EECS3311-Winter2019-Section: M – Project Report

Submitted electronically by:

Team members	**Name**	**Prism Login**	**Signature**
Member 1:	*Jiahao Kong*	*ansonkg*	Kong Jiahao
Member 2:	*Tao Deng*	*dengtao*	Deng Tao
***Submitted under Prism Account:**	*dengtao*		

Contents:

1. Requirements for Project "Battleship"

The battleship project basically introduces a ship-related battleship game. This game is designed with four difficulties: easy, medium, hard and advanced.

- For easy mode, the game board is 4x4 with total number of 2 ships (different size, hit bigger size can get more scores), and the player require to have 8 chances of fire operations to shoot the ship, and 2 chances of bomb operations.
- For medium mode, the game board is 6x6 with total number of 3 ships (different size, hit bigger size can get more scores), and the player require to have 16 chances of fire operations to shoot the ship, and 3 chances of bomb operations.
- For hard mode, the game board is 8x8 with total number of 5 ships (different size, hit bigger size can get more scores), and the player require to have 24 chances of fire operations to shoot the ship, and 5 chances of bomb operations.
- For hard mode, the game board is 12x12 with total number of 7 ships (different size, hit bigger size can get more scores), and the player require to have 40 chances of fire operations to shoot the ship, and 7 chances of bomb operations.

The fire operation is required to hit a specific area in the game board and the bomb operation is required to hit two vertical or horizontal adjacent areas.

Players are required to hit all ships within the fixed chances of fire and bomb operations to win the game, otherwise they will lose. And players also can play a new game when they finish playing the last one whatever they win or lose, the sink ship will count as the score and the score will be recorded until they do not want to play. However, when the players start to play a game, they cannot play new game mode until they finish the old game mode or give up.

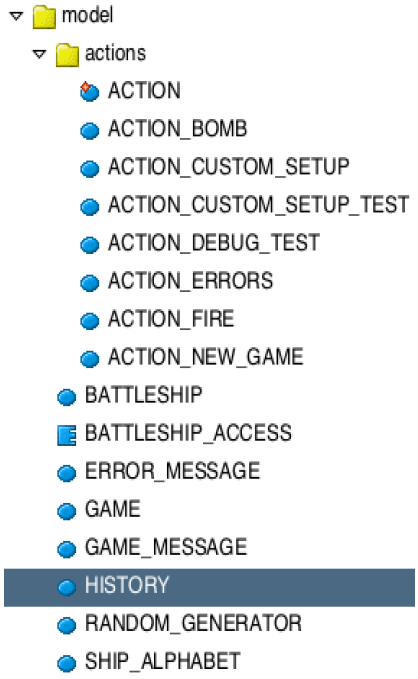
In addition, players also have unlimited times of redo or undo operations for the last step of operation they give. Undo operation is required the game to remember the old state before execution. This step is for player to retract the false move in the game, but it is required to work in fire and bomb operation only, which cannot work to play different difficulties of game. So as redo operation, redo is required the game to do exactly the same execution with the last step. Redo the same operations of fire and bomb might cause the game output error messages. Moreover, players also have an give up operation to stop play the game.

There is a debug test operation as well to show each ship's position in the game board, the goal is testing hit the ships or not.

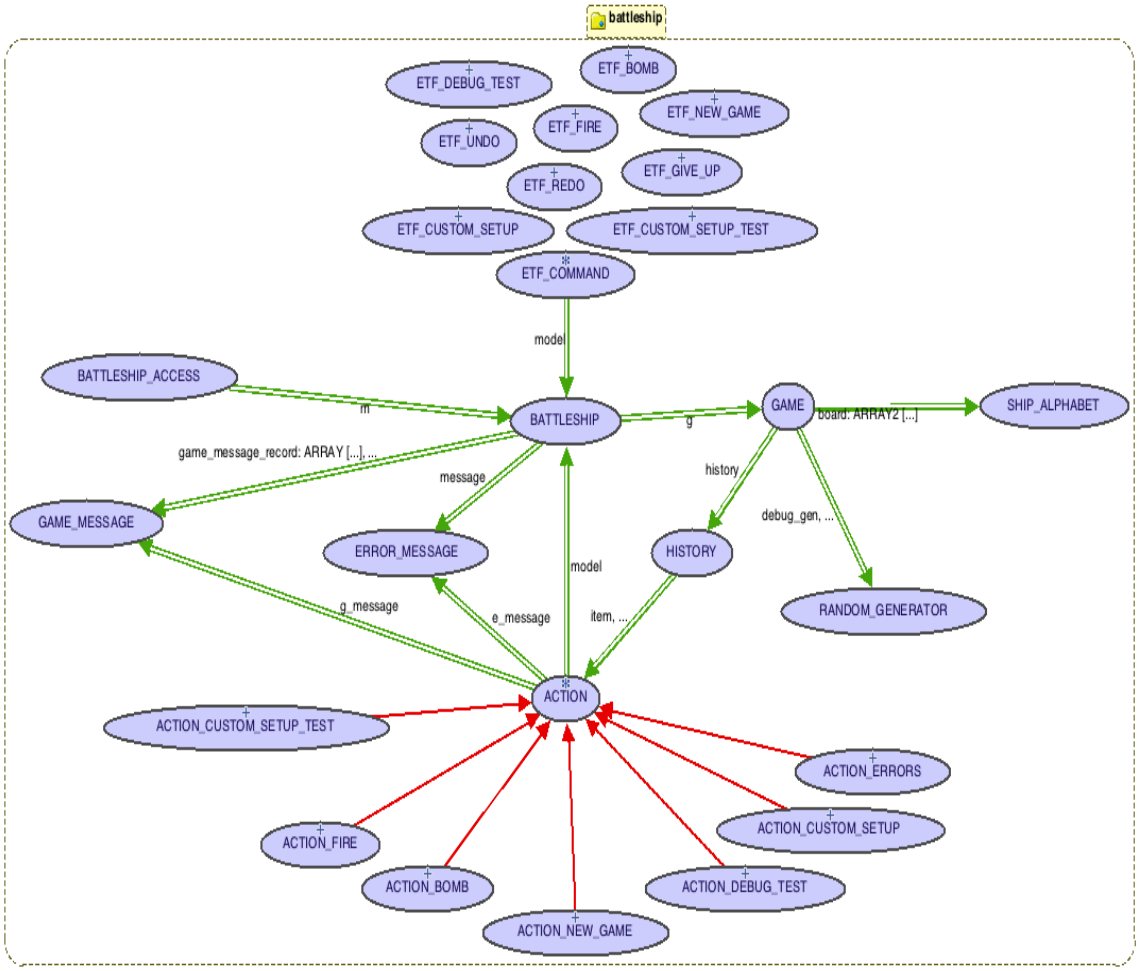
And the players also can design their own game, which design the size of the game board, the total number of ships, the total chances of fire and the total changes of bomb. But they require to make sure the size of game board cannot smaller than 4x4 or bigger than 12x12, and the total number of ships cannot smaller than 1 or bigger than 7, the total chances of fire cannot less than 1 or more than 144, the total chances of bomb cannot less than 1 or more than 7. For check what they design there is another operation called customer debug test, which check the correctness of the players' design.

