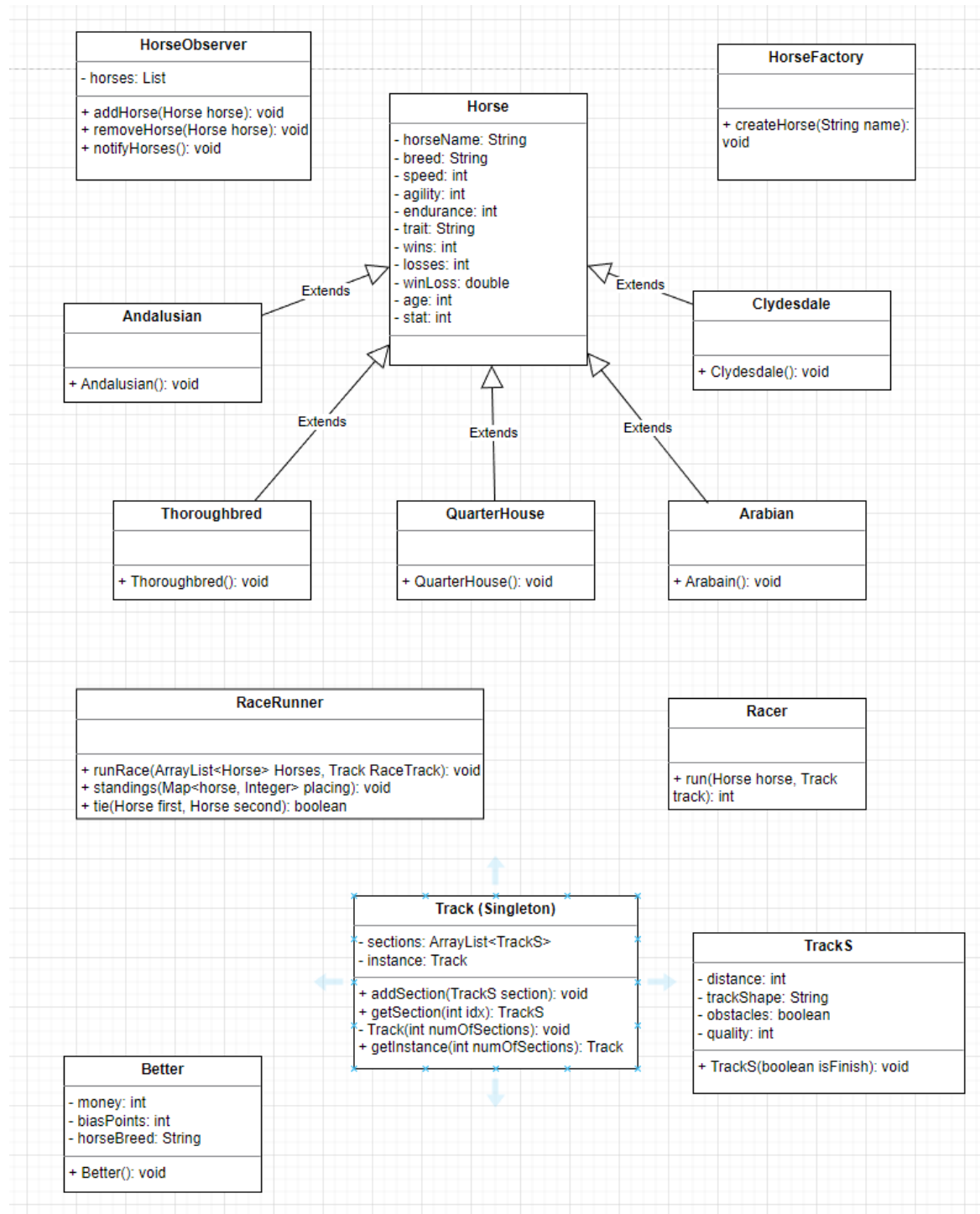Brehn Heil
Tommy Hoang
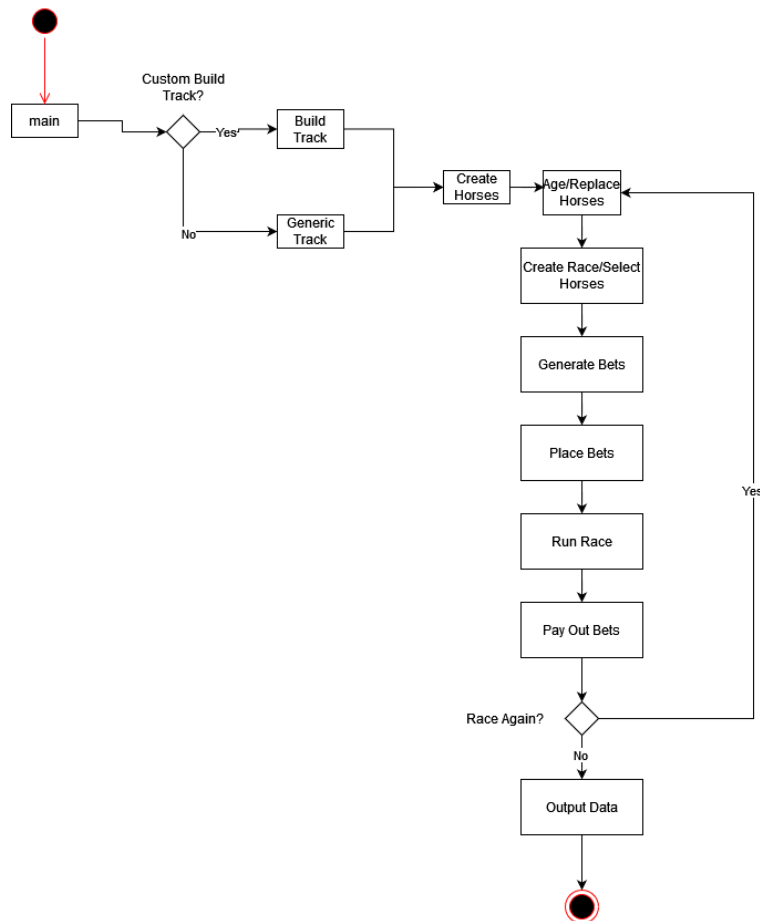Christopher Parker

