**BGL cas360 Coding Challenge**

**– Kan Ranganathan**

**Bike Simulation**

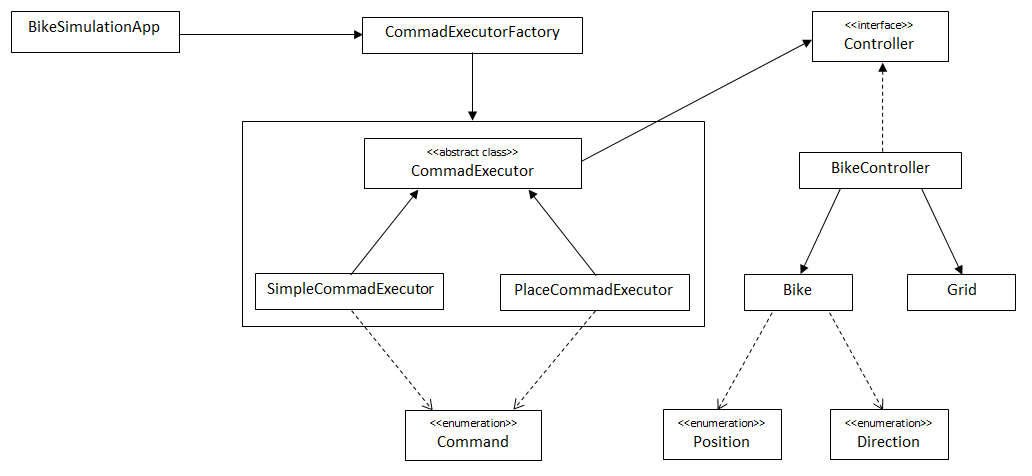
Both applications are command applications. Which accept input from STDIN and a FILE and, write output to STDOUT. Phonewords application executes only one command to convert number to word based on set of rules, which involves lots of text manipulation. Whereas, Bike Simulation application executes five different commands, which involves simple switch logic to interact with the different objects. I think Bike Simulation is better to demonstrate OOP skills and have chosen for this coding challenge.

My design is Object Oriented and approach is mixture of Top-Down and Bottom-Up approaches. In order to understand the problem and identify the main objects or entities, Top-Down approach is used. For this problem, there are three main entities:

* USER INTERFACE, which accepts command from STDIN or a FILE, and outputs to STDOUT
* COMMAND EXECUTION, which validates and executes a command
* CORE, which controls the movement of a bike on a grid

And then the Bottom-Up approach is used in designing each main entity. Starting with the CORE, objects **Bike** and **Grid** are identified and, characteristics and methods are defined. Movement of a **Bike** cannot happen on its own, another object must involve, which can be a driver or some kind of a **Controller**. This object needs more information to be defined. Then COMMAND EXECUTION part is looked into and the main object is identified, its functionality is execution of a command and therefore it is named as **CommandExecutor**, its only functionality is to execute command, there are five different commands (PLACE, FORWARD, TURN\_LEFT, TURN\_RIGHT, GPS\_REPORT), but PLACE command is different from others and includes additional parameters, so there are two different characteristics and therefore two more inherited objects (**SimpleCommandExecutor**, **PlaceCommandExecutor** ) are introduced. By defining these objects further provides information for the **Controller**. **CommandExecutor** also acts as a bridge to the **Controller**, which becomes an interface and, another object **BikeController** that implements **Controller** is introduced. USERINTERFACE is part of the main application and interacts with the different **CommandExecutor** based on the command and therefore another object **CommandExecutorFactory** is introduced. When designing or defining all the objects, encapsulation, inheritance, single responsibility, dependency inversion and extensibility together with some design patterns are taken into account. There are also some enumerations and supporting classes, some of them are in the following design diagram.

**Bike Simulation Design:**



**Solution in Java**

* **Source code and executable:** <https://github.com/kan-r/ProjectBGL>
* **Technologies used:** Java 15.0.1 2020-10-20, Eclipse 2020-09 (4.17.0), Maven 3.6.3
* **Instruction to execute the solution:**

This can be **executed** from the **bike-simulation**\**target** directory as:

**java -jar bike-simulation-0.0.1.jar**

(to accept input from STDIN)

**java -jar bike-simulation-0.0.1.jar *FileNameWithThePath***

(to accept input from a FILE)

* **Instruction to build the solution:**

This can be built from **bike-simulation** directory using “mvn package”

**Note:** please feel free to contact me if there is any issue or clarification.