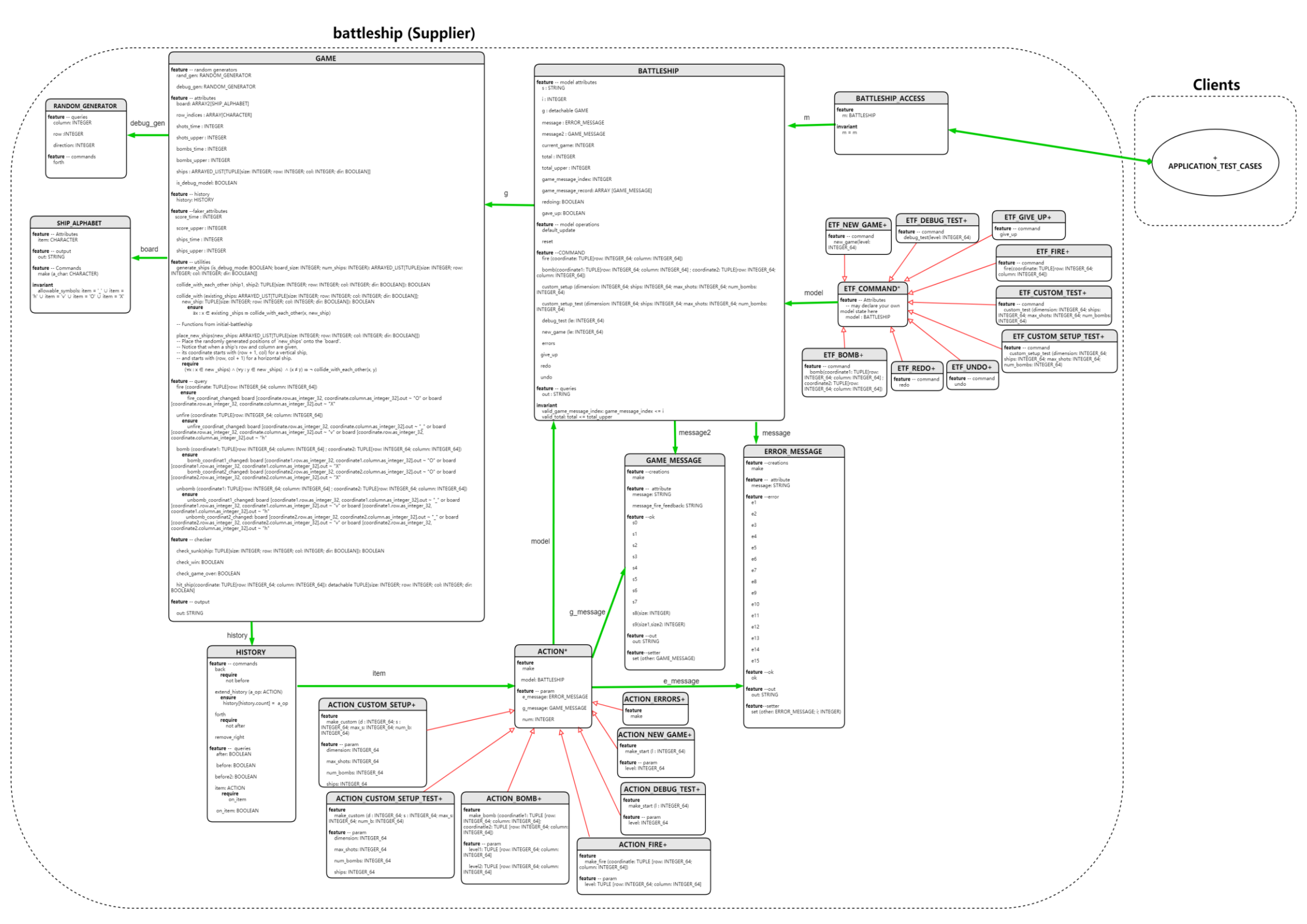
2. BON class diagram overview (architecture of the design)