Project Name: Horse Racing Simulator

Final State of Systems Statement: Unfortunately, we could not complete the fourth design pattern due to time constraints and other circumstances (that we were given an extension for). Other than that, we have completed the other three design patterns and the code is fully functional.

Final Class Diagrams and Comparison Statement:

UML diagram:

HorseObserver

- horses: List

+ addHorse(Horse horse): void
+ removeHorse(Horse horse): void
+ notifyHorses(): void

Horse

- horseName: String
- breed: String
- speed: int
- agility: int
- endurance: int
- trait: String
- wins: int
- losses: int
- winLoss: double
- age: int
- stat: int

HorseFactory

+ createHorse(String name): void

Extends

Andalusian

+ Andalusian(): void

Extends

Clydesdale

+ Clydesdale(): void

Extends   *Extends*   *Extends*

Thoroughbred

+ Thoroughbred(): void

QuarterHouse

+ QuarterHouse(): void

Arabian

+ Arabain(): void

RaceRunner

+ runRace(ArrayList<Horse> Horses, Track RaceTrack): void
+ standings(Map<horse, Integer> placing): void
+ tie(Horse first, Horse second): boolean

Racer

+ run(Horse horse, Track track): int

Track (Singleton)

- sections: ArrayList<TrackS>
- instance: Track

+ addSection(TrackS section): void
+ getSection(int idx): TrackS
- Track(int numOfSections): void
+ getInstance(int numOfSections): Track

TrackS

- distance: int
- trackShape: String
- obstacles: boolean
- quality: int

+ TrackS(boolean isFinish): void

Better

- money: int
- biasPoints: int
- horseBreed: String

+ Better(): void

Class diagram:

Third-Party Code vs. Original code Statement:

Although we did not copy code from geeksforgeeks.org, we have used information from it as it teaches us how to implement a design pattern and clarifies what it is and what it is used for. There are other sites similar to geeksforgeeks.org that we use to learn from, but we did not copy code from them.

Statement on the OOAD Process:

Positive: Learning new design patterns that can create more efficient and organized code that can reduce error as well as time.

Negative: The design patterns can be difficult to implement especially if we already have an easier way of doing it. It may seem that a design pattern can be more time consuming than it is worth.

Negative: UML diagrams don't really seem to impact our way of thinking nor our coding style. It is obvious that it is like a blueprint for our code, but most of the things that we do in the semester projects seem laid out for us so we already know what to implement. UML diagrams seem only beneficial in a real-world situation than in a classroom.