CAT 2

UML Analysis

Alejandro Pérez Bueno

Nov 29, 2024

Table of Contents

Self-Responsibility Declaration

I certify that I have completed CAT2 entirely individually and only with the help deemed appropriate by the teaching staff of this subject according to the FAQs about plagiarism. I understand that unoriginal work and/or the use of generative AI will mean that the submitted activity will not be corrected, and a grade of D will automatically be assigned.

Question 1

Section 1

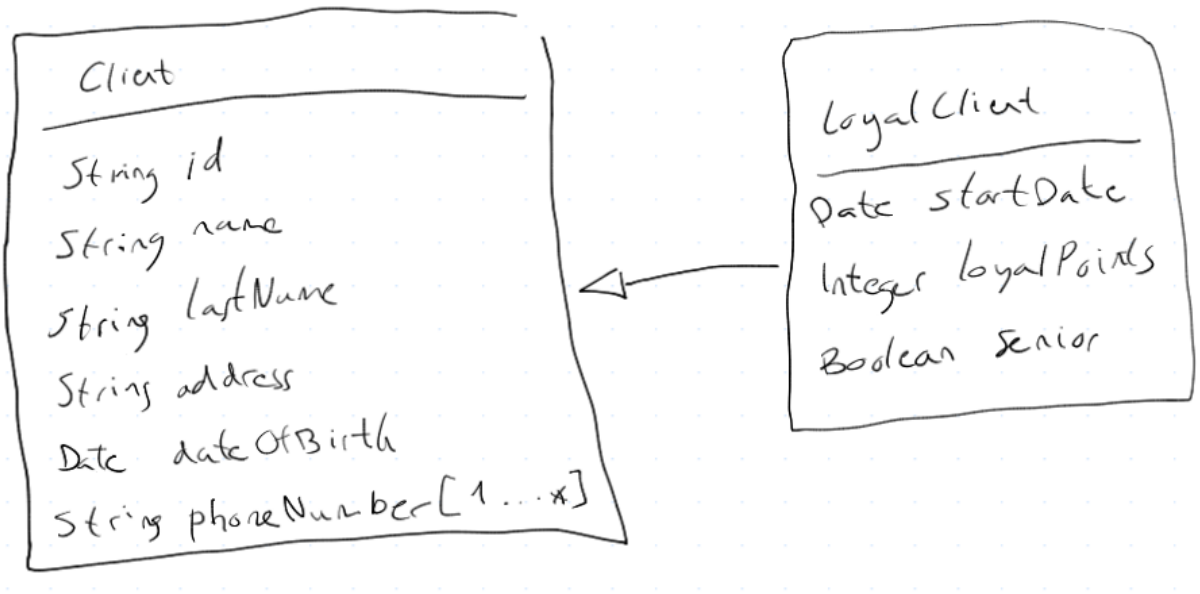


Figure 1: Class Diagram

Section 2

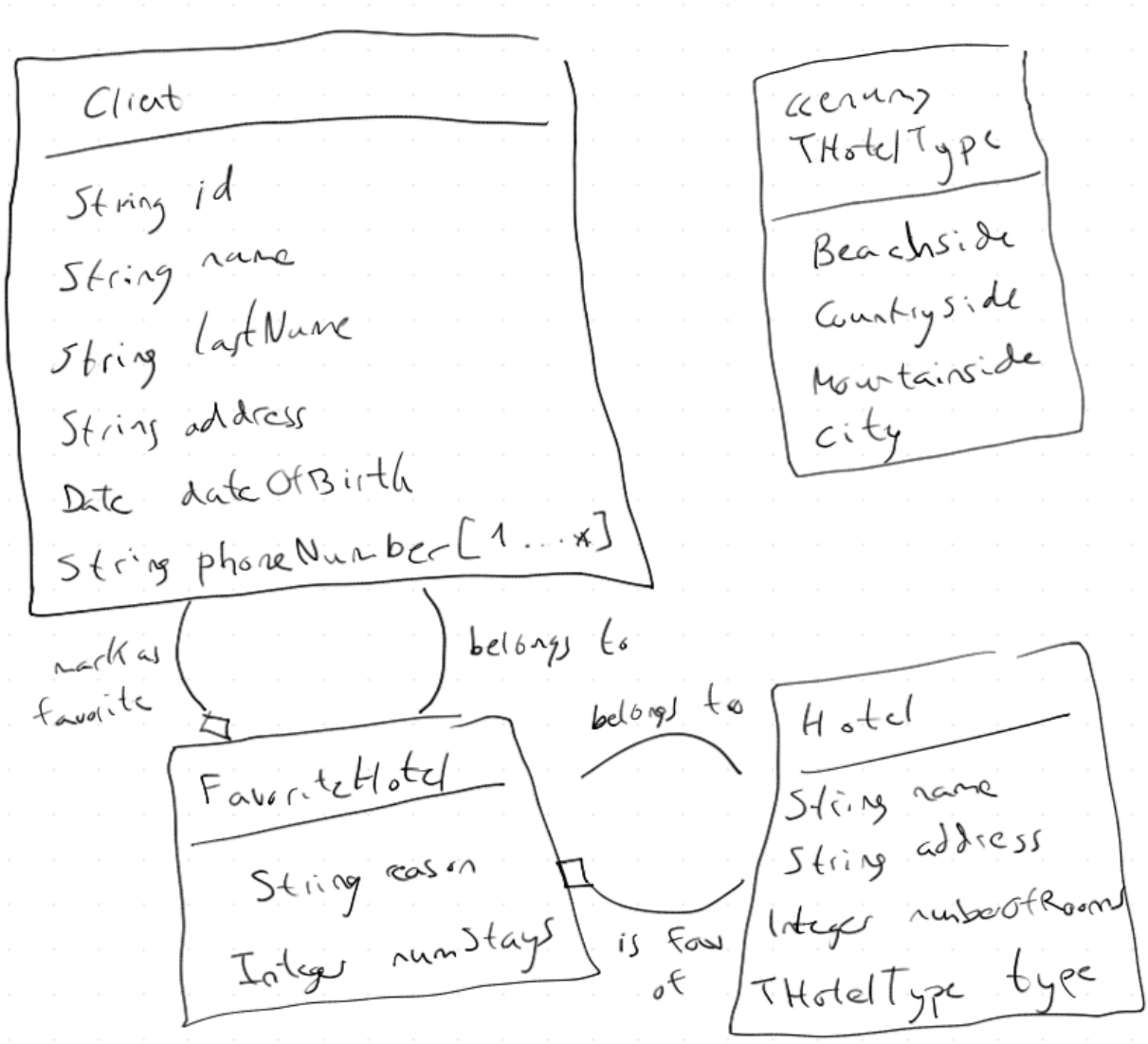


