project>  & 'C:\Program Files\Java\jdk-1.8\bin\java.exe' '-cp' 'C:\User
9ddd189ac0\redhat.java\jdt_ws\project_5a5225b7\bin' 'test'
false
true
PS C:\Users\User\Documents\cs\OOP\project>
```

6) Lastly, testing the methods setSymbol and setConfidence. When the object is created, we passed the value for the symbol, name and confidence to the constructor. The 2 methods setSymbol and setConfidence allows to modify the values stored in the fields after the object is created as the fields are set to private. In the code, we modified the symbol from '#' to '$' and the confidence from 0.5 to 0.8.
As illustrated by the test code and the execution, the horse object is created with a # symbol and a confidence of 0.5 and these values are printed out initially. Then, setConfidence and setSymbol is called and values 0.8 and $ is passed to it. Lastly, yje updated values are printed out; the old values are lost. This shows that these 2 methods are working as normal.



Race Class

Below is a todo list of all the modifications I made for the Race class

TODO
1. Able to have an arbitrary number of tracks.
2. Able to have empty lanes.
3. Lanes assigned cannot be more than number_of_lanes.
4. PrintLane uses a for loop to print out each horses name
5. Prints out the winning horse's name, Prints out 2 lines of spacing before printing the winner
6. Increase confidence if won
7. If confidence is 1, do not increase further
8. When the horse falls, an X should be printed instead of a ?
9. When all horses fall, there is no winner, program should terminate
10. Names and current confidence should be printed out after the race lane.
11. decrease confidence after falling
12. If confidence is 0, do not decrease further
13. Terminal window does not clear properly when printing out the race.
14. Distance must be >= 0
15. When starting, confidence of each horse must be between 0 to 1
16. Accepts meters or yards.

1) Instead of having a variable for each horse object, I created an arrayList called horses to hold the objects. The lane number is specified by the index of the arrayList. By doing this, I need to change the constructor to accept another argument, which is the total number of lanes in the race. In the constructor, I used a for loop to fill up the arrayList with null, x times (where x is the number of lanes) to set the size of the arrays list. This is done because if a user accidentally assigns a horse to a lane number which is > x, an error could be raised. Throughout the entire code, the accessing for each horse object will be changed to horses.get(i) where i is the lane number. Next, the method add horse would also need to be changed. Since I used an arrayslist, I can use the set method to add a new horse object in a specific index.

```java
public class Race
{
    private int raceLength;
    private List<Horse> horses;

    private boolean laneError = false;

    /**
     * Constructor for objects of class Race
     * Initially there are no horses in the lanes
     *
     * @param distance the length of the racetrack (in metres/yards...)
     */
    public Race(int distance, int no_of_lanes)
    {
        // initialise instance variables
        this.raceLength = distance;
        horses = new ArrayList<Horse>();
        for(int i = 0; i < no_of_lanes; i++)
        {
            horses.add(e:null);
        }
    }

    /**
     * Adds a horse to the race in a given lane
     *
     * @param theHorse the horse to be added to the race
     * @param laneNumber the lane that the horse will be added to
     */
    public void addHorse(Horse theHorse, int laneNumber)
    {

        if (laneNumber <= horses.size() && laneNumber >=1)
        {
            horses.set(laneNumber-1, theHorse);
        }
        else{
            System.out.println("Cannot add horse to lane " + laneNumber + " because there is no such lane");
            laneError = true;
        }

    }
```

Testing

```java
test.java > test > main(String[])
1   public class test {
        Run | Debug
2       public static void main(String[] args) {
3           Race race = new Race(distance:10, no_of_lanes:4);
4           Horse horse1 = new Horse(horseSymbol:'+', horseName:"Frank", horseConfidence:0.2);
5           Horse horse2 = new Horse(horseSymbol:'*', horseName:"Bob", horseConfidence:0.4);
6           Horse horse3 = new Horse(horseSymbol:'&', horseName:"cOW", horseConfidence:0.5);
7           Horse horse4 = new Horse(horseSymbol:'!', horseName:"name", horseConfidence:0.7);
8
9           race.addHorse(horse1, laneNumber:1);
10          race.addHorse(horse2, laneNumber:2);
11          race.addHorse (horse3, laneNumber:3);
12          race.addHorse(horse4, laneNumber:4);
13
14          race.startRace();
15
16      }
17  }
18
```

```
PROBLEMS 12   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

============
|     +     | Frank (Current confidence 0.2)
|          *| Bob (Current confidence 0.4)
|    X      | cOW (Current confidence 0.5)
|       X   | name (Current confidence 0.7)
============

And the winner is Bob
PS C:\Users\User\Documents\cs\OOP\project V2>
```

2) To able to have an empty lane, changes to the method printLane is required. If the arraylist has a null in any of its positions, it prints out an empty lane. Else, it prints out a normal horse lane with the horse symbol. To print an empty lane, a character '|' is printed first. Then the method multiplePrint is called and it prints an empty space " " x amount of times, where x is the distance of the race. Lastly '|' is printed to indicate the end of the lane.

