Template Method Design Pattern:

Title Natural language description of the implemented Template Method Design Pattern.

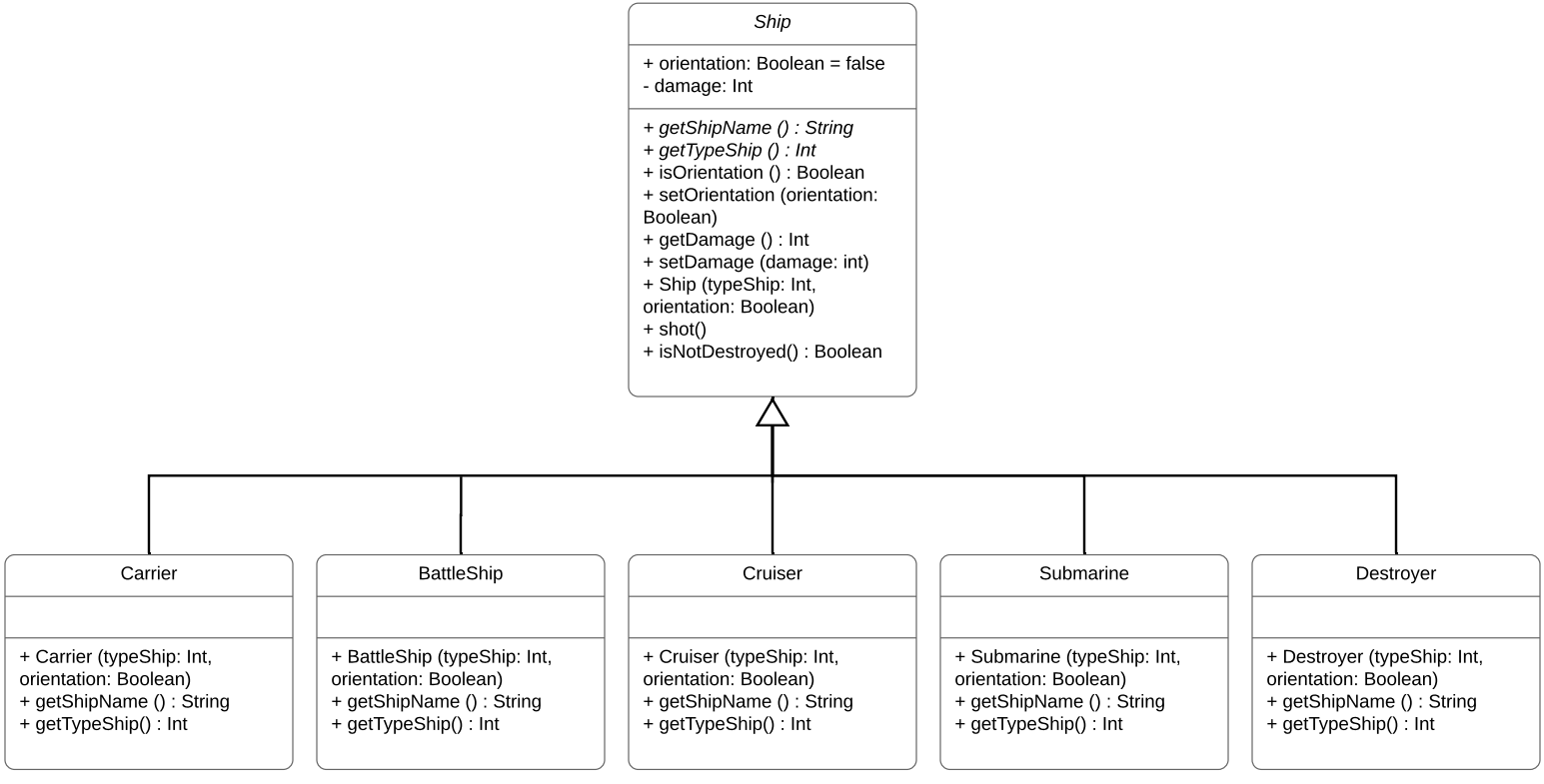
Author/Date SEM_GROUP_79 / 08-01-2020

Modification/Date no modification / 10-01-2020

Purpose Adding more ships for future releases will be easier.