Figure 2: Class Diagram

Section 3

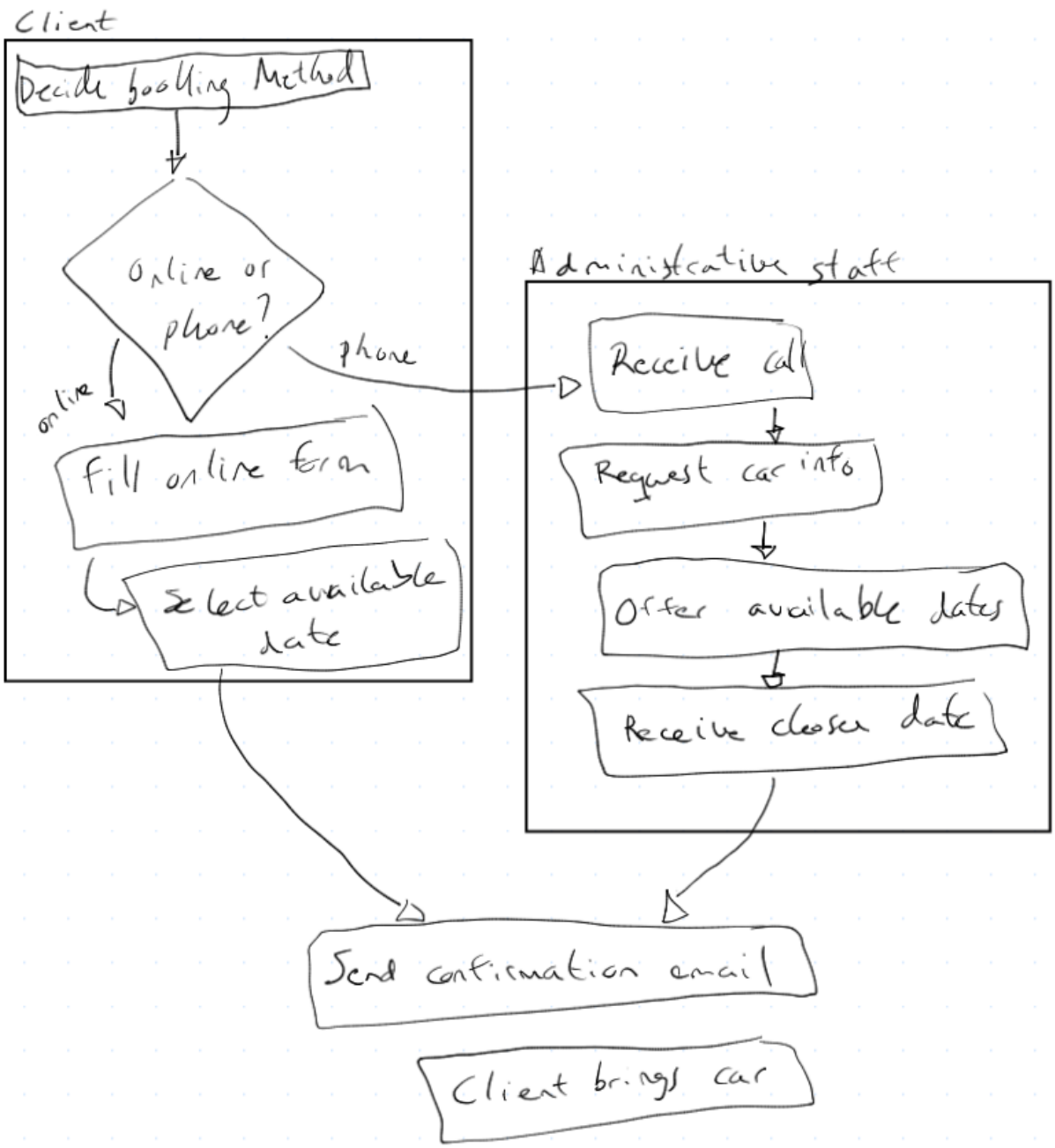
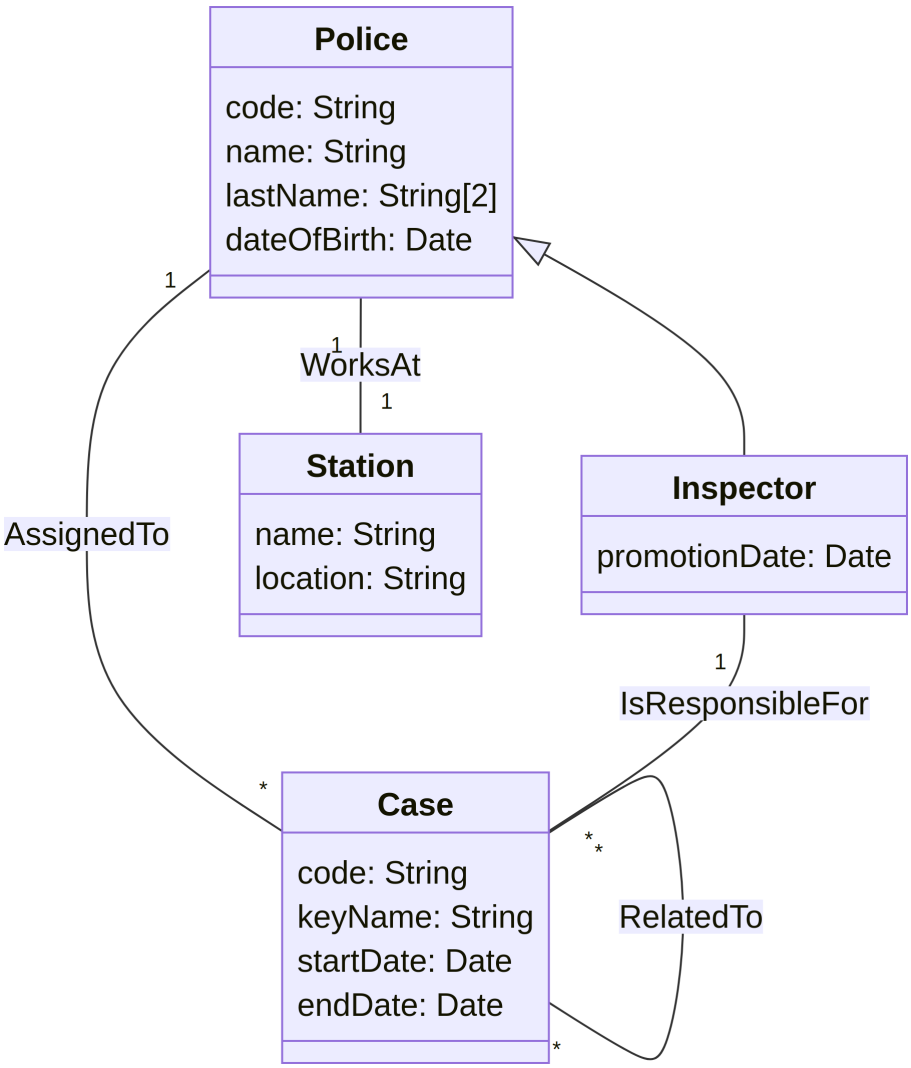


Figure 3: Class Diagram

> Disclaimer
>
> Due to problems installing Visual Paradigm on *Linux*, I will add diagrams drawn with *mermaid.js* or *PlantUML* moving on.