```
private void printLane(Horse theHorse)
{
    //TODO handles empty lanes
    if(theHorse == null)
    {
        System.out.print(s:"|");
        multiplePrint(aChar:' ', raceLength+1);
        System.out.print(s:"|");
    }
    else
    {
        //calculate how many spaces are needed before
        //and after the horse
        int spacesBefore = theHorse.getDistanceTravelled();
        int spacesAfter = raceLength - theHorse.getDistanceTravelled();

        //print a | for the beginning of the lane
        System.out.print(c:'|');

        //print the spaces before the horse
        multiplePrint(aChar:' ',spacesBefore);
```

Testing

Race.java | Horse.java | test.java

test.java > ...

```
1   public class test {
        Run | Debug
2       public static void main(String[] args) {
3           Race race = new Race(distance:10, no_of_lanes:4);
4           Horse horse1 = new Horse(horseSymbol:'+', horseName:"Frank", horseConfidence:0.2);
5           Horse horse2 = new Horse(horseSymbol:'*', horseName:"Bob", horseConfidence:0.4);
6           Horse horse4 = new Horse(horseSymbol:'!', horseName:"name", horseConfidence:0.7);
7
8           race.addHorse(horse1, laneNumber:1);
9           race.addHorse(horse2, laneNumber:2);
10          race.addHorse(horse4, laneNumber:4);
11
12          race.startRace();
13
14      }
15  }
16
```

PROBLEMS 13 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
=============
|          +| Frank (Current confidence 0.2)
|      X    | Bob (Current confidence 0.4)
|           |
|  X        | name (Current confidence 0.7)
=============

And the winner is Frank
PS C:\Users\User\Documents\cs\OOP\project V2>
```

In the above test code, I removed horse3, In the terminal, lane 3 is empty. The code is working as expected.

3) To prevent assigning horses to lanes which are more than the specified number of lanes, I have implemented some input validation. In the method addHorse, if the number is less than 1 or more than the size of the arrayList, then it moves to the else statement which sets the boolean variable laneError to true and prints out a warning. Then, in the method StartRace, an if statement tests if the laneError is true, it sets the boolean variable finished to true, which prevents the race from starting.

```
public void addHorse(Horse theHorse, int laneNumber)
{

    if (laneNumber <= horses.size() && laneNumber >=1)
    {
        horses.set(laneNumber-1, theHorse);
    }
    //TODO input validation for laneNumber
    else{
        System.out.println("Cannot add horse to lane " + laneNumber + " because there is no such lane");
        laneError = true;
    }

}
```

```
public void startRace()
{

    //declare a local variable to tell us when the race is finished
    boolean finished = false;

    //reset all the lanes (all horses not fallen and back to 0).
    for (int i = 0; i < horses.size(); i++) {
        if (horses.get(i) != null) {
            horses.get(i).goBackToStart();
        }
    }

    //TODO input validation for laneError
    if(laneError)
    {
        finished = true;
        System.out.println(x:"Cannot start the race due to lane error.");
    }
    //TODO Input validation for raceLength
```

Testing

In the test code, I created a new race with 3 lanes. Then I assigned horse1 to lane 3. In the terminal, the race does not start and warns me that lane 3 does not exist.

test.java > test > main(String[])

```java
public class test {
    Run | Debug
    public static void main(String[] args) {
        Race race = new Race(distance:10, no_of_lanes:2);
        Horse horse1 = new Horse(horseSymbol:'+', horseName:"Frank", horseConfidence:0.2);
        Horse horse2 = new Horse(horseSymbol:'*', horseName:"Bob", horseConfidence:0.8);

        race.addHorse(horse1, laneNumber:3);
        race.addHorse(horse2, laneNumber:2);

        race.startRace();

    }
}
```