The file structure in the cluster model is as follows



The main BON diagram is below




battleship (Supplier)
GAME
feature -- random generators
rand_gen: RANDOM_GENERATOR
debug_gen: RANDOM_GENERATOR
feature -- attributes
board: ARRAY2[SHIP_ALPHABET]
row_indices : ARRAY[CHARACTER]
shots_time : INTEGER
shots_upper : INTEGER
bombs_time : INTEGER
bombs_upper : INTEGER
ships : ARRAYED_LIST[TUPLE[size: INTEGER; row: INTEGER; col: INTEGER; dir: BOOLEAN]]
is_debug_model: BOOLEAN
feature -- history
history: HISTORY
feature --faker_attributes
score_time : INTEGER
score_upper : INTEGER
ships_time : INTEGER
ships_upper : INTEGER
feature -- utilities
generate_ships (is_debug_mode: BOOLEAN; board_size: INTEGER; num_ships: INTEGER): ARRAYED_LIST[TUPLE[size: INTEGER; row: INTEGER; col: INTEGER; dir: BOOLEAN]]
collide_with_each_other (ship1, ship2: TUPLE[size: INTEGER; row: INTEGER; col: INTEGER; dir: BOOLEAN]): BOOLEAN
collide_with (existing_ships: ARRAYED_LIST[TUPLE[size: INTEGER; row: INTEGER; col: INTEGER; dir: BOOLEAN]]; new_ship: TUPLE[size: INTEGER; row: INTEGER; col: INTEGER; dir: BOOLEAN]): BOOLEAN
ensure
∃x : x ∈ existing_ships ⇒ collide_with_each_other(x, new_ship)
-- Functions from initial-battleship
place_new_ships(new_ships: ARRAYED_LIST[TUPLE[size: INTEGER; row: INTEGER; col: INTEGER; dir: BOOLEAN]])
-- Place the randomly generated positions of `new_ships' onto the `board'.
-- Notice that when a ship's row and column are given,
-- its coordinate starts with (row + 1, col) for a vertical ship,
-- and starts with (row, col + 1) for a horizontal ship.
require
(∀x : x ∈ new_ships) ∧ (∀y : y ∈ new_ships) ∧ (x ≠ y) ⇒ ¬ collide_with_each_other(x, y)
feature -- query
fire (coordinate: TUPLE[row: INTEGER_64; column: INTEGER_64])
ensure
fire_coordinat_changed: board [coordinate.row.as_integer_32, coordinate.column.as_integer_32].out ~ "O" or board [coordinate.row.as_integer_32, coordinate.column.as_integer_32].out ~ "X"
unfire (coordinate: TUPLE[row: INTEGER_64; column: INTEGER_64])
ensure
unfire_coordinat_changed: board [coordinate.row.as_integer_32, coordinate.column.as_integer_32].out ~ "_" or board [coordinate.row.as_integer_32, coordinate.column.as_integer_32].out ~ "v" or board [coordinate.row.as_integer_32, coordinate.column.as_integer_32].out ~ "h"
bomb (coordinate1: TUPLE[row: INTEGER_64; column: INTEGER_64] ; coordinate2: TUPLE[row: INTEGER_64; column: INTEGER_64])
ensure
bomb_coordinat1_changed: board [coordinate1.row.as_integer_32, coordinate1.column.as_integer_32].out ~ "O" or board [coordinate1.row.as_integer_32, coordinate1.column.as_integer_32].out ~ "X"
bomb_coordinat2_changed: board [coordinate2.row.as_integer_32, coordinate2.column.as_integer_32].out ~ "O" or board [coordinate2.row.as_integer_32, coordinate2.column.as_integer_32].out ~ "X"
unbomb (coordinate1: TUPLE[row: INTEGER_64; column: INTEGER_64] ; coordinate2: TUPLE[row: INTEGER_64; column: INTEGER_64])
ensure
unbomb_coordinat1_changed: board [coordinate1.row.as_integer_32, coordinate1.column.as_integer_32].out ~ "_" or board [coordinate1.row.as_integer_32, coordinate1.column.as_integer_32].out ~ "v" or board [coordinate1.row.as_integer_32, coordinate1.column.as_integer_32].out ~ "h"
unbomb_coordinat2_changed: board [coordinate2.row.as_integer_32, coordinate2.column.as_integer_32].out ~ "_" or board [coordinate2.row.as_integer_32, coordinate2.column.as_integer_32].out ~ "v" or board [coordinate2.row.as_integer_32, coordinate2.column.as_integer_32].out ~ "h"
feature -- checker
check_sunk(ship: TUPLE[size: INTEGER; row: INTEGER; col: INTEGER; dir: BOOLEAN]): BOOLEAN
check_win: BOOLEAN
check_game_over: BOOLEAN
hit_ship(coordinate: TUPLE[row: INTEGER_64; column: INTEGER_64]): detachable TUPLE[size: INTEGER; row: INTEGER; col: INTEGER; dir: BOOLEAN]
feature -- output
out: STRING
RANDOM_GENERATOR
feature -- queries
column: INTEGER
row :INTEGER
direction: INTEGER
feature -- commands
forth
debug_gen
SHIP_ALPHABET
feature -- Attributes
item: CHARACTER
feature -- output
out: STRING
feature -- Commands
make (a_char: CHARACTER)
invariant
allowable_symbols: item = '_' ∪ item = 'h' ∪ item = 'v' ∪ item = 'O' ∪ item = 'X'
board
BATTLESHIP
feature -- model attributes
s : STRING
i : INTEGER
g : detachable GAME
message : ERROR_MESSAGE
message2 : GAME_MESSAGE
current_game: INTEGER
total : INTEGER
total_upper : INTEGER
game_message_index: INTEGER
game_message_record: ARRAY [GAME_MESSAGE]
redoing: BOOLEAN
gave_up: BOOLEAN
feature -- model operations
default_update
reset
feature --COMMAND
fire (coordinate: TUPLE[row: INTEGER_64; column: INTEGER_64])
bomb(coordinate1: TUPLE[row: INTEGER_64; column: INTEGER_64] ; coordinate2: TUPLE[row: INTEGER_64; column: INTEGER_64])
custom_setup (dimension: INTEGER_64; ships: INTEGER_64; max_shots: INTEGER_64; num_bombs: INTEGER_64)
custom_setup_test (dimension: INTEGER_64; ships: INTEGER_64; max_shots: INTEGER_64; num_bombs: INTEGER_64)
debug_test (le: INTEGER_64)
new_game (le: INTEGER_64)
errors
give_up
redo
undo
feature -- queries
out : STRING
invariant
valid_game_message_index: game_message_index <= i
valid_total: total <= total_upper
g
BATTLESHIP_ACCESS
feature
m: BATTLESHIP
invariant
m = m
m
Clients
APPLICATION_TEST_CASES
ETF_NEW_GAME+
feature -- command
new_game(level: INTEGER_64)
ETF_DEBUG_TEST+
feature -- command
debug_test(level: INTEGER_64)
ETF_GIVE_UP+
feature -- command
give_up
ETF_FIRE+
feature -- command
fire(coordinate: TUPLE[row: INTEGER_64; column: INTEGER_64])
ETF_CUSTOM_TEST+
feature -- command
custom_test (dimension: INTEGER_64; ships: INTEGER_64; max_shots: INTEGER_64; num_bombs: INTEGER_64)
ETF_CUSTOM_SETUP_TEST+
feature -- command
custom_setup_test (dimension: INTEGER_64; ships: INTEGER_64; max_shots: INTEGER_64; num_bombs: INTEGER_64)
model
ETF_COMMAND*
feature -- Attributes
-- may declare your own model state here
model : BATTLESHIP
ETF_BOMB+
feature -- command
bomb(coordinate1: TUPLE[row: INTEGER_64; column: INTEGER_64] ; coordinate2: TUPLE[row: INTEGER_64; column: INTEGER_64])
ETF_REDO+
feature -- command
redo
ETF_UNDO+
feature -- command
undo
message2
message
GAME_MESSAGE
feature --creations
make
feature -- attribute
message: STRING
message_fire_feedback: STRING
feature --ok
s0
s1
s2
s3
s4
s5
s6
s7
s8(size: INTEGER)
s9(size1,size2: INTEGER)
feature --out
out: STRING
feature--setter
set (other: GAME_MESSAGE)
ERROR_MESSAGE
feature --creations
make
feature -- attribute
message: STRING
feature --error
e1
e2
e3
e4
e5
e6
e7
e8
e9
e10
e11
e12
e13
e14
e15
feature --ok
ok
feature --out
out: STRING
feature--setter
set (other: ERROR_MESSAGE; i: INTEGER)
model
g_message
history
HISTORY
feature -- commands
back
require
not before
extend_history (a_op: ACTION)
ensure
history[history.count] = a_op
forth
require
not after
remove_right
feature -- queries
after: BOOLEAN
before: BOOLEAN
before2: BOOLEAN
item: ACTION
require
on_item
on_item: BOOLEAN
item
ACTION*
feature
make
model: BATTLESHIP
feature -- param
e_message: ERROR_MESSAGE
g_message: GAME_MESSAGE
num: INTEGER
e_message
ACTION_CUSTOM_SETUP+
feature
make_custom (d : INTEGER_64; s : INTEGER_64; max_s: INTEGER_64; num_b: INTEGER_64)
feature -- param
dimension: INTEGER_64
max_shots: INTEGER_64
num_bombs: INTEGER_64
ships: INTEGER_64
ACTION_ERRORS+
feature
make
ACTION_NEW_GAME+
feature
make_start (l : INTEGER_64)
feature -- param
level: INTEGER_64
ACTION_DEBUG_TEST+
feature
make_start (l : INTEGER_64)
feature -- param
level: INTEGER_64
ACTION_CUSTOM_SETUP_TEST+
feature
make_custom (d : INTEGER_64; s : INTEGER_64; max_s: INTEGER_64; num_b: INTEGER_64)
feature -- param
dimension: INTEGER_64
max_shots: INTEGER_64
num_bombs: INTEGER_64
ships: INTEGER_64
ACTION_BOMB+
feature
make_bomb (coordinate1: TUPLE [row: INTEGER_64; column: INTEGER_64]; coordinate2: TUPLE [row: INTEGER_64; column: INTEGER_64])
feature -- param
level1: TUPLE [row: INTEGER_64; column: INTEGER_64]
level2: TUPLE [row: INTEGER_64; column: INTEGER_64]
ACTION_FIRE+
feature
make_fire (coordinate: TUPLE [row: INTEGER_64; column: INTEGER_64])
feature -- param
level: TUPLE [row: INTEGER_64; column: INTEGER_64]

In this project, our main purpose is creating a redo and undo pattern and this is the biggest difference between lab4 and project. For creating redo and undo pattern, we created the ACTION deferred class. We used Information Hiding principle, Dynamic Binding and Polymorphism.

