“Horse” Racing and Training

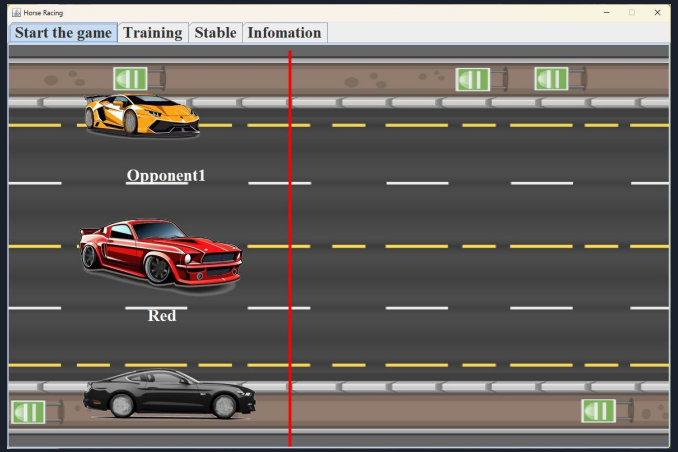
Buddies: Xinye Bao, Jaden Phan, Junyou Guo

Project Overview

- High level overview of the problem statement
  - We made a “horse” racing game, similar to the betting for the Kentucky Derby but with some creative liberties
- Limitations:
  - Single player,
  - Only 3 “horses” to choose from
- Rules:
  - Select a “horse” you think will win
  - If your horse places:
    - 1st: $15
    - 2nd: $5
    - 3rd: -$5

Game description and rules / images

- Very similar to betting on horses like Derby games but modified
- Race against two opponents, each time you win you make money
- Use your money to upgrade your horse by buying feed
- Goal:
  - Train your “horse” to be the fastest and win every time!
  - Earn as much money as possible



Project Requirements

Functional Requirements

- Correctly keep track of each “horse” level
- Random speed of opponents
- Ability to select different “horses” to race as

Non-Functional Requirements

- Animation of “horses” across playing area
- Images should have no background for better visual experience

Project Solution Approach-GUI

Game/UI features:(JavaSwing...)

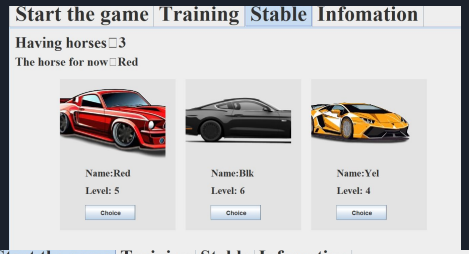
-Gaming modes choosing interface:
Buttons and Car image showing up

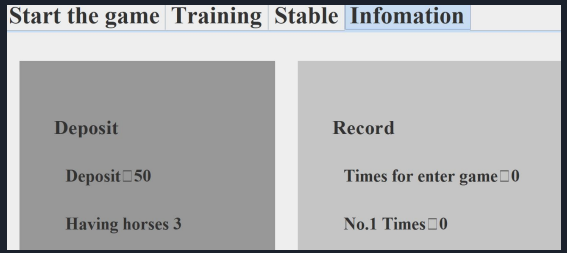
-Training and Stores(upgrade car)