PROBLEMS 13 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\User\Documents\cs\OOP\project V2>  & 'C:\Program Files\Java\jdk-1.8\bin\java.exe' '-cp' 'C:\Users
_bd936da5\bin' 'test'
Cannot add horse to lane 3 because there is no such lane
Cannot start the race due to lane error.
PS C:\Users\User\Documents\cs\OOP\project V2>
```

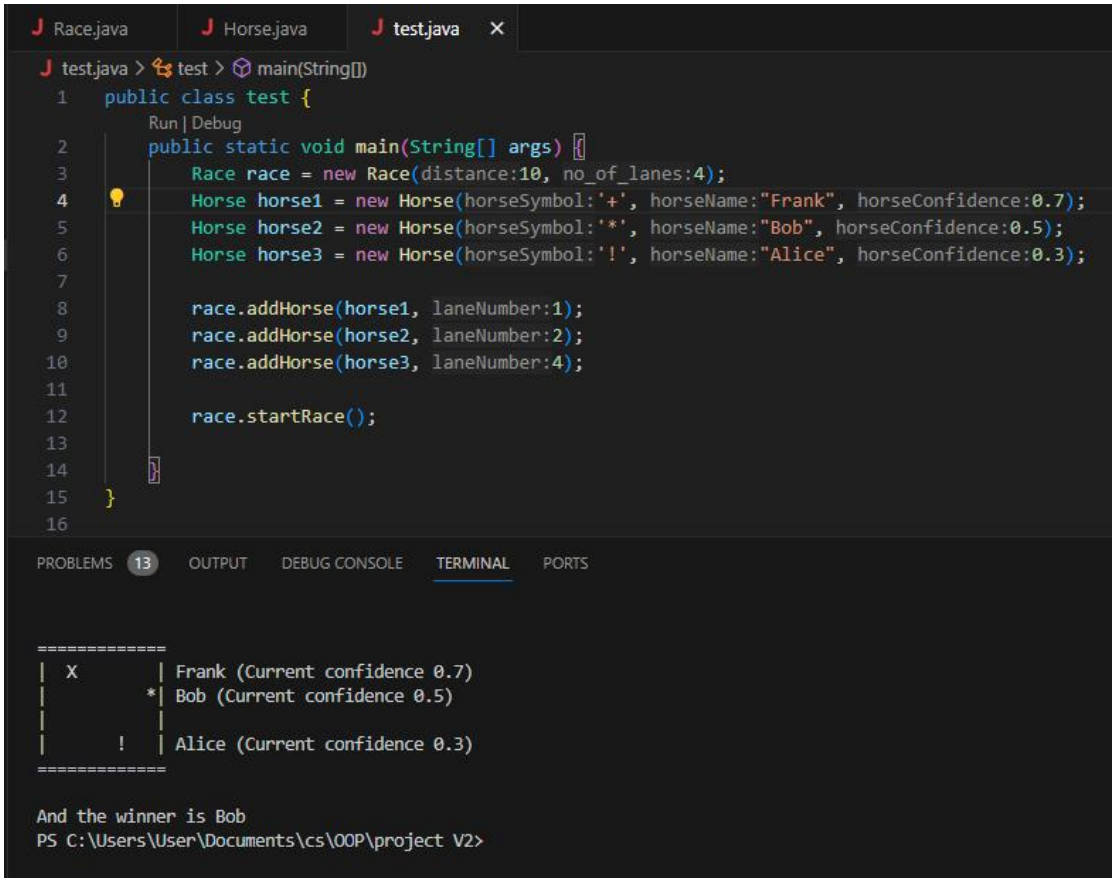
4) To accommodate for the ArraysList, a for loop is needed to print out the lanes for each of the horses in the method printRace.

Race.java | Horse.java | test.java

Race.java > Race > printRace()

```java
public class Race
    private void printRace()

        //TODO use a for loop to print each lane

        for(int i = 0; i < horses.size(); i++)
        {
            printLane(horses.get(i));
            System.out.println();
        }

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

Testing

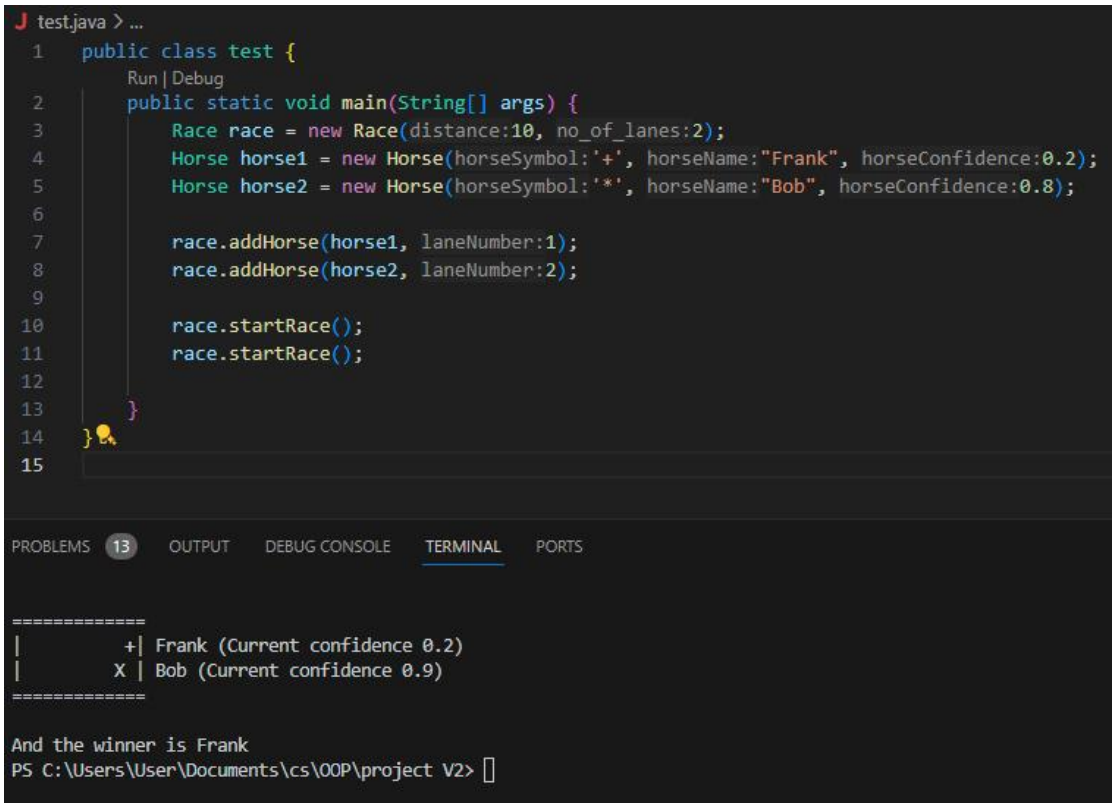


5,6,7) To print out the name of the winner and increase the confidence after winning, I further modified the method startRace. The code loops through the arrayList and the if statement detects if any horses has won. If true, it prints out the winner. Then, if the horses confidence level is below 1.0 (as the maximum confidence level is 1.0), then the confidence level is increased slightly by 0.1.

```java
//if any of the horses has won, the race is finished

//TODO print the winner
//TODO increase the confidence of the winner by 0.1. if 1, do not increase further
System.out.println();
for(int i = 0; i < horses.size(); i++)
{
    if(horses.get(i) != null && raceWonBy(horses.get(i)))
    {
        finished = true;

        System.out.println("And the winner is " + horses.get(i).getName());
        if(horses.get(i).getConfidence() < 1.0)
        {
            horses.get(i).setConfidence(horses.get(i).getConfidence() + 0.10);
        }

        break;
    }
}
```