Question 2

Here is the UML Class diagram:



Question 3

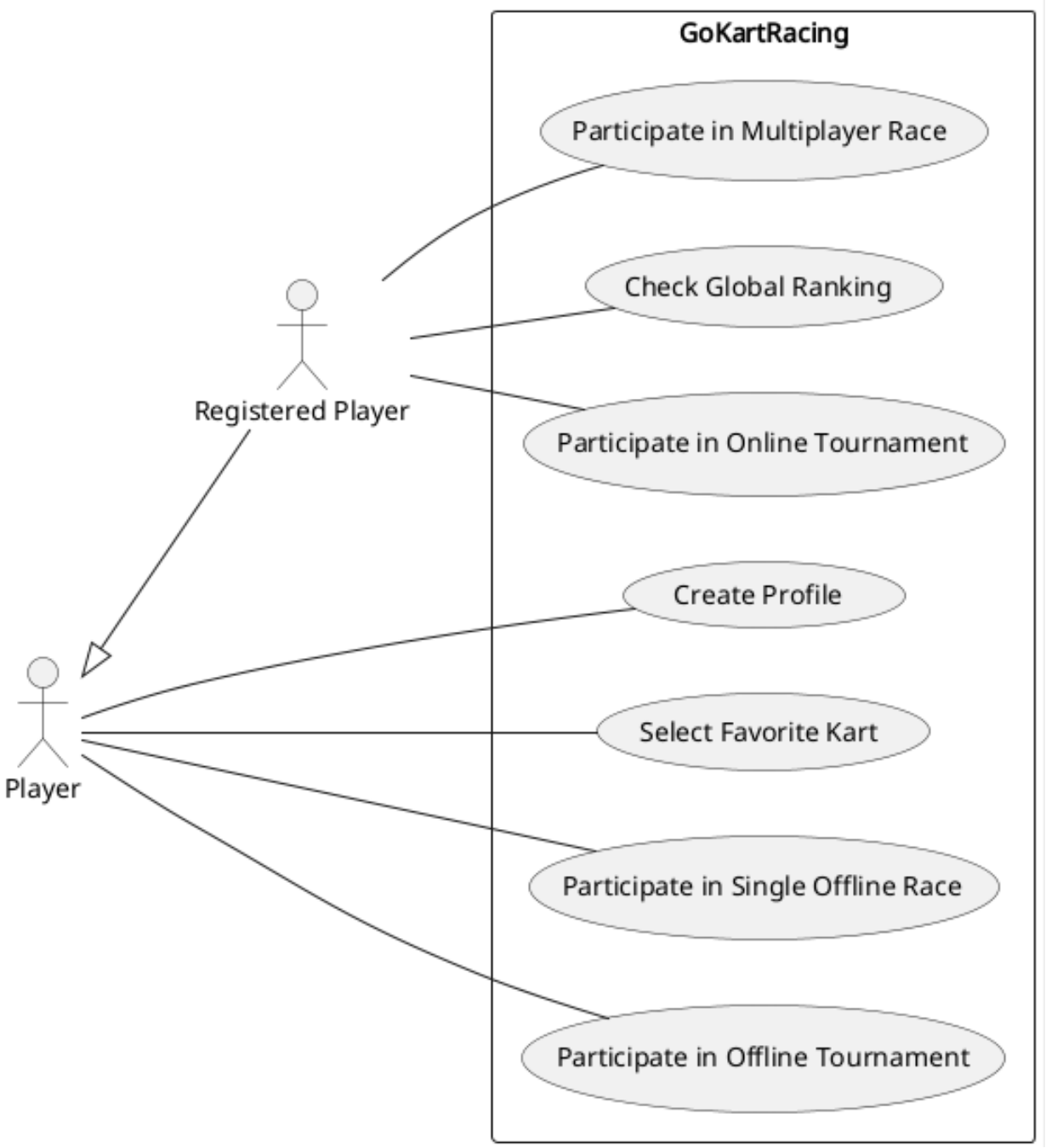


Figure 4: Use case diagram

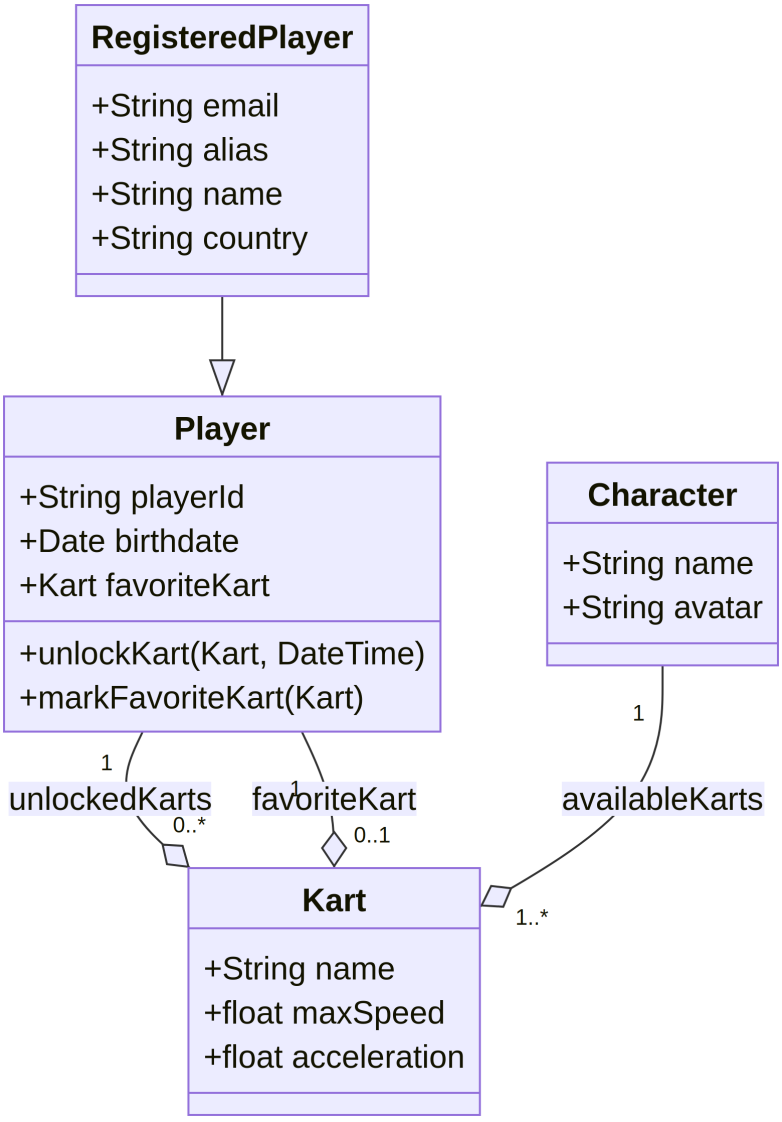
- **Actors:**
  - **Player:** This actor represents anyone who has downloaded the game.

  - **Registered Player:** This actor represents a subset of players who have created an account, allowing access to online features.
- **Use Cases:**
  - **Create Profile:** Allows players to create profiles to store basic information and game preferences.
  - **Select Favorite Kart:** Allows players to choose their preferred kart for races.
  - **Participate in Single Offline Race:** Allows players to race against AI opponents on a single track.
  - **Participate in Offline Tournament:** Allows players to engage in a tournament against AI opponents across four pre-selected tracks.
  - **Participate in Multiplayer Race:** Enables registered players to compete against other players online.
  - **Participate in Online Tournament:** Allows registered players to take part in tournaments with other players online.
  - **Check Global Ranking:** Allows registered players to see their current standing on the global ELO-based ranking system.
- **Relationships:** The lines connecting actors to use cases illustrate that the actor can execute those specific actions within the game.

> **Note**
>
> This diagram is focused on identifying user-level use cases, leaving out lower-level details like "Select Track" or "Choose Opponent". It also avoids depicting any inclusion or extension relationships between use cases.