Firstly, we created ACTION class as parent of other action classes, so we can use Dynamic Binding. We defined the history feature of GAME class can only include ACTION class and its descendent classes, so we can store any action which has its own features. For example, ACTION_BOMB can store level1 and level2 as its features, but also ACTION_CUSTOM_SETUO can store dimension, ships, max_shots and num_bombs as its features. Because dynamic type is changeable at runtime, instances of descendent classes of ACTION can be created and have their own features.

Secondly, we will talk about Polymorphism in our project. Because the define of Polymorphism is an object variable may have “multiple possible shapes” and we have mentioned the descendent classes of ACTION class, each descendent class can define how to implement undo, execute and redo routine functions. This is the reason why we can have different actions when user invoke undo and redo routine functions. Moreover, when user want to add new comment to our project, it is easy and will not interrupt other existing commands.

Finally, descendent classes of ACTION class inherit the undo, execute and redo routine functions from ACTION class. Although these functions’ can be invoked by users, users can see these functions’ detail and their structures. Hence, we also use and follow information hiding principle.

The design for BATTLESHIP and BATTLESHIP_ACCESS is singleton design pattern, we can make sure users can only declare BATTLESHIP once and the instance of BATTLESHIP is unique.

3. Table of modules — responsibilities and information hiding

In actions entity

1.	ACTION	**Responsibility:** Represent a branch of operations which perform in BATTLESHIP.	**Alternative :** None
	Abstract	**Secret:** Use inheritance to define some operations ("undo/redo/execute") to have specific features for all its descendants to use. And only BATTLESHIP can call these operations.	

1.1.	ACTION_BOMB	**Responsibility:** Represent an operation for bombing adjacent areas in the game board.	**Alternative :** None
	Concrete	**Secret:** Define specific "undo/redo/execute" operations for only BATTLESHIP can call.	

1.2.	ACTION_CUSTOM_SETUP	**Responsibility:** Represent an operation for setting up the game with specific game board size, number of ship and number of firing/bombing chances.	**Alternative:** None
	Concrete	**Secret:** Define specific "undo/redo/execute" operations for only BATTLESHIP can call.	

1.3.	ACTION_CUSTOM_SETUP_TEST	**Responsibility:** Represent an operation for testing the set up for the game with specific game board size, number of ship and number of firing/bombing chances.	**Alternative :** None
	Concrete	**Secret:** Define specific "undo/redo/execute" operations for only BATTLESHIP can call.	

1.4.	ACTION_DEBUG_TEST	**Responsibility:** Represent an operation for testing the the game, which can show the game board size, number of ship and number of firing/bombing chances.	**Alternative :** None
	Concrete	**Secret:** Define specific "undo/redo/execute" operations for only BATTLESHIP can call.	

1.5.	ACTION_ERRORS	**Responsibility:** Represent an operation for outputting possible errors messages.	**Alternative :** None
	Concrete	**Secret:** Define specific "undo/redo/execute" operations for only BATTLESHIP can call.	

1.6.	ACTION_FIRE	**Responsibility:** Represent an operation for shooting specific area in the game board.	**Alternative:** None
	Concrete	**Secret:** Define specific "undo/redo/execute" operations for only BATTLESHIP can call.	

1.7.	ACTION_NEW_GAME	**Responsibility:** Represent an operation for creating new game mode.	**Alternative:** None
	Concrete	**Secret:** Define specific "undo/redo/execute" operations for only BATTLESHIP can call.	

In model entity

1.	BATTLESHIP_ACCESS	**Responsibility:** Provide one instance of BATTLESHIP in the user interface.	**Alternative:** None
	Concrete (Expanded)	**Secret:** None	

1.1.	BATTLESHIP	**Responsibility:** Representation of how BATTLESHIP works.	**Alternative:** None
	Concrete	**Secret:** **1.** "not_start : BOOLEAN", "no_shooting_remaining : BOOLEAN", "invalid_coordinate(coordinate: TUPLE[row: INTEGER_64; column: INTEGER_64]) : BOOLEAN", "already_fired(coordinate: TUPLE[row: INTEGER_64; column: INTEGER_64]) : BOOLEAN", "no_bombs_remaining : BOOLEAN" check the basic game design for ETF_COMMAND. **2.** "play_game(le : INTEGER_64; is_debug_mode : BOOLEAN)", "play_custom_game(is_debug_mode : BOOLEAN; dimension: INTEGER_64; ships : INTEGER_64; max_shots : INTEGER_64; num_bombs : INTEGER_64)", "message_set_after_fire(game : GAME; game_message : GAME_MESSAGE)", "set_game_message_before (game_message : GAME_MESSAGE; num : INTEGER)", "set_error_message_before (error_message : ERROR_MESSAGE; e_m : ERROR_MESSAGE; num : INTEGER)" for helping ACTION to set up.	

2.	GAME	**Responsibility:** Representation of specific operations in BATTLESHIP works.	**Alternative:** None
	Concrete	**Secret:** None	

3.	GAME_MESSAGE	**Responsibility:** Represent the outputting game messages when game works normal.	**Alternative :** None
	Concrete	**Secret:** None	