Testing

As illustrated, the winner is printed at the bottom of the race at the end, startRace is called twice to test the increase of the confidence level. In the first race, horse2 (Bob) won so its confidence level increased from 0.8 to 0.9



8)If a horse has fallen, an ‘X’ should be printed out instead, to fix this, I just changed the code to print an ‘X’ instead of a ‘?’ in the method printLane

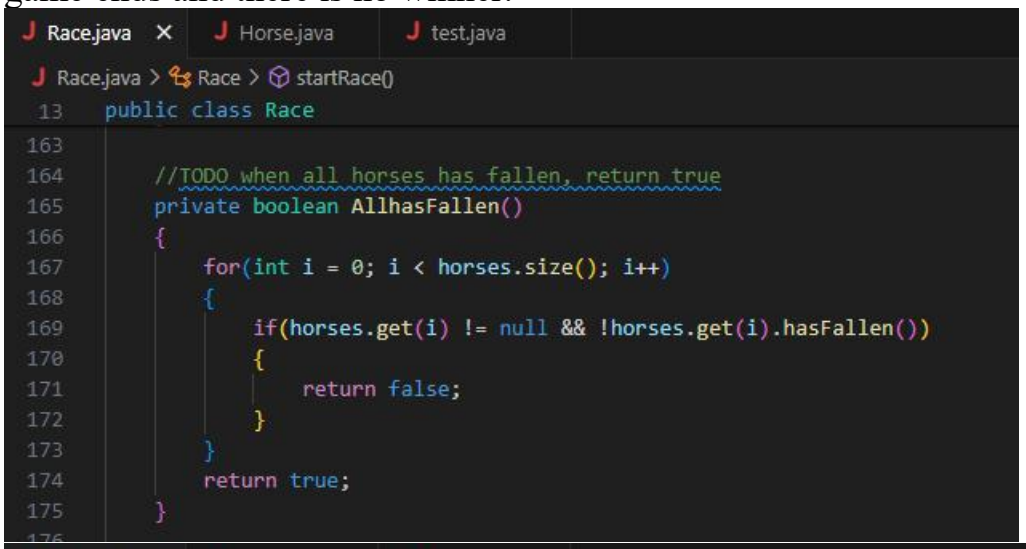
```
Race.java  Horse.java  test.java
Race.java > Race > printLane(Horse)
 13    public class Race
272        private void printLane(Horse theHorse)
294            //if the horse has fallen then print dead
295            //else print the horse's symbol
296            if(theHorse.hasFallen())
297            {
298                System.out.print(c:'X');
299            }
300            else
301            {
302                System.out.print(theHorse.getSymbol());
303            }
304
305            //print the spaces after the horse
306            multiplePrint(aChar:' ',spacesAfter);
```