Overview There is an abstract Ship class which are extended by the specific ship types such as Carrier, BattleShip, Cruiser, Submarine, Destroyer & Mini.

Cross References The game shall offer five ships with specified size and orientation upon starting a new game.



Factory Method Design Pattern:

Title Natural language description of the implemented Factoriy Method Design Pattern.

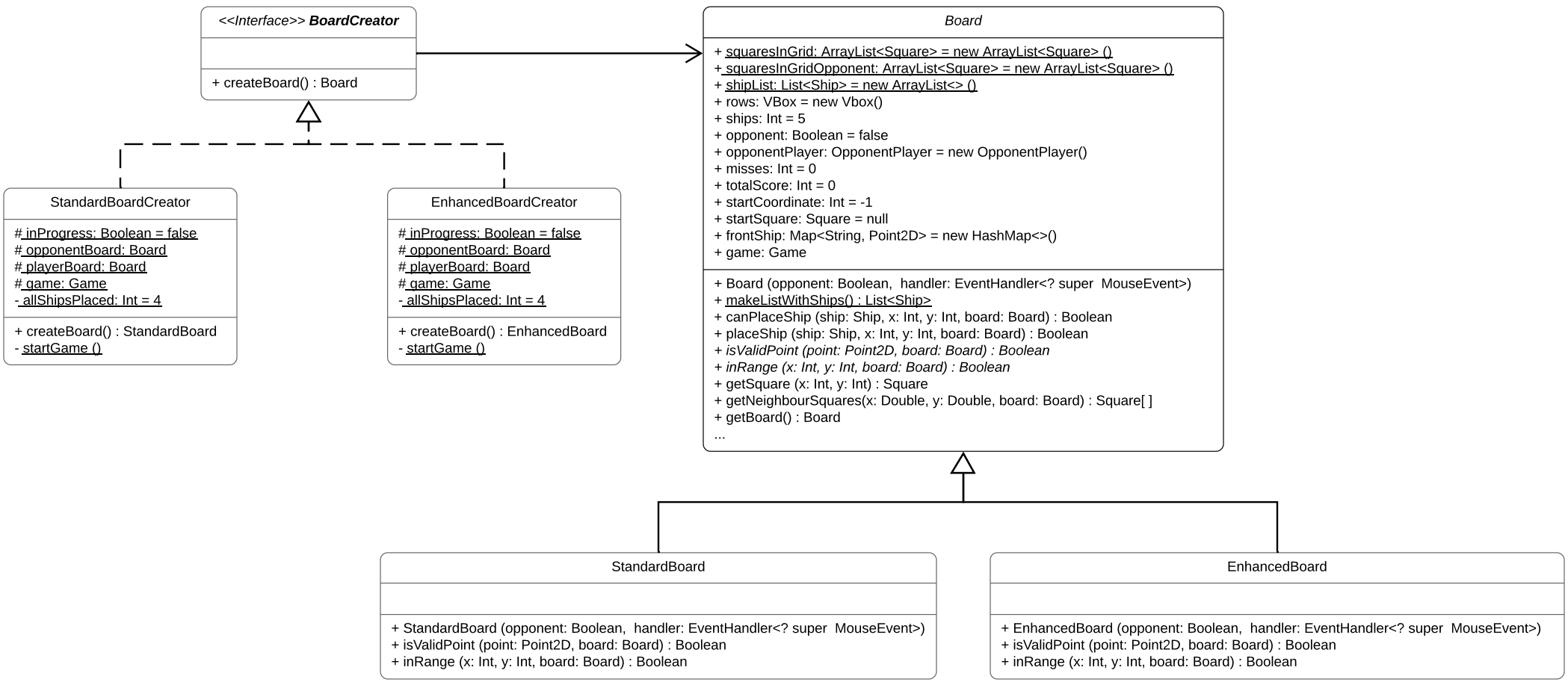
Author/Date SEM_GROUP_79 / 08-01-2020

Modification/Date Added an interface BoardCreator / 15-01-2020

Purpose Displaying different type of boards. For future releases adding more levels will be easier.