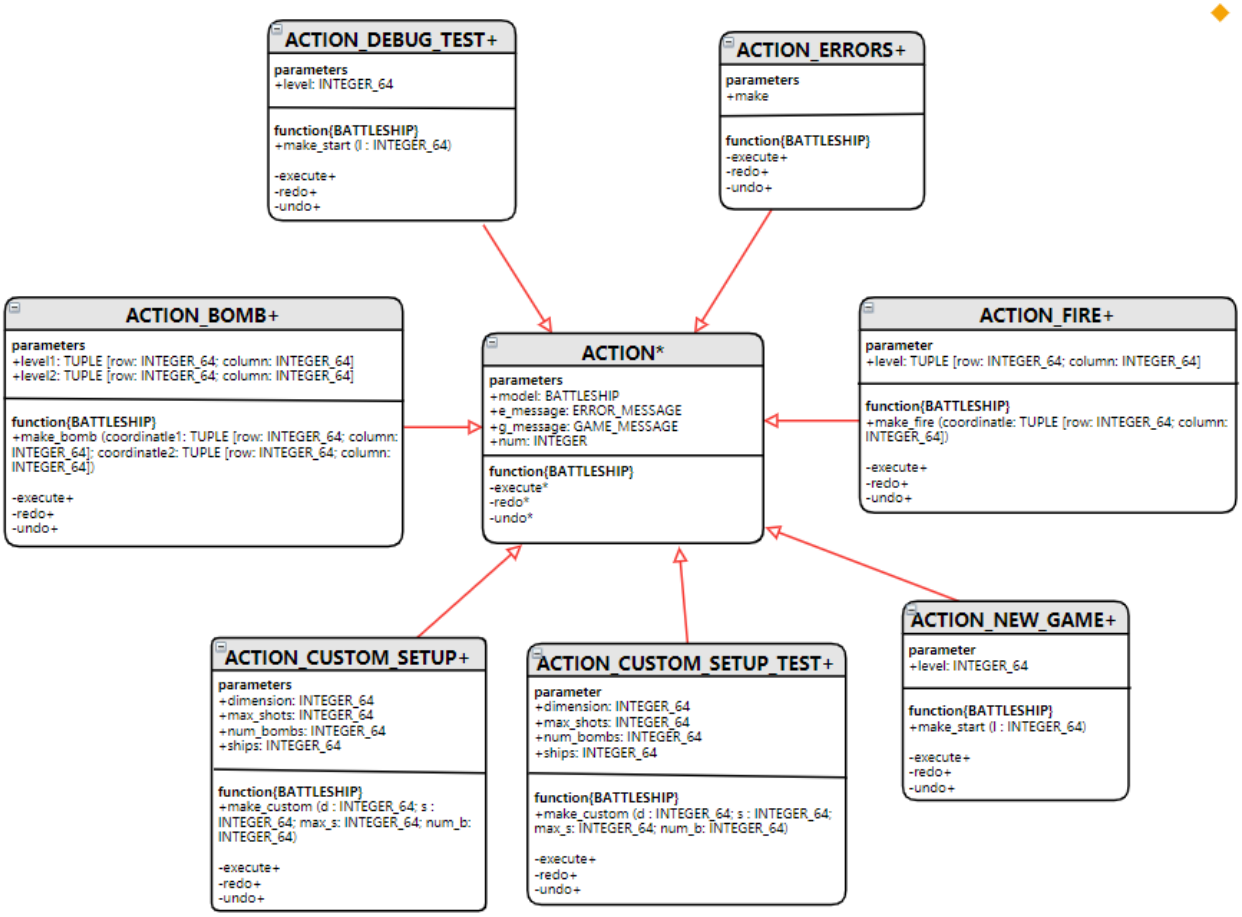
4.	ERROR_MESSAGE	**Responsibility:** Represent the outputting error messages when game has some error occuring.	**Alternative :** None
	Concrete	**Secret:** None	

5.	HISTORY	**Responsibility:** Represent a list stores all actions had played in the game. Its implementation is ARRAYED_LIST [G]	**Alternative:** LINKED_LIST[G]
	Concrete	**Secret:** None	

6.	RAMDON_GENERATOR	**Responsibility:** Represent "new_game" operation and "debug_test" base on different creations which generate randomly.	**Alternative :** None
	Concrete	**Secret:** None	

7.	SHIP_ALPHABET	**Responsibility:** Represent different symbols in the game board after different operations.	**Alternative:** None
	Concrete	**Secret:** None	

4. Expanded description of design decisions



We elect the most important module in our project by using the BON diagram which provided in Part 2. We find that ACTION is the most important module in our design.

Our considerations are based on different aspects. At first, our design of user interface has a strong relation with redo/undo design patterns. Redoing step exactly do the previous operation/command again, so the redo pattern should have all operations' execution method for supporting. And undoing step need to remember the old states before execution happened. And execution is also a very important part in redo/undo design pattern, executing plays the main role to do the previous operation/command again, that is why redoing step is the re-executing step, once your execution is correct, you redo should also be correct. So the redo/undo patterns basically need all the operations design information. ACTION is the central point of our redo/undo design pattern. There are different operations in our project, so for each single operation, they should different redo and undo function and output different game message as well. So ACTION is like a parent class which should be abstract, and let each operation (ACTION_BOMB, ACTION_CUSTOM_SETUP, ACTION_CUSTOM_SETUP_TEST, ACTION_DEBUG_TEST, ACTION_ERRORS, ACTION_FIRE, ACTION_NEW_GAME) should be as its children (descendants) and has their own redo, undo and execution to do.

The second reason is its basic design. Based on our official BON diagram, the significance of ACTION is connecting three major classes BATTLESHIP, GAME and HISTORY. The parameter "model" in ACTION is pointer to BATTLESHIP,

which let the redo/undo design patterns work well with all operations in the game. HISTORY is an important design pattern in redo/undo pattern as well, its job is record the states, in order to let redo and undo to do the correct executions. And the parameter “history” in GAME is pointer to HISTORY and record the states, and parameter “item” in HISTORY is pointer to ACTION, which help the redo/undo to remember the correct states.

5. Significant Contracts (Correctness)

In our design, most of significant contracts are executed in descendent classes of ETF_COMMAND class. When the input command is invalid, the error messages will be prompted out in our system. Therefore, other significant contracts will be implemented by precondition, postcondition and class invariant in other classes.

GAME
preconditions: ● **require** new_ship are placed in random positions in the board postconditions: ● **ensure** make sure new_ship is collided with the existing ship in the board ● **ensure** fire_coordinate_changed: make sure that if hit the ship marked "X" in the board, otherwise marked "O" ● **ensure** unfire_coordinate_changed: make sure the states in board, marked "_" if there is no ship in the position, marked "h" if the ship located in this position is placed horizontally, marked "v" if the ship located in this position is placed vertically. ● **ensure** bomb_coordinate1_changed: make sure that if hit the ship marked "X" in the board, otherwise marked "O" ● **ensure** bomb_coordinate2_changed: make sure that if hit the ship marked "X" in the board, otherwise marked "O" ● **ensure** unbomb_coordinate1_changed: make sure the states in board, marked "_" if there is no ship in the position, marked "h" if the ship located in this position is placed horizontally, marked "v" if the ship located in this position is placed vertically. ● **ensure** unbomb_coordinate2_changed: make sure the states in board, marked "_" if there is no ship in the position, marked "h" if the ship located in this position is placed horizontally, marked "v" if the ship located in this position is placed vertically.

GAME class, keeps states of all game features of each game, is the most important class in our design. In the fire() and bomb() routine functions, users are allowed to input valid coordination, so we made sure that the inputted coordination in the board has been changed to 'X' or 'O' by postcondition. otherwise, the postconditions of unfire() and unbomb() routine functions are used to make sure the inputted coordination in the board has been changed to '_', 'v' or 'h'. What is more, the collide_with () and place_new_ships() routine function are checked whether the new_ship has collided to other existing ships by verified their coordinations.

HISTORY
preconditions: ● **require** on_item ● **require** not after ● **require** not before postconditions:

- **ensure** history[history.count] = a_op

The model of HISTORY class is a list, so we have to keep some features of history feature. For example, the after() and before() routine functions are used as the precondition of forth() and back() routine functions. Moreover, after we extended a new action into history feature, we will check whether the action has been inputted and as the last element in history feature. As precondition of item() routine function, we need to check the current cursor validation.

BATTLESHIP
Invariant: ● **valid_game_message_index**: game_message_index <= i ● **valid_total**: total <= total_upper

In BATTLESHIP class, we must make sure game_message_index less or equal to i. And total less or equal to total_upper.