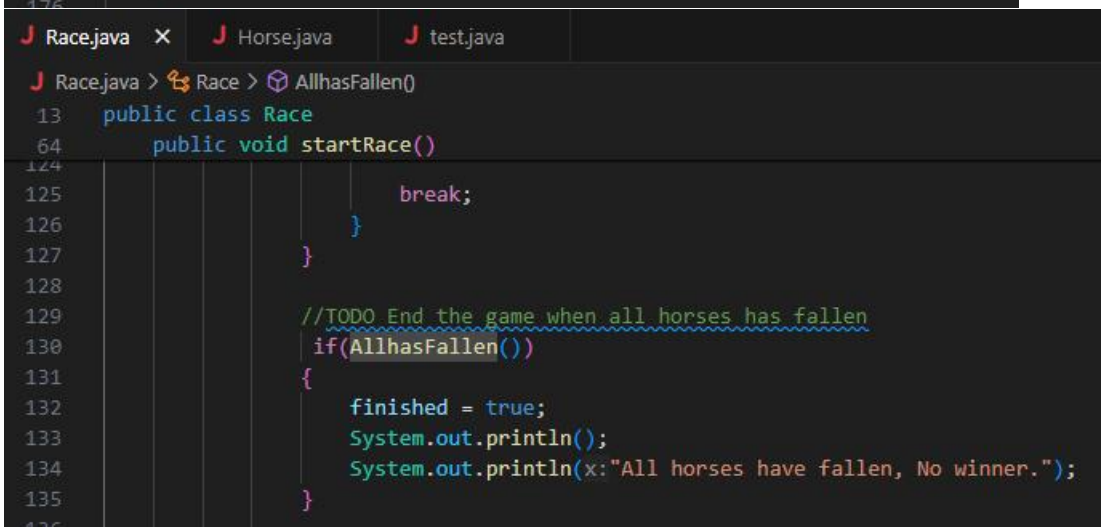
Testing

The test code below shows that the horse in lane 1 has fallen, so an X is printed out.

```
Race.java  Horse.java  test.java
test.java > test > main(String[])
 1   public class test {
         Run | Debug
 2       public static void main(String[] args) {
 3           Race race = new Race(distance:20, no_of_lanes:2);
 4           Horse horse1 = new Horse(horseSymbol:'+', horseName:"Frank", horseConfidence:0.9);
 5           Horse horse2 = new Horse(horseSymbol:'*', horseName:"Bob", horseConfidence:0.9);
 6
 7           race.addHorse(horse1, laneNumber:1);
 8           race.addHorse(horse2, laneNumber:2);
 9
10           race.startRace();
11       }
12   }
13

PROBLEMS 13  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

=====================
|      X            | Frank (Current confidence 0.9)
|                  *| Bob (Current confidence 0.9)
=====================

And the winner is Bob
```

9)Originally, when all horses fall, the program never terminates. To fix this, I created a method called AllhasFallen and it detects if all the horses has fallen, it returns true, if it detects there is a horse still standing, it returns false. This method is called in the method startRace. If its true, the game ends and there is no winner.





Testing

Race.java | Horse.java | test.java

test.java > test > main(String[])

```
1   public class test {
        Run | Debug
2       public static void main(String[] args) {
3           Race race = new Race(distance:30, no_of_lanes:2);
4           Horse horse1 = new Horse(horseSymbol:'+', horseName:"Frank", horseConfidence:0.9);
5           Horse horse2 = new Horse(horseSymbol:'*', horseName:"Bob", horseConfidence:0.9);
6
7           race.addHorse(horse1, laneNumber:1);
8           race.addHorse(horse2, laneNumber:2);
9
10          race.startRace();
11      }
12  }
13
```

PROBLEMS 13 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
================================
|              X               | Frank (Current confidence 0.9)
|         X                    | Bob (Current confidence 0.9)
================================


All horses have fallen, No winner.
PS C:\Users\User\Documents\cs\OOP\project V2>
```

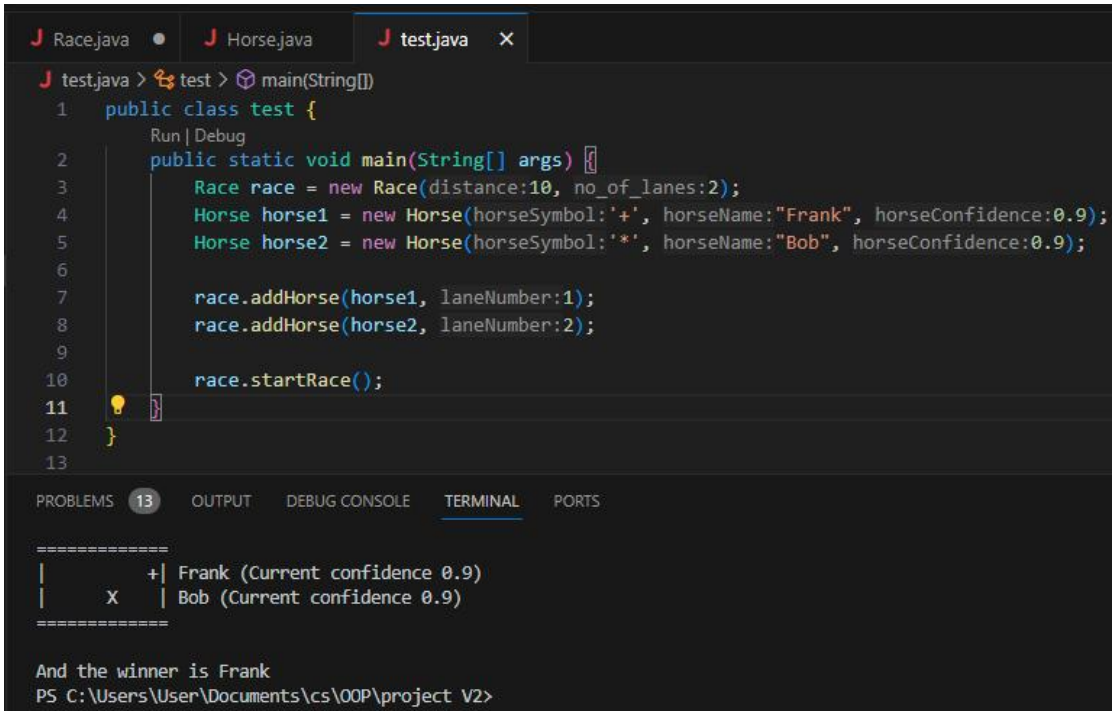
10)To print out the name and confidence of the horse, I added a print function at the end of the method printLane to print out the horses name and its confidence.

Race.java | Horse.java | test.java

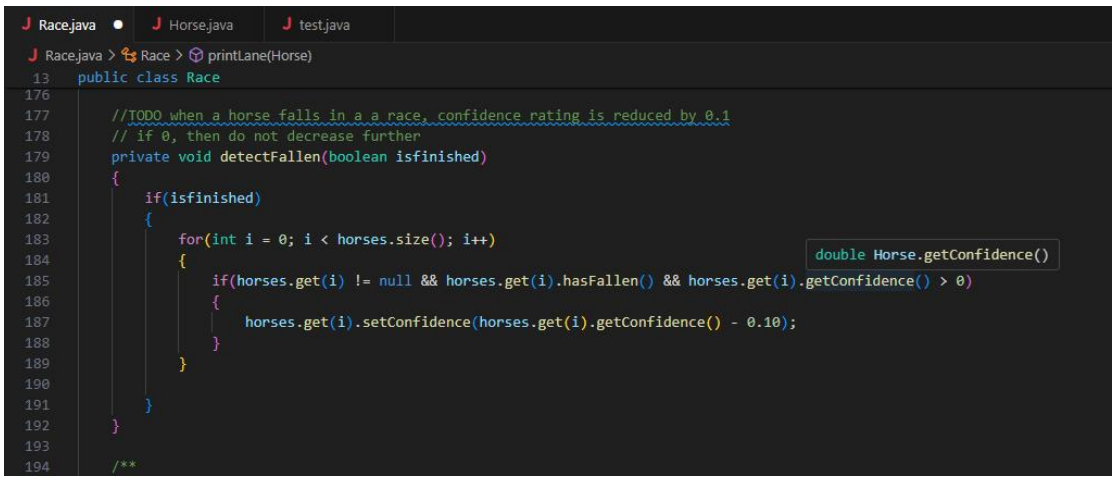
Race.java > Race > printLane(Horse)

