**Horse Racing Simulation Assignment**

**Overview**

Welcome to the Horse Racing Simulation assignment! In this collaborative project, you and your partner(s) will create a Horse Racing simulation using Java. The goal is to reinforce your understanding of the first 5 units of the AP Computer Science curriculum, encourage collaboration, and enhance your skills in using git and GitHub.

**Project Components**

**1. Provided Code**

You will start with a base code that includes a **Horse** class, a **Race** class, and a **HorseRacingHelper** class. These classes encapsulate essential functionalities for simulating horse races. Your primary task is to build upon this foundation, adding features to create an engaging simulation.

**2. Requirements**

a. Betting System

Implement a betting system where users can place bets on horses to win, place, or show. Display the odds for each bet type based on horse ratings and race type. Optionally, students can explore advanced bet types such as boxing and exactors.

b. Race Simulation

Modify the **HorseRacing** class to allow users to participate in multiple races. Display race information, including the type of race and preferences of each horse. Allow users to make bets and simulate races. Students should ensure the results of the races match the preferences of the horses and how well they match up with a horse’s preferences.

c. Documentation

Provide a detailed explanation of the implemented features, including how preferences affect race outcomes and the logic behind the betting system.

**Betting Terminology**

* **Win:** A bet placed on a horse to finish first.
* **Place:** A bet placed on a horse to finish either first or second.
* **Show:** A bet placed on a horse to finish first, second, or third.
* **Boxing:** A bet where you select multiple horses to finish in any order. For example, in a boxed exacta, the selected horses can finish first and second in any order.
* **Exacta:** A bet where you predict the horses that will finish first and second in the correct order.

Betting odds represent the potential profit you can make relative to your stake. There are different formats for expressing odds, and in horse racing, fractional odds are commonly used. In fractional odds, the first number represents the potential profit, and the second number represents the stake.

For example:

* **3-1 Odds:** For every $1 you stake, you can win $3. If you win, you get back your stake ($1) plus the profit ($2).
* **3-2 Odds:** For every $2 you stake, you can win $3. If you win, you get back your stake ($2) plus the profit ($1).

Let's create an example race with 5 horses and calculate the odds for each horse based on their preferences and ratings. Assume it's a dirt race with a length of 6.5 furlongs.

A screenshot of a computer

Description automatically generated

Now, let's break down the odds for the first horse, Speedy Sprinter:

* **Odds for Win (3-1):** For every $1 you bet, you can win $3. If you bet $10 and Speedy Sprinter wins, you get back $30 (profit of $20 plus your initial $10 stake).
* **Odds for Place (5-2):** For every $2 you bet, you can win $5. If you bet $10 and Speedy Sprinter finishes first or second, you get back $25 (profit of $15 plus your initial $10 stake).
* **Odds for Show (2-1):** For every $1 you bet, you can win $2. If you bet $10 and Speedy Sprinter finishes first, second, or third, you get back $20 (profit of $10 plus your initial $10 stake).

Repeat the same process for each horse based on their odds. The odds are influenced by the horse's ratings, preferences, and the type of race.

The odds for each horse are determined based on a combination of their ratings, preferences, and the type of race. Here's a breakdown of the factors influencing the chosen odds for each horse:

1. **Speedy Sprinter (Horse #1):**
   * This horse has high ratings for both dirt and grass, indicating versatility.
   * Preferred length matches the race length, providing an advantage.
   * Well-rounded performance across terrains suggests favorable odds for all bet types.
   * Chosen odds reflect the horse's overall strength in the race.
2. **Mud Master (Horse #2):**
   * Strong preference for mud races gives an advantage in a dirt race with a high mud rating.
   * Lower ratings in other terrains influence slightly longer odds.
   * The odds reflect the specific advantage in mud and balanced consideration for other terrains.
3. **Grass Gallop (Horse #3):**
   * High grass rating indicates strong performance on grass.
   * Balanced ratings across terrains suggest consistent performance.
   * Preferred length matches the race length, contributing to favorable odds.
   * Chosen odds reflect the horse's strength on grass and overall competitiveness.
4. **Swift Strider (Horse #4):**
   * Does not like Dirt races. For that reason, this is a long shot.
   * The preferred length is longer than the race length and this horse may get tired. Decreasing odds of winning.
5. **Terrain Tracker (Horse #5):**
   * High dirt rating suggests strength in a dirt race.
   * Slightly lower odds compared to top contenders due to preferences being less aligned with the race type.
   * Balanced ratings contribute to reasonable odds for all bet types.

In summary, the chosen odds aim to simulate a realistic representation of each horse's strengths and weaknesses based on their ratings, preferences, and the race conditions. Horses with stronger alignments to the race type and length generally have shorter odds, reflecting a higher likelihood of success, while others have longer odds to provide a fair betting landscape.

Top of Form

**Evaluation Criteria**

40% of your grade will be based on an interview with the teacher to assess your understanding of the code. During the interview, you should be prepared to explain the logic behind your code, discuss design choices, and demonstrate a clear comprehension of the implemented features. This ensures that you not only write code collaboratively but also understand and can articulate the reasoning behind your contributions.

**Collaboration Guidelines**

1. **Communication:** Keep communication open and clear with your partner(s). Discuss decisions, share insights, and ask for feedback.
2. **Code Reviews:** Regularly review each other's code to catch errors, suggest improvements, and maintain consistent quality.
3. **GitHub Practices:** Ensure code on GitHub is functional before pushing updates. Retrieve the latest code from GitHub before making your changes.
4. **Responsibilities:** Clearly define roles and tasks within the group based on individual strengths and expertise.
5. **Documentation:** Thoroughly document your code with comments and explanations. This helps everyone understand the project.
6. **Meetings:** Schedule regular meetings to discuss progress, address challenges, and plan next steps. Use suitable communication tools.
7. **Respect and Feedback:** Respect each other's ideas and contributions. Provide constructive feedback and be open to receiving feedback.

Effective collaboration improves learning and results in a stronger project.

**Advanced Implementation Expectations for Three-Member Groups**

Within a three-member team context, there is an anticipation for the group to go beyond the foundational project features. Members are encouraged to explore advanced functionalities, such as implementing additional betting options like boxing and exactors, refining simulation algorithms for a more realistic experience, or enhancing the user interface for improved interactivity. With the increased manpower, this presents an opportunity to delve deeper into the project, showcasing creativity and innovation in the implementation. The goal is not just to meet the baseline requirements but to embrace the chance for a more comprehensive and sophisticated project outcome, enriching the learning experience for the entire team.