SHIP_ALPHABET
Invariant: ● **allowable_symbols**: item = '_' or item = 'h' or item = 'v' or item = 'O' or item = 'X'

In SHIP_ALPHABET class, we must make sure the item is only '_', 'h', 'v', 'O' or 'X'.

BATTLESHIP_ACCESS
Invariant: ● m = m

In BATTLESHIP_ACCESS class, we have to follow singleton pattern, so we have to make sure BATTLESHIP model only has been declared once.

6. Summary of Testing Procedures

Test Files	Description	Passed
at001.txt (Instructor provided)	Check the debug_test game mode of easy level and medium level, use fire and bomb operations to see if the users win or not.	✓
at002.txt (Instructor provided)	Check the error messages for different operations and check if record the scores or not.	✓
at101.txt (Instructor provided)	Check combination of debug_test mode, fire, bomb, undo/redo and give_up operations.	✓
at102.txt (Instructor provided)	Check combination of debug_test mode, new_game mode, fire, bomb, undo/redo, give_up and custom_setup_test operations.	✓
at01.txt	Basic check of advanced level game model with several operations and check the format of the game output.	✓
at02.txt	Basic check of hard level game model with several operations and check the format of the game output.	✓
at03.txt	Basic check of medium level game model with several operations and check the format of the game output.	✓
at04.txt	Check it is not possible play a new game when a game has already started.	✓
at05.txt	Check the undo operation cannot work at the beginning of creating new game.	✓
at06.txt	Test give_up operation only.	✓
at07.txt	Test branch of fire/bomb operation after using custom_setup_test design game.	✓
at08.txt	Test undo/redo operations only.	✓
at09.txt	Test give_up operation after the game is finished when win.	✓
at10.txt	Test give_up operation after the game is finished when lose.	✓
at11.txt	Test give_up operation when the game is not finished.	✓

```
ptl19 301 % python3 ETF_Test.py
Output produced by ../oracle.exe wrote to log/student/at001.expected.txt.
Output produced by ../oracle.exe wrote to log/student/at002.expected.txt.
Output produced by ../oracle.exe wrote to log/student/at01.expected.txt.
Output produced by ../oracle.exe wrote to log/student/at02.expected.txt.
Output produced by ../oracle.exe wrote to log/student/at03.expected.txt.
Output produced by ../oracle.exe wrote to log/student/at04.expected.txt.
Output produced by ../oracle.exe wrote to log/student/at05.expected.txt.
Output produced by ../oracle.exe wrote to log/student/at06.expected.txt.
Output produced by ../oracle.exe wrote to log/student/at07.expected.txt.
Output produced by ../oracle.exe wrote to log/student/at08.expected.txt.
Output produced by ../oracle.exe wrote to log/student/at09.expected.txt.
Output produced by ../oracle.exe wrote to log/student/at10.expected.txt.
Output produced by ../oracle.exe wrote to log/student/at101.expected.txt.
Output produced by ../oracle.exe wrote to log/student/at102.expected.txt.
Output produced by ../oracle.exe wrote to log/student/at11.expected.txt.
========================================================================
Running acceptance test from file ../tests/acceptance/student/at001.txt.
/eecs/home/dengtao/battleship/tests/acceptance/student/at001.txt
Output produced by ../EIFGENs/battleship/W_code/battleship wrote to log/student/at001.actual.txt.
========================================================================
Running acceptance test from file ../tests/acceptance/student/at002.txt.
/eecs/home/dengtao/battleship/tests/acceptance/student/at002.txt
Output produced by ../EIFGENs/battleship/W_code/battleship wrote to log/student/at002.actual.txt.
========================================================================
Running acceptance test from file ../tests/acceptance/student/at01.txt.
/eecs/home/dengtao/battleship/tests/acceptance/student/at01.txt
Output produced by ../EIFGENs/battleship/W_code/battleship wrote to log/student/at01.actual.txt.
========================================================================
Running acceptance test from file ../tests/acceptance/student/at02.txt.
/eecs/home/dengtao/battleship/tests/acceptance/student/at02.txt
Output produced by ../EIFGENs/battleship/W_code/battleship wrote to log/student/at02.actual.txt.
========================================================================
Running acceptance test from file ../tests/acceptance/student/at03.txt.
/eecs/home/dengtao/battleship/tests/acceptance/student/at03.txt
Output produced by ../EIFGENs/battleship/W_code/battleship wrote to log/student/at03.actual.txt.
========================================================================
Running acceptance test from file ../tests/acceptance/student/at04.txt.
/eecs/home/dengtao/battleship/tests/acceptance/student/at04.txt
Output produced by ../EIFGENs/battleship/W_code/battleship wrote to log/student/at04.actual.txt.
========================================================================
Running acceptance test from file ../tests/acceptance/student/at05.txt.
/eecs/home/dengtao/battleship/tests/acceptance/student/at05.txt
Output produced by ../EIFGENs/battleship/W_code/battleship wrote to log/student/at05.actual.txt.
========================================================================
Running acceptance test from file ../tests/acceptance/student/at06.txt.
/eecs/home/dengtao/battleship/tests/acceptance/student/at06.txt
Output produced by ../EIFGENs/battleship/W_code/battleship wrote to log/student/at06.actual.txt.
========================================================================
Running acceptance test from file ../tests/acceptance/student/at07.txt.
/eecs/home/dengtao/battleship/tests/acceptance/student/at07.txt
Output produced by ../EIFGENs/battleship/W_code/battleship wrote to log/student/at07.actual.txt.
========================================================================
Running acceptance test from file ../tests/acceptance/student/at08.txt.
/eecs/home/dengtao/battleship/tests/acceptance/student/at08.txt
Output produced by ../EIFGENs/battleship/W_code/battleship wrote to log/student/at08.actual.txt.
========================================================================
Running acceptance test from file ../tests/acceptance/student/at09.txt.
/eecs/home/dengtao/battleship/tests/acceptance/student/at09.txt
Output produced by ../EIFGENs/battleship/W_code/battleship wrote to log/student/at09.actual.txt.
========================================================================
Running acceptance test from file ../tests/acceptance/student/at10.txt.
/eecs/home/dengtao/battleship/tests/acceptance/student/at10.txt
Output produced by ../EIFGENs/battleship/W_code/battleship wrote to log/student/at10.actual.txt.
========================================================================
Running acceptance test from file ../tests/acceptance/student/at101.txt.
/eecs/home/dengtao/battleship/tests/acceptance/student/at101.txt
Output produced by ../EIFGENs/battleship/W_code/battleship wrote to log/student/at101.actual.txt.
========================================================================
Running acceptance test from file ../tests/acceptance/student/at102.txt.
/eecs/home/dengtao/battleship/tests/acceptance/student/at102.txt
Output produced by ../EIFGENs/battleship/W_code/battleship wrote to log/student/at102.actual.txt.
========================================================================
Running acceptance test from file ../tests/acceptance/student/at11.txt.
/eecs/home/dengtao/battleship/tests/acceptance/student/at11.txt
Output produced by ../EIFGENs/battleship/W_code/battleship wrote to log/student/at11.actual.txt.
============================
Test Results: 15/15 passed.
============================
All tests pass!!!
============================
Test Results: 15/15 passed.
============================
ptl19 302 %
```