```
13  public class Race
272     private void printLane(Horse theHorse)
309             System.out.print(c:'|');
310
311             //TODO after every lane, print name and confidence
312             System.out.print(" "+ theHorse.getName() + " (Current confidence " + theHorse.getConfidence() + ")");
313     }
314
315     }
316
317
```

Testing



11,12) After the race has finished, confidence has to be decreased after a horse has fallen, however, the minimum confidence must not be less than 0. The new method, detectFallen loops through the arrayList of horses if the race has ended. If any of the horses fell, then the confidence is decreased by 0.1. the additional test horses.get(i).getConfidence() > 0 checks if the confidence level is more than 0. If false, the confidence will not get decreased. The method detectFallen is then called in the method startRace.

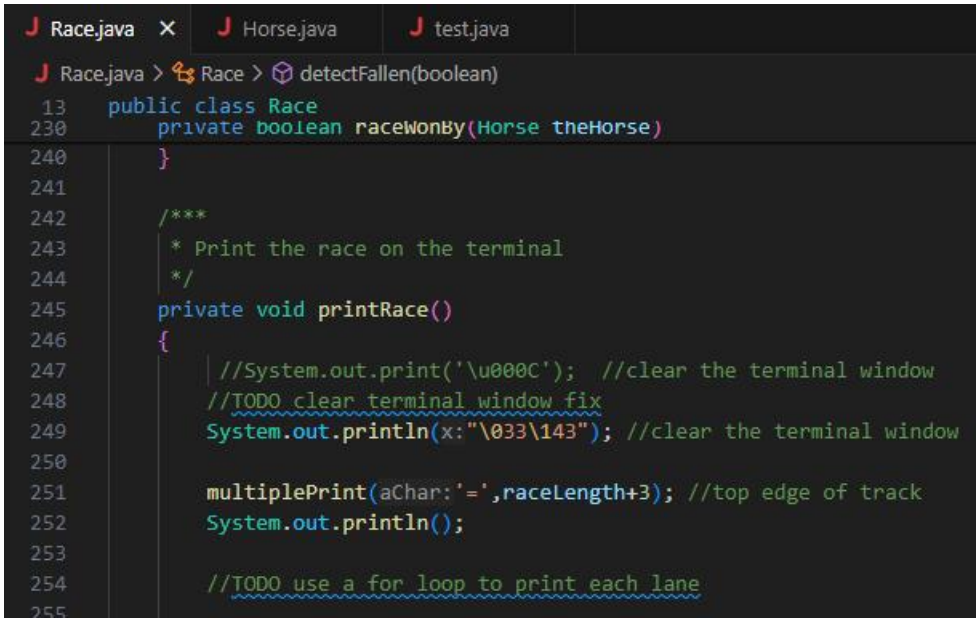


```
Race.java > Race > printLane(Horse)
13  public class Race
64      public void startRace()
                    System.out.println();
                    System.out.println(x:"All horses have fallen, No winner.");
                }

                detectFallen(finished);

                //wait for 100 milliseconds
                try{
                    TimeUnit.MILLISECONDS.sleep(timeout:100);
                }catch(Exception e){}
            }
        }


