Lab 3: Horse Race (Classes & Objects)

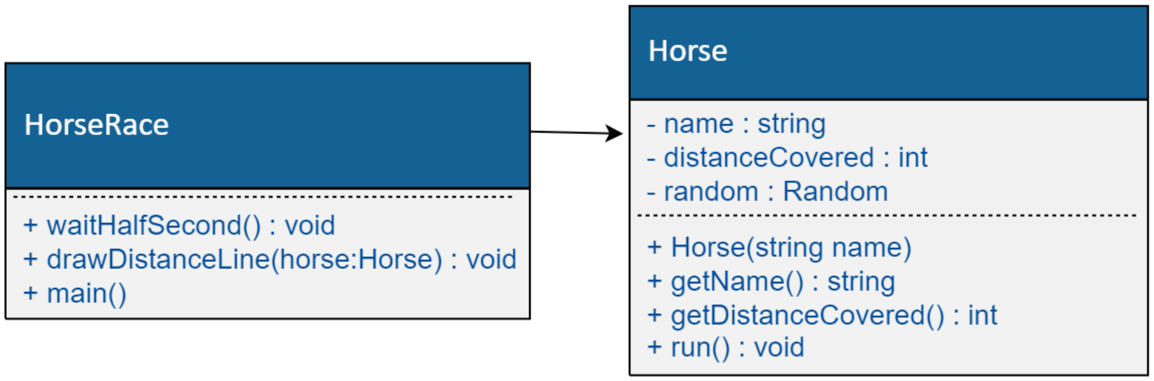
OOP1 – Fred Stiebler

Program Description

In this lab you will write a Java Console App that will simulate a **horse race.**

Required Classes and Methods

The UML Class Diagram below shows the class structure required for this app.



The HorseRace Class

Static methods

1. `waitHalfSecond()` stops code execution for half a second.
   💡 Hint 1: Create a **constant** named HALF_SECOND with a value of 500. We will be measuring time in milliseconds, so 1 second is 1000 milliseconds, and half a second is 500 milliseconds.
   💡 Hint 2: Call the method `sleep` from the class `Thread`. It can generate an exception so put this inside a try-catch: `Thread.sleep(HALF_SECOND)`
2. `drawDistanceLine()` Receives a `Horse` as parameter and draws a line of `'.'` characters representing the distance covered by the horse. 💡 **Hint**: use a `for` loop to draw `'.'` characters

The Horse Class

Private variables

1. `name` holds the horse's name. It **starts without a value**.
2. `distanceCovered` holds the **total distance currently covered by the horse**.
   It starts with a **default value of 0**.
3. `random` will initialise a new **object of the Java's built-in class** `Random`.

Constructor method

- Initialises a new `Horse` object by **receiving its name** as a **parameter**.
- Stores the new horse's name in the `private` variable **name**.

Getter (accessor) methods, one for each of the Horse's private variables.

3. `getName()` should be called **to access the horse's name**.
4. `getDistanceCovered()` should be called **to access the distance covered by the horse**.

The `public` method `run()`

- Makes the horse **run at a random speed**, by having a 50% chance of adding 1 to the `private` variable `distanceCovered`.
- **Will add a random number from `0` to `1`** (inclusive) to the `private` variable `distanceCovered`.

Simulating the Race

You must instantiate at least 5 Horse objects. To display the race you will need a **loop structure** that will keep calling the `run()` method **from each Horse object,** and then pass each **Horse** to the `drawDistanceLine()` to draw their **distance covered** using lines made out of the `'.'` character.

The race is updated twice per second (2 FPS). The easiest way of achieving that is by **waiting half a second after each iteration** of the loop structure. E.g. within `main()`

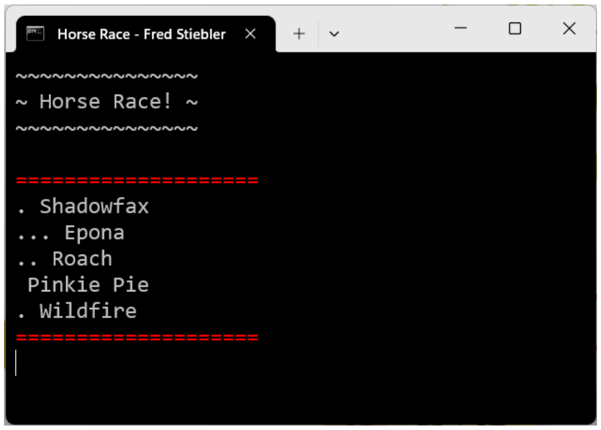
When a Horse covers a distance of 20 `'.'` characters, we have a winner, and the race is over.

