SOEN 6441

ADVANCED PROGRAMMING PRACTICES

PROJECT: RISK DOMINATION GAME

CODING STANDARDS

TEAM 11

TEAM MEMBER	STUDENT ID
Zankhanaben Patel	40067635
Koshaben Patel	40094385
Piyush Thummar	40125029
Raj Mistry	40119206
Jaswanth Banavathu	40080737

SUBMITTED TO: JOEY PAQUET

CODING CONVENTIONS

Following coding conventions are set of rules enforced throughout lifecycle of software development.

Code Layout

1. **Indentation**:
    - Four spaces of tab key is applied to any file of project
    - Method of the function body and functions are indented with tab. Statements such as **if, switch, while, for, do while** and its body is indented.

2. **Blank Lines**:
    - One blank line is used in the following circumstances:

    Between package and import statements
    Between class declarations and data members in a file.
    Between methods of class.

3. **Blank Spaces**:
    - Blank spaces are used between keywords, variable assignments, object creation in order to enhance redability.

    Example:
    ```
    Switch (condition) {
        //logic of code
    }
    ```

    Example:
    ```
    Int APP = 500;
    ```

    Example:
    ```
    MyClass myclass = new MyClass();
    ```

4. **File Organization**: Structure of a files of project is described below:
    Files are saved inrespective packages.

    Main packages of the project are:
    Model, View, Controller, Config, Constant, Dto, Logging, Service, ServiceImpl, Strategy, StrategyImpl, ObserverPattern, Builder, Adapter.

.

Naming conventions

- **Classes**:

Class names chosen for a file is a noun. First letter of a class is capitalized and consecutive letter of the name of the class is also capitalized. Class name should be meaningful, descriptive, precise. Abbreviation or acronyms of the words are avoided.

```
@Controller
public class WelcomeController implements Initializable {

    @FXML
    private Button btnPlayGame;
```

- **Methods**:
  Camel case notation is used in writing method names in class.

```
    private String attackPhase(String command) {

        if (!attackMove || command.contains("attackmove")) {

            String[] dataArray = command.split(" ");
            List<String> commandData = Arrays.asList(dataArray);
```

- **Attributes**:
  Attributes of concept class start is written using Camel case notation.

```
public class RiskCard {

    /**
     * This data member indicate card number
     */
    private int cardNumber;
```

- **Variables:**
  Variable names should be descriptive, precise and meaningful. Variable name should be mnemonic. Variable name of one character is avoided and chosen only for temporary variables. Temporary variable names are i, j, k, m, n, o, p for integer type variables. For iterating loop temporary variable named itr, x or any abbreviation of attribute can be used and is up to developer.

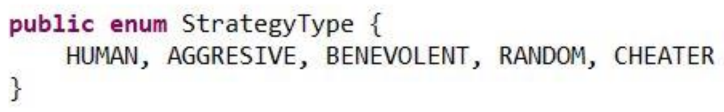
  Example:
  Public static final String NEUTRAL = “NEUTRAL”;

```
@Service
public class MapManagementImpl implements MapManagementInterface {

    public static final String NEW_LINE = "line.separator";
    public static final String MAP_DIR_PATH = "src/main/resources/maps/";

    public static final String NAME = "name";
    public static final String FILES = "[files]";
```

- **Constants**:
  Constants or unchangeable variables cannot be altered throughout the program. These variables are declared in capitals. Enum, a special class which is frequently used for representing or creating a group of constants.
  To create a enum, the keyword enum should be used and all the constants are declared in uppercase letters separated by commas.



```
public enum StrategyType {
    HUMAN, AGGRESIVE, BENEVOLENT, RANDOM, CHEATER
}
```

Comment

Comments are integrated in order to enhance readability and understandability. JavaDoc comments are integrated thoroughly for every classes and methods.

- **Documentation Comments:**
  Documentation comment is done as per JavaDoc conventions.
  Following set of annotations are to be incorporated. However, all files does not contain every listed annotations:

  @author: Name of the author who wrote piece of code, there can be atmost two authors.
  @param: Param annotation is used to define parameters of the methods and its description.
  @return: Return annotation is used to define return type of a method and its description.
  @throws: Throws annotation is used to define the type of exception.
  @see: See annotation is used to enhance understandability.
  @inheritDoc: This annotation inherits from "nearest" class.
  @version: This annotation specifies the current working software version.

References

1. https://www.oracle.com/technetwork/java/javase/documentation/index-137868.html

2. Joey Paquet: API documentation generation tools: Javadoc

3. Joey Paquet: coding conventions