Question 4

Section A: Player, Character, and Kart Information

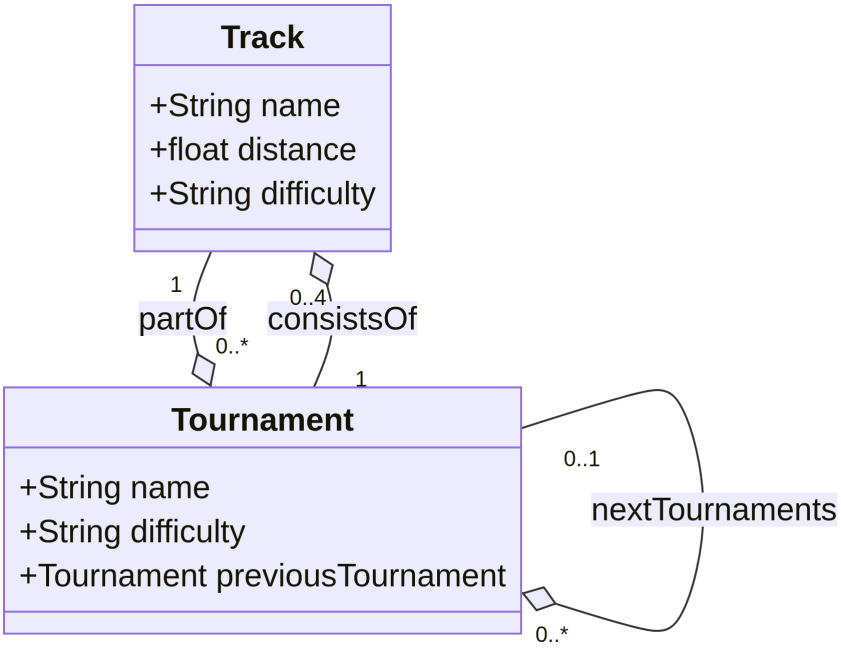


Restrictions, integrity contraints and other derived info:

- **Player Class:**
  - Stores `playerId` and `birthdate`.
  - Has a method `unlockKart(Kart, DateTime)` to record when a kart is unlocked.
  - Has a method `markFavoriteKart(Kart)` to set a favorite kart, which must be unlocked first.
- **RegisteredPlayer Class:**
  - Inherits from `Player`.

  - Stores additional information: `email`, `alias`, `name`, and `country`.
  - **Constraint:** No two `RegisteredPlayer` instances can have the same `email`.
- **Character Class:**
  - Each character has a unique `name` and `avatar`.
  - Each character is associated with one or more karts.
- **Kart Class:**
  - Each kart has a unique `name`, `maxSpeed`, and `acceleration`.
  - A kart is linked to exactly one character.
- **Relationships:**
  - A `Player` can have multiple `unlockedKarts` and one `favoriteKart`.
  - A `Character` can have multiple `availableKarts`.

Section B: Track and Tournament Information



Restrictions, integrity contraints and other derived info:

- **Track Class:**
  - Stores `name`, `distance`, and `difficulty` (easy, medium, high, very high).
- **Tournament Class:**
  - Stores `name` and `difficulty`.
  - **Derived Information:** The `difficulty` of a tournament is determined by the highest difficulty of its tracks.
  - Can have a `previousTournament` and multiple `nextTournaments` to recommend an order of play.

- **Relationships:**
  - A `Track` can be part of multiple `Tournaments`.
  - A `Tournament` consists of exactly four `Tracks`.
  - A `Tournament` can have a recommended `previousTournament` and multiple `nextTournaments`.