⚠ More than one **Horse** could win the race, if they covered 20 at the same time!

```
// Keep updating the race until a horse wins
while(racing)
{
    // Race logic here
    waitHalfSecond(); // Will stop execution for half a second
}
```

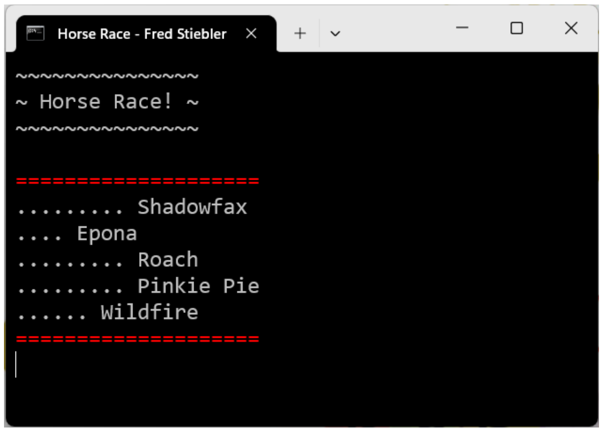
Console Output Examples

Try to match these output images as close as possible (**colours are optional**)



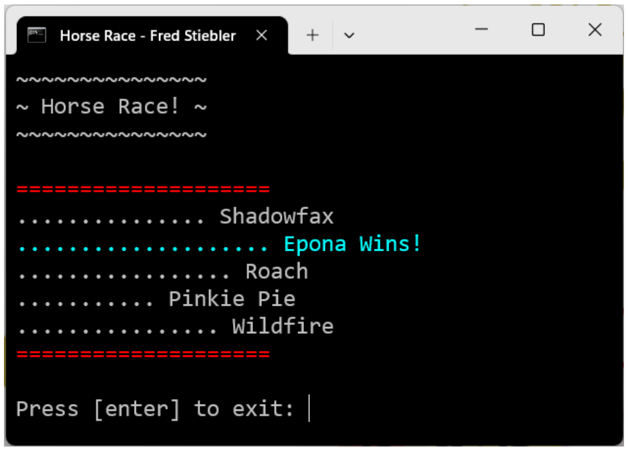
```
~~~~~~~~~~~~~~~
~ Horse Race! ~
~~~~~~~~~~~~~~~

====================
. Shadowfax
... Epona
.. Roach
 Pinkie Pie
. Wildfire
====================
```



```
~~~~~~~~~~~~~~~
~ Horse Race! ~
~~~~~~~~~~~~~~~

====================
......... Shadowfax
.... Epona
......... Roach
......... Pinkie Pie
...... Wildfire
====================
```



```
~~~~~~~~~~~~~~~
~ Horse Race! ~
~~~~~~~~~~~~~~~

====================
............... Shadowfax
.................... Epona Wins!
................. Roach
........... Pinkie Pie
................ Wildfire
====================

Press [enter] to exit:
```

⭐ Bonus Marks Opportunity (up to 5%)

Update your code to use arrays!

1. Replace the many `Horse` **objects** in your `main()` with an **array that each index holds a** `Horse` **object.**
2. Update the rest of the code to **access horses from the array** instead of from many `Horse` objects**.**
   - ✗ **Do not access each index manually**. I.e. horses[1] horses [2] etc
   - ✓ **Use a for loop to go through each index.** I.e. horses[index]

Style Guide and Documentation

To be eligible for full marks on this or any lab in this course your application must conform to the requirements as outlined above and the course Style Guide, in this case making sure to include:

- Your code follows our Java style guide.
- **Appropriately declared data types for all possible variables and constants.**
- Appropriate and complete **program documentation** (code comments).