Refactoring Code

- **Add brackets to single line loops.**

- **Creating new methods.**
  - In some files certain method contained too much of code so a bunch of code were extracted from the complex method to create a new method for better understanding of code as well as to reduce confusion.

    {

    }

- **Map modules centralized.**
  - Previously several class were to be changed if there was any updation in map file.
  - Now all the map modules are centralized in a Map Central Method which will occupy less space and time, as no other file would be updated rather than Map Central.

    {

    }

- **Adding brackets.**
  - In many methods the control statement which have one line of command were not having brackets.
  - This created confusion so brackets were added to reduce confusion.

- **Restructuring of code.**
    - There were many “if..else” statements, they created ambiguity in code. So some were replaced by other methods and some were replaced by “switch()..” statement.
    - Refactoring by combining them into a single conditional expression and extracting it.

- **Remove unnecessary comments.**
    - Comments were written to describe different methods or variables. Now variables are changed to appropriate “self-explanatory” name so unnecessary comments are now deleted.

- **Renaming of Necessary Variables and Methods.**
    - Adding new functionalities to the project tends to add new methods and class, because of that some methods and classes were renamed, so as to remove unambiguity and create a clear picture of each and every method.

- **Consolidate Duplicate Conditional Fragments:**
    - Previously, same fragment of code was in all branches of a conditional expression, so by refactoring them by moving it outside of the expression and adding the fragment after all the conditions were completed.