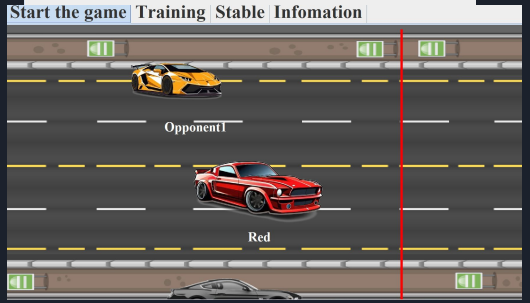
-Stable: to select our current cars

-Player Information:
Properties and Game records

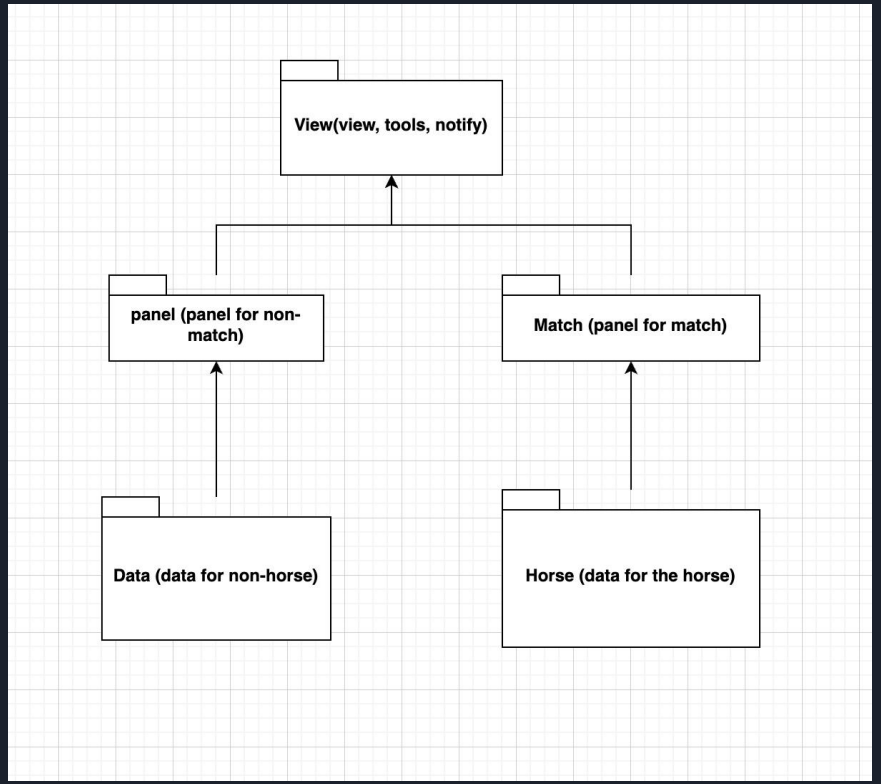
-Dynamic Gaming scene:
The cars are moving on the track







UML Design - We do this with 7 packages



Team Collaboration Approaches

- **Coding Sessions**
  - We did both online chatting and meeting offline to talk about our program process
  - Primarily individual coding for flexibility
- **Discord**

  Our mainly messaging tools,we notice each other when we changed our files
- **GitHub**

  Effective version control
  We completed our projects in staggered manner.
  We have 2 branches, one is Main and one is Jaden's version.

Testing

Testing approaches

- Unit tests-MainGameTest:
  To test if the 3 Buttons which control the miles work
- Integration tests-LevelUpChangeIntegrationTest
  In `LevelUpChange.java`, the `isLevelUp` method modifies the level of the selected horse in `MatchHorses`. We wrote a test to check if the level of the selected horse in `MatchHorses` changes when the `isLevelUp` method is called.
  After this test we optimized the parameters of probability of upgrading the car
- User tests-
  Advantages: The different fonts, and the dynamic effect of car race
  Need Improvement: No feedback when we click any buttons, and in the user's screen he can't see the 3rd car's graph(The size of game Ui should has Universality)

```
public class MainGameTest {
    @Test
    void testMain4Miles() {
        App.main(new String[]{"4"});
        assertNotNull(MainView.getInstance());
    }

    @Test
    void testMain6Miles() {
        App.main(new String[]{"6"});
        assertNotNull(MainView.getInstance());
    }

    @Test
    void testMain8Miles() {
        App.main(new String[]{"8"});
```

```
public class LevelUpChangeIntegrationTest {
    @Test
    void testIsLevelUpChangesSelectedHorseLevel() {

        MatchHorses matchHorses = new MatchHorses();
        HorseBase selectedHorse = new SlowBeforeFastHorse( name: "SelectedHorse",  level: 5,  imgPath: "src/main/
        matchHorses.setSelectedHorse(selectedHorse);

        LevelUpChange levelUpChange = new LevelUpChange();

        levelUpChange.setMatchHorses(matchHorses);

        int initialLevel = matchHorses.getSelectedHorse().getLevel();

        levelUpChange.isLevelUp();
```

Live Demo Time !

Lessons

- We learn that to continue the incomplete codes of someone else is really really hard,so the abundant communication matters a lot.
- The assignments distributions matter a lot.(Let team members focus what they do best.)
- At first we tried to do it 100% separately, and we found that didn't work well and we switch it to group hacking session.