Question 5

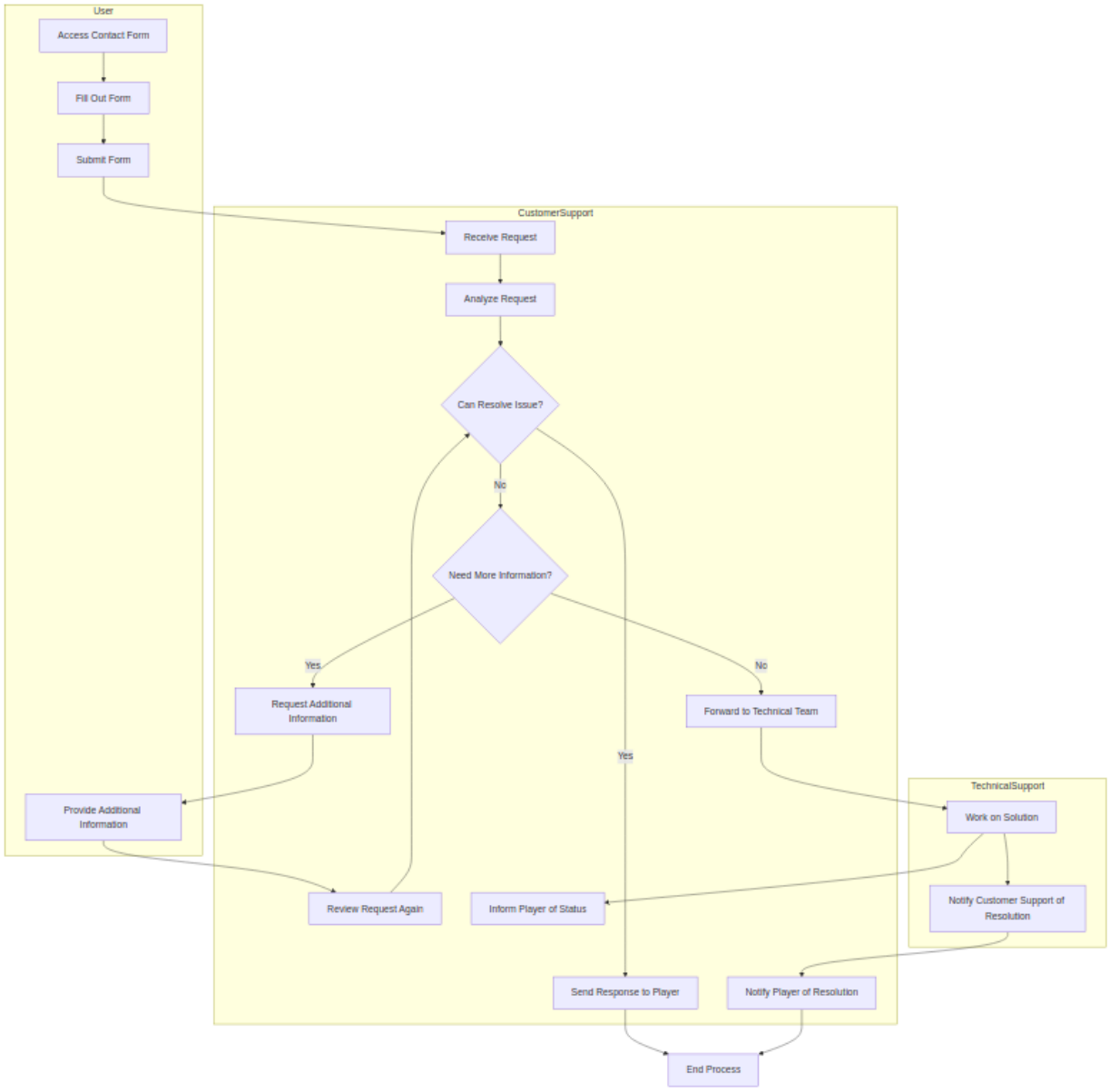


Figure 5: Activity diagram

Explanation:

- **User Swimlane:**
  - **Access Contact Form:** The user accesses the contact form on the website.
  - **Fill Out Form:** The user fills out the form with the necessary details.
  - **Submit Form:** The user submits the form, initiating the process.
  - **Provide Additional Information:** If requested, the user provides additional information to customer support.
- **Customer Support Swimlane:**
  - **Receive Request:** Customer support receives the request from the user.
  - **Analyze Request:** Customer support analyzes the request to determine the next steps.
  - **Can Resolve Issue?:** Decision point to check if customer support can resolve the issue.
    - **Yes:** If the issue can be resolved, customer support sends a response to the player, ending the process.
    - **No:** If the issue cannot be resolved, further steps are taken.
  - **Need More Information?:** Decision point to check if more information is needed.
    - **Yes:** Customer support requests additional information from the user.
    - **No:** The request is forwarded to the technical team.
  - **Review Request Again:** Customer support reviews the request again after receiving additional information.
  - **Forward to Technical Team:** If technical support is needed, the request is forwarded to the technical team.
  - **Inform Player of Status:** Customer support informs the player of the status while the technical team works on the solution.
  - **Notify Player of Resolution:** Once the technical team resolves the issue, customer support notifies the player.
- **Technical Support Swimlane:**
  - **Work on Solution:** The technical team works on resolving the issue.
  - **Notify Customer Support of Resolution:** Once resolved, the technical team notifies customer support.

