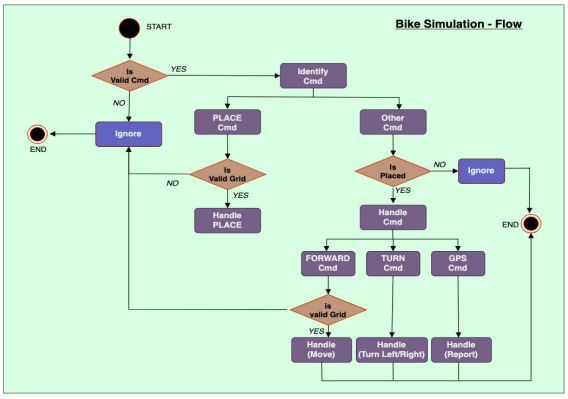
Solution design and approach - Bike Simulator

Problem selection

As part of this coding challenge, the first decision point was selecting the problem. My primary criteria for selection was, how suitable is the problem to demonstrate my knowledge and understanding of OOP concepts which will align with the Java programming language. The Bike Simulation option was chosen because, given the nature of the problem it made sense to represent the bike and the grid in terms of objects.

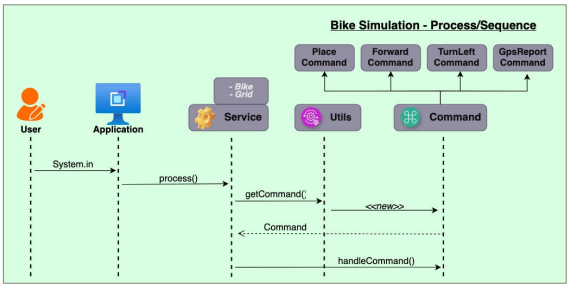
Design and approach

The approach to solve this problem was following the systematic process I would follow when designing a solution.



The first step was to document, in a diagrammatic form, the example scenarios provided to run the program with expected outcomes. This gave me a good understanding about how to model the objects, what attributes to assign and what operations are required for the objects to behave as per the given rules.

Following this I completed the sequence diagram to represent the sequential flow of the program. I used some common design patterns (Facade, Service layer, Command handler etc…) to formulate the sequence.



To maintain the separation of concerns, I structured the solution into multiple layers.

- Application layer - Handles the IO with STDIN and files
- Service layer - Responsible for orchestration, coordination and business logic
- Model layer - Encapsulate the domain models and their own behaviours
- Utility layer - Reusable utility commands and logging

I choose the above design so that it can be easily extended(for example, having an API layer in front). Each class follows the single responsibility principle, keeping concerns well separated. I also added centralised logging to maintain cleaner output and with the possibility of turning on-off debugging statements.

The code with the instructions on how to run and further documentation is made available at: **https://github.com/ruchirad/bike_simulation**

Ruchira De Alwis