    }
```

Testing
Initially the confidence for horse1 is set to 0.9. when it falls it is reduced by 0.1 to 0.8. On the other hand, horse2 had a confidence level set to 0.5 and after it wins, the confidence is increased to 0.6

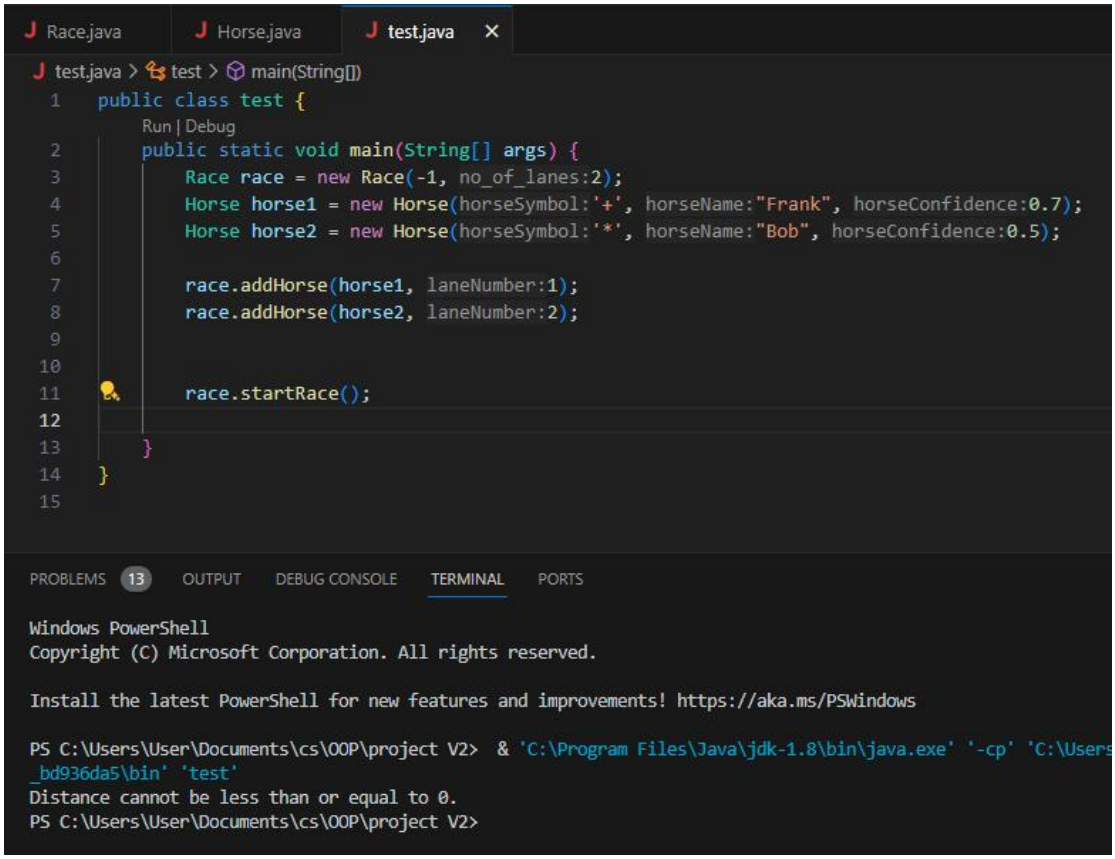
```
Race.java  Horse.java  test.java
test.java > test > main(String[])
1   public class test {
        Run | Debug
2       public static void main(String[] args) {
3           Race race = new Race(distance:20, no_of_lanes:2);
4           Horse horse1 = new Horse(horseSymbol:'+', horseName:"Frank", horseConfidence:0.9);
5           Horse horse2 = new Horse(horseSymbol:'*', horseName:"Bob", horseConfidence:0.5);
6
7           race.addHorse(horse1, laneNumber:1);
8           race.addHorse(horse2, laneNumber:2);
9
10          race.startRace();
11
12          System.out.println( horse1.getName() + " has a confidence of " + horse1.getConfidence());
13          System.out.println( horse2.getName() + " has a confidence of " + horse2.getConfidence());
14      }
15  }
16

PROBLEMS 13  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

======================
|          X         | Frank (Current confidence 0.9)
|                   *| Bob (Current confidence 0.5)
======================

And the winner is Bob
Frank has a confidence of 0.8
Bob has a confidence of 0.6
PS C:\Users\User\Documents\cs\OOP\project V2>
```

13)There is also an issue where the terminal window does not clear properly as the original character code provided (‘\u000C’) was incorrect. This is corrected to “\033\143”



14) Input validation for distance. If the race object is created with a distance of less than or equal to 0, then the race will not start. This is done by using an if statement in the method startRace.

Testing



15) When starting a race, if one of the horses is created with a confidence level of less than 0 or more than 1, the race will not start. To do this, I implemented an if statement in the method startRace which calls the method checkConfidence. The method checkConfidence uses a for loop and an if statement to check if any of the horse's confidence level is > 0 or < 0. if true, it prints out a warning message and returns true.

```
Race.java ● | Horse.java | test.java
Race.java > Race > startRace()
13  public class Race {
59      public void startRace() {
79          System.out.println(x:"Distance cannot be less than or equal to 0.");
80      }
81      //TODO input validation for confidence
82      if (checkConfidence()) {
83          finished = true;
84          System.out.println(x:"Confidence cannot be less than 0 or greater than 1.");
85      }
86
```

Testing

```
Race.java | Horse.java | test.java ×
test.java > test > main(String[])
1   public class test {
        Run | Debug
2       public static void main(String[] args) {
3           Race race = new Race(distance:10, no_of_lanes:2);
4           Horse horse1 = new Horse(horseSymbol:'+', horseName:"Frank", -0.7);
5           Horse horse2 = new Horse(horseSymbol:'*', horseName:"Bob", horseConfidence:0.5);
6
7           race.addHorse(horse1, laneNumber:1);
8           race.addHorse(horse2, laneNumber:2);
9
10
11          race.startRace();
12
13      }
14  }
15

PROBLEMS 14  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\User\Documents\cs\OOP\project V2>  & 'C:\Program Files\Java\jdk-1.8\bin\java.exe' '-cp' 'C:\U
_bd936da5\bin' 'test'
Confidence for Frank is -0.7
Confidence cannot be less than 0 or greater than 1.
PS C:\Users\User\Documents\cs\OOP\project V2>
```

16) For the program to accept either meters or yards, I need to overload the constructor. I created a new constructor with a different signature than the original constructor and it has a signature of int, String, int where the String is the unit. In this new constructor I use if-then-else statements to determine the distance., if the unit is “metres” or “m”, I assign the distance to the instance variable raceLength. If it is “yards” or yd, then the code converts it to metres by mulitplying 0.9144, then converting it

into an integer.. If any other units are entered, then it prints out an error message and sets the distance to 0, which prevents the race from starting.