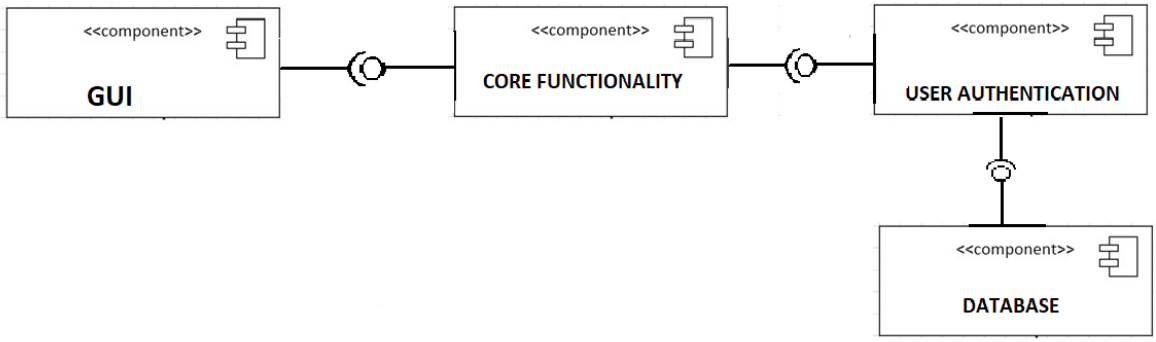
Overview There is an interface BoardCreator which is implemented by the two type of boards, resulting in StandardBoardCreatore and EnhancedBoardCreator. The creators create there own type of board upon starting a new game. The Board class is an abstract class extended by the StandardBoard and EnhancedBoard. In the Board classes the features of each board are eleborated

Cross References The game shall offer different game modes defined by the boardshape.



Naval Battle

Software Architecture



Graphical User Interface

Description: This component comprises of FXML, controllers and a Main. The main is the starting point of our entire game. It shows the login/register screen from where the user can enter their details and start playing. The FXML files hold the actual look/blueprint of our game windows and all have corresponding controllers which contain methods which are executed when a button on that screen is pressed. After the Login/register screen the home-screen is displayed where the user can start a game, watch a tutorial on how to play, see their progress/high-scores etc.

Role/Responsibility: The GUI is the main point of interaction of the user with our program. The user enters credentials while logging in/registering, places ships, shoots the enemy board, views information like high score etc, from the GUI

Motivation: We decided to use JavaFX and Scene-Builder because we used them previously in the OOP Project, and because Scene Builder provides a screen outlook of what the window actually looks like and all the implementation can be done that way, furthermore the JavaFX and Scene-Builder go hand in hand so we decided to used JavaFX as well.

Core functionality

Description: The functionality of the actual game. The core functionality contains the methods and classes to actually shoot on the enemy board, the enemy to shoot on you, the method to create and place ships etc. This component contains classes that are strongly interwoven with the GUI and the database. Most of the core functionality is coded by ourselves, except for the parts that interweave with the GUI and the database.

Function: The function is to dictate how the game actually works, from placing ships to scoring hits on the enemy to finishing the game and store the high score.

Motivation: For making the core functionality we coded a lot ourselves, because there was no specific library available that fitted our desires. For the parts the of the core functionality that interweave with the GUI and the database, we used the libraries that were used for the GUI and the database to avoid inconsistency issues between the core functionality of the gameplay and the GUI or database.

User Authentication

Description: Every time a user starts the game and enters their log-in details, a new User object is created with those credentials, and the database verifies if the username exists, if not the user must register, if it does then the password is obtained from the database and checked against the password provided, if they match then authentication is successful and if not the user must try again.

Role/Responsibility: Checks that only registered users can play the game and every new user must first register, also it make sure that all usernames assigned are unique, so there is not possibility of conflict, if the player base grows.

Motivation: For establishing a connection we used the Driver Manager class, Prepared Statements(requirement) and result Sets which contain the data retrieved from the database, because the data is returned from the database as a tuple.

Database

Description: The database used for a remote database located at projects.ewi.tudelft.nl. The database has 2 tables-Users(stores username and password) and Scores(stores user-id and the score of the user).We use JDBC and drivers to connect to the database. The database is used during user authentication, and while storing new scores to the database. The connection is made using the Connect class.

Role/Responsibility: The database stores user credentials, the user progress, scores for each and every match etc. Without the database the user cannot even login/register and therefore can not even start a match.

Motivation: Given that the theme/motto of our game is “keep it simple”, we decided to use JDBC instead of the Spring, as we believe it keeps the design easier to implement and maintain.