7. Appendix (Contract view of all classes)

----------ACTION----------

```
note
    description: "Summary description for {ACTION}."
    author: ""
    date: "$Date$"
    revision: "$Revision$"

deferred class interface
    ACTION

feature

    make

    model: BATTLESHIP

feature -- param

    e_message: ERROR_MESSAGE

    g_message: GAME_MESSAGE

    num: INTEGER_32

end -- class ACTION
```

----------ACTION_BOMB----------

```
note
    description: "Summary description for {ACTION_BOMB}."
    author: ""
    date: "$Date$"
    revision: "$Revision$"

class interface
    ACTION_BOMB

create
    make_bomb

feature

    make_bomb (coordinate1: TUPLE [row: INTEGER_64; column: INTEGER_64]; coordinate2: TUPLE [row: INTEGER_64; column: INTEGER_64])

feature -- param

    level1: TUPLE [row: INTEGER_64; column: INTEGER_64]

    level2: TUPLE [row: INTEGER_64; column: INTEGER_64]

end -- class ACTION_BOMB
```

----------ACTION_CUSTOM_SETUP----------

```
note
    description: "Summary description for {ACTION_CUSTOM_SETUP}."
    author: ""
    date: "$Date$"
    revision: "$Revision$"

class interface
    ACTION_CUSTOM_SETUP

create
    make_custom

feature

    make_custom (d: INTEGER_64; s: INTEGER_64; max_s: INTEGER_64; num_b: INTEGER_64)

feature -- param

    dimension: INTEGER_64

    max_shots: INTEGER_64

    num_bombs: INTEGER_64

    ships: INTEGER_64

end -- class ACTION_CUSTOM_SETUP
```

----------ACTION_CUSTOM_SETUP_TEST----------

```
note
    description: "Summary description for {ACTION_CUSTOM_SETUP_TEST}."
    author: ""
    date: "$Date$"
    revision: "$Revision$"

class interface
    ACTION_CUSTOM_SETUP_TEST

create
    make_custom

feature

    make_custom (d: INTEGER_64; s: INTEGER_64; max_s: INTEGER_64; num_b: INTEGER_64)

feature -- param

    dimension: INTEGER_64

    max_shots: INTEGER_64

    num_bombs: INTEGER_64

    ships: INTEGER_64

end -- class ACTION_CUSTOM_SETUP_TEST
```

----------ACTION_DEBUG_TEST----------

```
note
    description: "Summary description for {ACTION_DEBUG_TEST}."
    author: ""
    date: "$Date$"
    revision: "$Revision$"

class interface
    ACTION_DEBUG_TEST

create
    make_start

feature

    make_start (l: INTEGER_64)

feature -- param

    level: INTEGER_64

end -- class ACTION_DEBUG_TEST
```

----------ACTION_ERRORS----------

```
note
    description: "Summary description for {ACTION_ERRORS}."
    author: ""
    date: "$Date$"
    revision: "$Revision$"

class interface
    ACTION_ERRORS

create
    make

end -- class ACTION_ERRORS
```

----------ACTION_FIRE----------

```
note
    description: "Summary description for {ACTION_FIRE}."
    author: ""
    date: "$Date$"
    revision: "$Revision$"

class interface
    ACTION_FIRE

create
    make_fire

feature

    make_fire (coordinate: TUPLE [row: INTEGER_64; column: INTEGER_64])

feature -- param

    level: TUPLE [row: INTEGER_64; column: INTEGER_64]

end -- class ACTION_FIRE
```

----------ACTION_NEW_GAME----------

```
note
    description: "Summary description for {ACTION_NEW_GAME}."
    author: ""
    date: "$Date$"
    revision: "$Revision$"

class interface
    ACTION_NEW_GAME

create
    make_start

feature

    make_start (l: INTEGER_64)

feature -- param

    level: INTEGER_64

end -- class ACTION_NEW_GAME
```

----------BATTLESHIP----------

```
note
    description: "A default business model."
    author: "Jackie Wang"
    date: "$Date$"
    revision: "$Revision$"

class interface
    BATTLESHIP

create {BATTLESHIP_ACCESS}
    make

feature -- model attributes

    current_game: INTEGER_32

    g: detachable GAME

    game_message_index: INTEGER_32

    game_message_record: ARRAY [GAME_MESSAGE]

    gave_up: BOOLEAN

    i: INTEGER_32

    message: ERROR_MESSAGE

    message2: GAME_MESSAGE

    redoing: BOOLEAN

    s: STRING_8

    total: INTEGER_32

    total_upper: INTEGER_32
```

```
feature -- model operations

    default_update
            -- Perform update to the model state.

    reset
            -- Reset model state.

feature -- queries

    out: STRING_8
            -- New string containing terse printable representation
            -- of current object

feature --COMMAND

    bomb (coordinate1: TUPLE [row: INTEGER_64; column: INTEGER_64]; coordinate2: TUPLE [row: INTEGER_64; column: INTEGER_64])

    custom_setup (dimension: INTEGER_64; ships: INTEGER_64; max_shots: INTEGER_64; num_bombs: INTEGER_64)

    custom_setup_test (dimension: INTEGER_64; ships: INTEGER_64; max_shots: INTEGER_64; num_bombs: INTEGER_64)

    debug_test (le: INTEGER_64)

    errors

    fire (coordinate: TUPLE [row: INTEGER_64; column: INTEGER_64])

    give_up

    new_game (le: INTEGER_64)

    redo

    undo

invariant
    valid_game_message_index: game_message_index <= i
    valid_total: total <= total_upper

end -- class BATTLESHIP
```

----------BATTLESHIP_ACCESS----------

```
note
    description: "Singleton access to the default business model."
    author: "Jackie Wang"
    date: "$Date$"
    revision: "$Revision$"

expanded class interface
    BATTLESHIP_ACCESS

create
    default_create

feature

    M: BATTLESHIP

invariant
        M = M

end -- class BATTLESHIP_ACCESS
```