```java
public class Race {
    // TODO Constructor to accept units (m or yd) for distance.
    // yards are converted to metres.
    //@param unit. The unit of the distance. Either metres(m) or yards(yd)
    public Race(int distance, String unit, int no_of_lanes) {
        // if its metres, no conversion needed
        if(unit.equals(anObject:"metres") || unit.equals(anObject:"m")){

            this.raceLength = distance;

        }
        // if its yards, convert to metres by x0.9144
        else if(unit.equals(anObject:"yards") || unit.equals(anObject:"yd")){

            double distance_converted_to_m = distance * 0.9144;
            this.raceLength = (int) distance_converted_to_m;

        }
        // if unit not accepted, set distance to 0 which prevents race from starting
        else{
            System.out.println(x:"Invalid unit, distance automatically set to 0. Please enter either metres(m) or yards(yd).");
            this.raceLength = 0;
        }

        horses = new ArrayList<Horse>();
        for (int i = 0; i < no_of_lanes; i++) {
            horses.add(e:null);
        }
    }
```

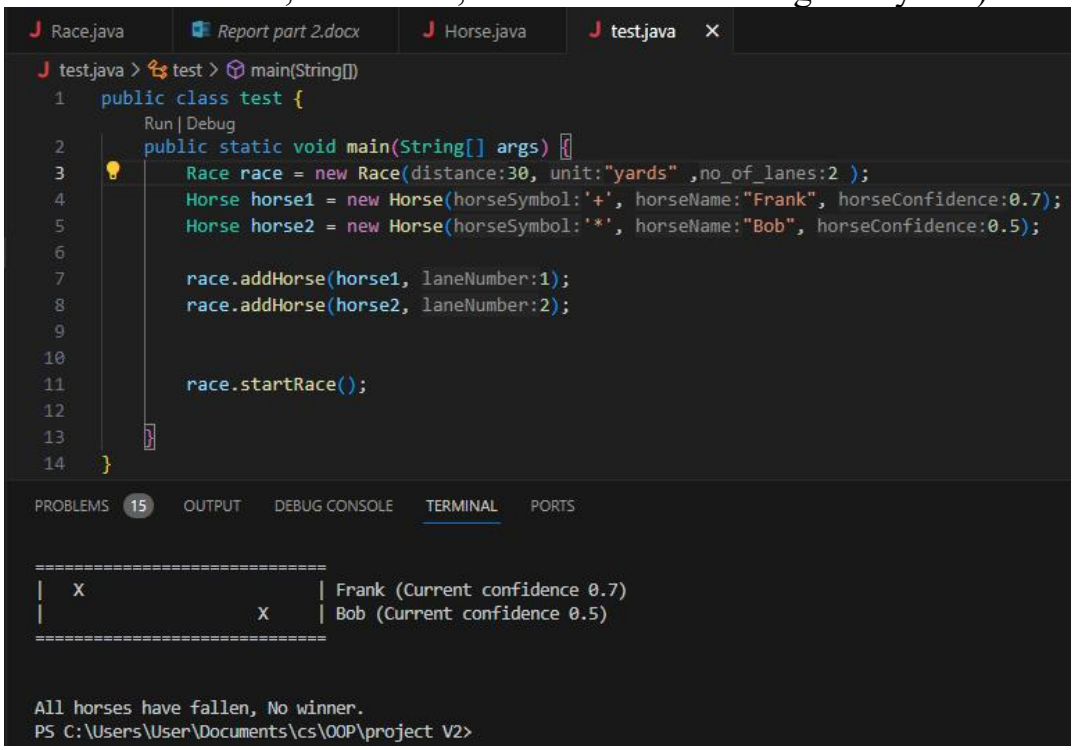
Testing
For metres:

```java
public class test {
    Run | Debug
    public static void main(String[] args) {
        Race race = new Race(distance:30, unit:"m" ,no_of_lanes:2 );
        Horse horse1 = new Horse(horseSymbol:'+', horseName:"Frank", horseConfidence:0.7);
        Horse horse2 = new Horse(horseSymbol:'*', horseName:"Bob", horseConfidence:0.5);

        race.addHorse(horse1, laneNumber:1);
        race.addHorse(horse2, laneNumber:2);


        race.startRace();

    }
}
```

```
==================================
|     X                          | Frank (Current confidence 0.7)
|   X                            | Bob (Current confidence 0.5)
==================================


All horses have fallen, No winner.
PS C:\Users\User\Documents\cs\OOP\project V2>
```

For Yards:

Note(the track is a little shorter than meters, the distance is set to 30 for both tests however, in this test, meters has been changed to yards)



For any other units (eg.cm):
As expected, the code will display an error message and the race would not start.