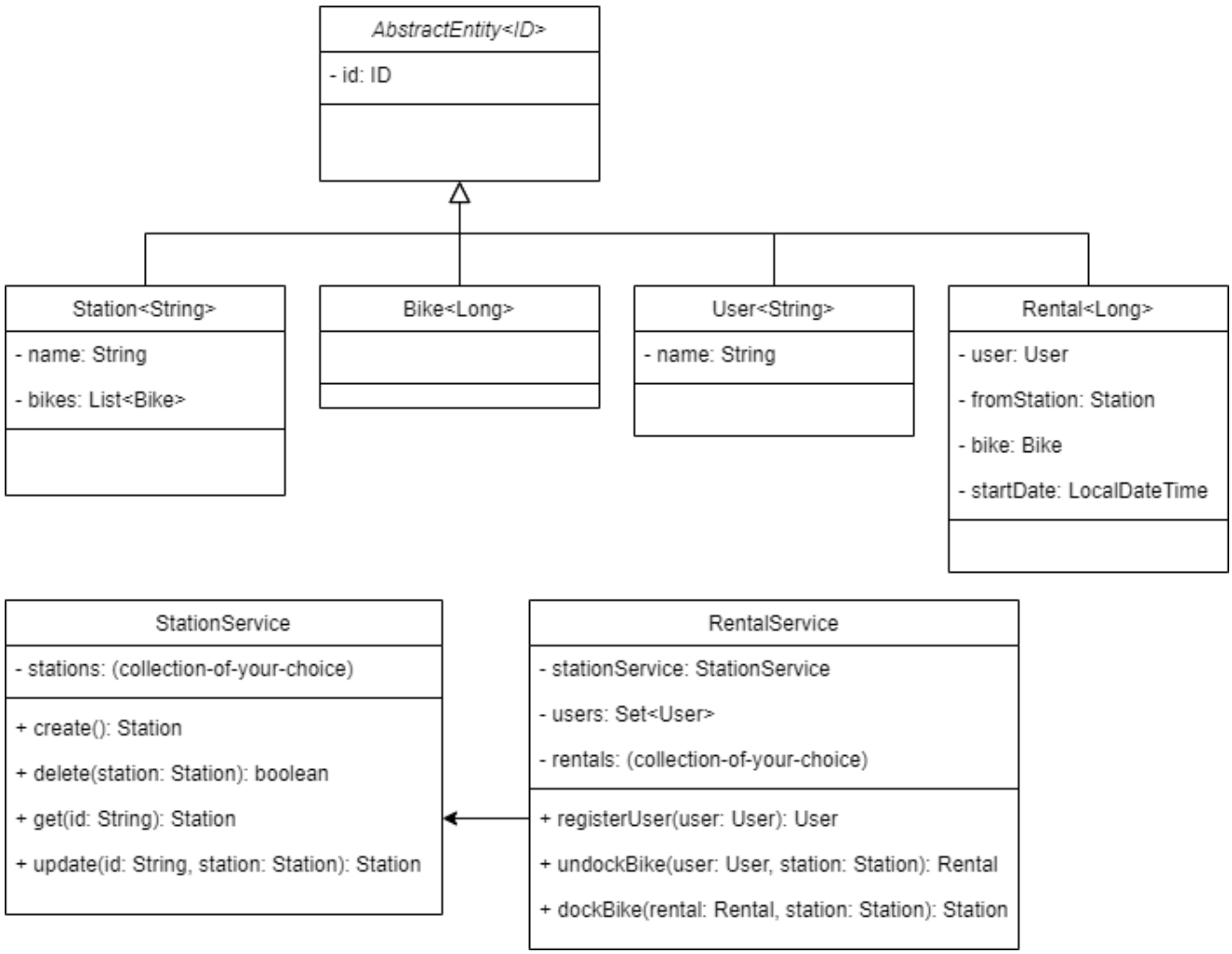
A. Bike rental

The application you'll have to build today represents the backbone of a bike rental system similar to any public bike rental system around the cities in the world. In large terms there will be some stations around the city where anyone could register and get a physical user card. In addition, bikes could be rented from any station and dropped off at any other station if there's enough space to fit the bike. The application should provide some methods in order to fulfill flows like: user registration, stations CRUD (create, read, update, delete) or rental management.

Figure 1: Guideline UML diagram



Requirements

1. Read and understand everything presented below + the UML diagram. Start by creating the classes structure presented above. Feel free to add attributes/classes as you wish in order to fulfill the upcoming requirements. Getters and setters were not added in the diagram but they should be there.

2. Implement `StationService` methods as follows:

- pick a collection of your choice for storing the stations data
- on creating a new station, generate 10 bikes and add them to the newly created station. Assign a unique id to your station (the station id should be one of the alphabet letters; start with 'A' and there won't be more than the available English alphabet letters). Make sure that the bike id is unique across all stations. The station capacity should be fixed: 20 spots.
- get method should return the station with the provided id or throw an unchecked exception

  `StationNotFoundException`
- update method should update station identified by the provided id or throw `StationNotFoundException` if we provide an id of an inexistent station.

3. On deleting a station all the bikes that are docked in that station will have to be moved to other stations beforehand otherwise station deletion must fail. Find the first three stations which have the most free spaces in the bikes array and move the bikes from the station that is about to be deleted to those stations. Assign the bikes to the other stations by trying to fill in all free spaces of the first station, then all the free spaces of the second one and the remaining in the third station. If there are not enough spots for all bikes the station must not be deleted, the bikes must not be reassigned and a checked exception must be thrown (pick a suitable name for it).

4. Implement `RentalService` methods as follows:

- pick a collection of your choice for storing the rentals data
- `registerUser` must assign a unique id to the provided user object and store it to the users set
- `undockBike` must throw an unchecked `EntityNotFoundException` if either the provided user or station is not found or if there is no bike left to pick up from the provided station. On successfully undocking the bike (any bike from the station could be rented), the station bikes array must be updated and a rental object must be created and stored to the rentals collection. A user could only rent a single bike, throw a checked `RentBikeException` if a user tries to rent a second bike.
- `dockBike` should throw a `RentBikeException` if you try to dock a bike to the same station in the first five minutes after picking it up, if there are no spots left in the station where docking is being performed or if the rental/station object is not being found. On successfully docking a bike update the array of bikes for the station where docking happened, remove the rental object from `rentals` collection and return the updated station object.

5. At some point the bike rental system manager would like to see some statistics on how the system is being used. Figure out a way to display to this manager statistics like how many users registered so far, display all the stations + the number of rentals performed on each of them and a list of all bikes + the number of rentals performed on each of them. It is up to you how to store this data, only make sure the statistics are being updated when needed. Create a method to get an object which would consist of all the statistics mentioned above. There is no need to display this data in a formatted way, we only need the numbers.