Question 6

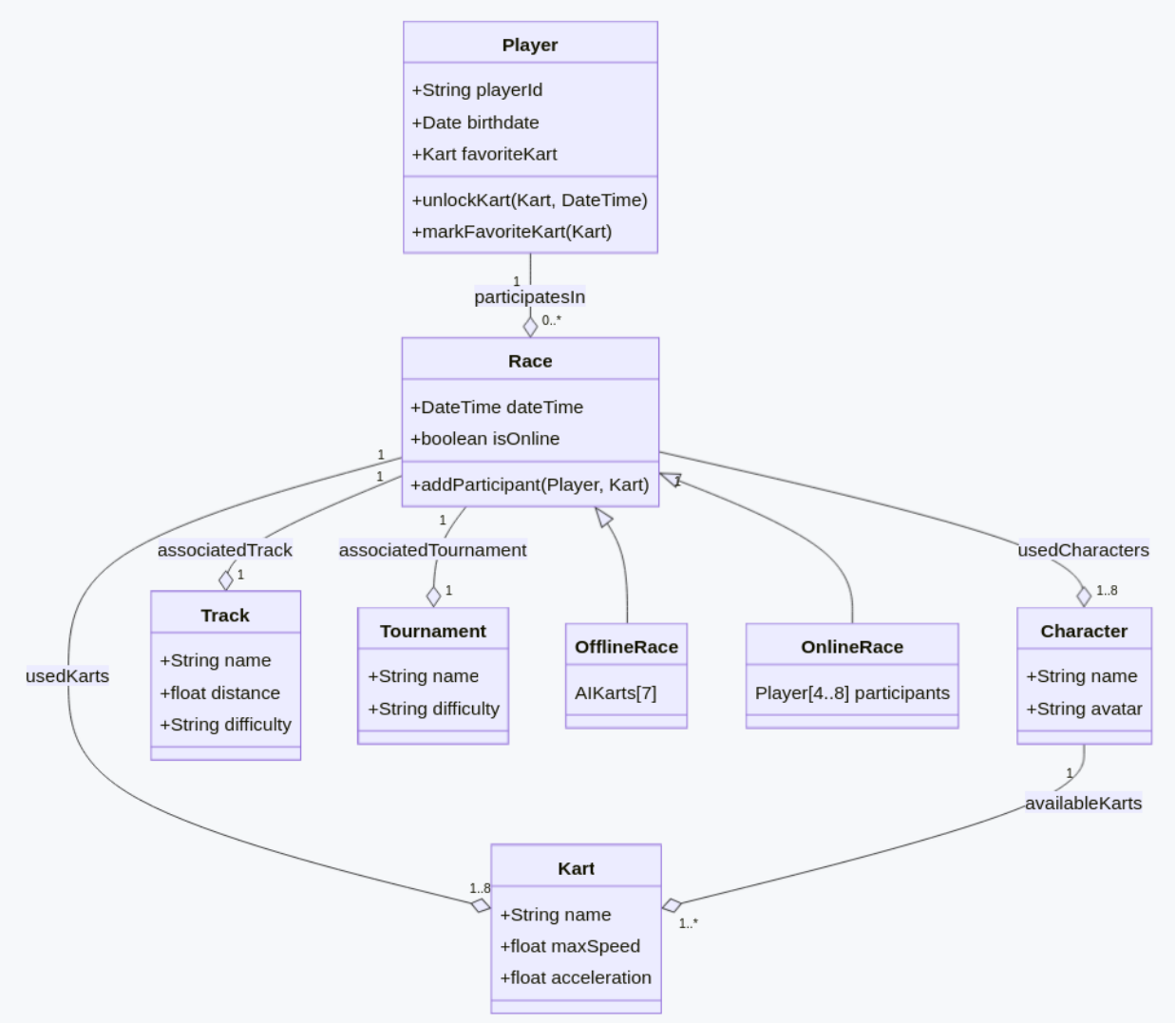


Figure 6: Class diagram

Explanation and Constraints:

- **Player Class:**
  - Players can participate in multiple races, but not at the same date and time.
  - Players must unlock a kart before using it in a race.
- **Kart Class:**
  - Each kart is linked to a single character.
  - A kart can only be used in a race if it is unlocked by the player.
- **Character Class:**

  - Each character has a unique name and avatar.
  - No character can appear multiple times as a racer in a single race.
- **Track and Tournament Classes:**
  - A race is associated with either a track or a tournament, but never both.
- **Race Class:**
  - Stores the `dateTime` of the race and whether it is `isOnline`.
  - Method `addParticipant(Player, Kart)` to add participants to the race.
  - A race can have between 1 and 8 karts and characters, ensuring no character appears more than once.
- **OfflineRace Class:**
  - Inherits from `Race`.
  - Includes 7 AI-controlled karts.
- **OnlineRace Class:**
  - Inherits from `Race`.
  - Includes between 4 and 8 human players as participants.

Key Constraints and Derived Information:

1. **Unique Character Constraint:** No character can appear multiple times in a race. Since each kart is linked to a single character, this ensures that no two karts of the same character are used in the same race.
2. **Race Participation Constraint:** A player cannot participate in multiple races scheduled at the same date and time.
3. **Kart Unlocking Constraint:** A player must unlock a kart before using it in any race.
4. **Race Association Constraint:** A race is associated with either a track or a tournament, but not both.
5. **Participant Count Constraint:**
   - Offline races include 1 player and 7 AI karts.
   - Online races include between 4 and 8 human players.