----------GAME----------

```
note
    description: "Summary description for {GAME}."
    author: ""
    date: "$Date$"
    revision: "$Revision$"

class interface
    GAME

create
    make

feature -- attributes

    board: ARRAY2 [SHIP_ALPHABET]

    bombs_time: INTEGER_32

    bombs_upper: INTEGER_32

    is_debug: BOOLEAN

    Row_indices: ARRAY [CHARACTER_8]

    ships: ARRAYED_LIST [TUPLE [size: INTEGER_32; row: INTEGER_32; col: INTEGER_32; dir: BOOLEAN]]

    shots_time: INTEGER_32

    shots_upper: INTEGER_32

feature -- checker

    check_game_over: BOOLEAN

    check_sunk (ship: TUPLE [size: INTEGER_32; row: INTEGER_32; col: INTEGER_32; dir: BOOLEAN]): BOOLEAN

    check_win: BOOLEAN

    hit_ship (coordinate: TUPLE [row: INTEGER_64; column: INTEGER_64]): detachable TUPLE [size: INTEGER_32; row: INTEGER_32; col: INTEGER_32; dir: BOOLEAN]
            --return the ship be hitted or NULL

feature -- creation

    make (is_debug_mode: BOOLEAN; board_size: INTEGER_32; shots_limit: INTEGER_32; bombs_limit: INTEGER_32; num_ships: INTEGER_32)

feature -- history

    history: HISTORY

feature -- output

    out: STRING_8
            -- Return string representation of current game.
            -- You may reuse this routine.

feature -- query

    bomb (coordinate1: TUPLE [row: INTEGER_64; column: INTEGER_64]; coordinate2: TUPLE [row: INTEGER_64; column: INTEGER_64]; message: GAME_MESSAGE)
        ensure
            bomb_coordinate1_changed: board [coordinate1.row.as_integer_32, coordinate1.column.as_integer_32].out ~ "O"
                    or board [coordinate1.row.as_integer_32, coordinate1.column.as_integer_32].out ~ "X"
            bomb_coordinate2_changed: board [coordinate2.row.as_integer_32, coordinate2.column.as_integer_32].out ~ "O"
                    or board [coordinate2.row.as_integer_32, coordinate2.column.as_integer_32].out ~ "X"
    fire (coordinate: TUPLE [row: INTEGER_64; column: INTEGER_64]; message: GAME_MESSAGE)
        ensure
            fire_coordinate_changed: board [coordinate.row.as_integer_32, coordinate.column.as_integer_32].out ~ "O"
                    or board [coordinate.row.as integer 32, coordinate.column.as integer 32].out ~ "X"
    unbomb (coordinate1: TUPLE [row: INTEGER_64; column: INTEGER_64]; coordinate2: TUPLE [row: INTEGER_64; column: INTEGER_64])
        ensure
            unbomb_coordinate1_changed: board [coordinate1.row.as_integer_32, coordinate1.column.as_integer_32].out ~ "_"
                    or board [coordinate1.row.as_integer_32, coordinate1.column.as_integer_32].out ~ "v"
                    or board [coordinate1.row.as_integer_32, coordinate1.column.as_integer_32].out ~ "h"
            unbomb_coordinate2_changed: board [coordinate2.row.as_integer_32, coordinate2.column.as_integer_32].out ~ "_"
                    or board [coordinate2.row.as_integer_32, coordinate2.column.as_integer_32].out ~ "v"
                    or board [coordinate2.row.as_integer_32, coordinate2.column.as_integer_32].out ~ "h"

    unfire (coordinate: TUPLE [row: INTEGER_64; column: INTEGER_64])
        ensure
            unfire_coordinate_changed: board [coordinate.row.as_integer_32, coordinate.column.as_integer_32].out ~ "_"
                    or board [coordinate.row.as_integer_32, coordinate.column.as_integer_32].out ~ "v"
                    or board [coordinate.row.as_integer_32, coordinate.column.as_integer_32].out ~ "h"
feature -- random generators

    debug_gen: RANDOM_GENERATOR
            -- deterministic generator for debug mode
            -- it's important to keep this as an attribute

    rand_gen: RANDOM_GENERATOR
            -- random generator for normal mode
            -- it's important to keep this as an attribute
```

```
feature -- utilities

    collide_with (existing_ships: ARRAYED_LIST [TUPLE [size: INTEGER_32; row: INTEGER_32; col: INTEGER_32; dir: BOOLEAN]];
                  new_ship: TUPLE [size: INTEGER_32; row: INTEGER_32; col: INTEGER_32; dir: BOOLEAN]): BOOLEAN
            -- Does `new_ship` collide with the set of `existing_ships`?
        ensure
                Result = across
                    existing_ships as existing_ship
                some
                    collide_with_each_other (new_ship, existing_ship.item)
                end

    collide_with_each_other (ship1, ship2: TUPLE [size: INTEGER_32; row: INTEGER_32; col: INTEGER_32; dir: BOOLEAN]): BOOLEAN
            -- Does `ship1` collide with `ship2`?

     generate_ships (is_debug_mode: BOOLEAN; board_size: INTEGER_32; num_ships: INTEGER_32):
              ARRAYED_LIST [TUPLE [size: INTEGER_32; row: INTEGER_32; col: INTEGER_32; dir: BOOLEAN]]
            -- places the ships on the board
            -- either deterministicly random or completely random depending on debug mode

    place_new_ships (new_ships: ARRAYED_LIST [TUPLE [size: INTEGER_32; row: INTEGER_32; col: INTEGER_32; dir: BOOLEAN]])
            -- Place the randomly generated positions of `new_ships` onto the board.
            -- Notice that when a ship's row and column are given,
            -- its coordinate starts with (row + 1, col) for a vertical ship,
            -- and starts with (row, col + 1) for a horizontal ship.
        require
                across
                    new_ships.Lower |..| new_ships.upper as i
                all
                    across
                        new_ships.Lower |..| new_ships.upper as j
                    all
                        i.item /= j.item implies not collide_with_each_other (new_ships [i.item], new_ships [j.item])
                    end
                end

feature --faker_attributes

    score_time: INTEGER_32

    score_upper: INTEGER_32

    ships_time: INTEGER_32

    ships_upper: INTEGER_32

end -- class GAME
```

----------ERROR_MESSAGE----------

```
note
    description: "Summary description for {ERROR_MESSAGE}."
    author: ""
    date: "$Date$"
    revision: "$Revision$"

class interface
    ERROR_MESSAGE

create
    make

feature --attributes

    message: STRING_8

feature --creations

    make

feature --error

    e1

    e10

    e11

    e12

    e13

    e14
```

```
    e15

    e2

    e3

    e4

    e5

    e6

    e7

    e8

    e9

feature --ok

    ok

feature --out

    out: STRING_8
            -- New string containing terse printable representation
            -- of current object

feature --setter

    set (other: ERROR_MESSAGE; i: INTEGER_32)

end -- class ERROR_MESSAGE
```

----------GAME_MESSAGE----------

```
note
    description: "Summary description for {GAME_MESSAGE}."
    author: ""
    date: "$Date$"
    revision: "$Revision$"

class interface
    GAME_MESSAGE

create
    make

feature --attributes

    message: STRING_8

    message_fire_feedback: STRING_8

feature --creations

    make

feature --ok

    s0

    s1

    s10

    s2

    s3

    s4

    s5
```

```
    s6

    s7

    s8 (size: INTEGER_32)

    s9 (size1, size2: INTEGER_32)

feature --out

    out: STRING_8
            -- New string containing terse printable representation
            -- of current object

feature --setter

    set (other: GAME_MESSAGE)

end -- class GAME_MESSAGE
```

----------HISTORY----------

```
note
    description: "Summary description for {HISTORY}."
    author: ""
    date: "$Date$"
    revision: "$Revision$"

class interface
    HISTORY

create
    make

feature -- comands

    back
        require
                not before

    extend_history (a_op: ACTION)
            -- remove all operations to the right of the current
            -- cursor in history, then extend with `a_op`
        ensure
                history [history.count] = a_op

    forth
        require
                not after

    remove_right
            --remove all elements
            -- to the right of the current cursor in history

feature -- queries

    after: BOOLEAN
            -- Is there no valid cursor position to the right of cursor?

    before: BOOLEAN
            -- Is there no valid cursor position to the left of cursor?

    before2: BOOLEAN
            -- Is there no valid cursor position to the left one of cursor?

    item: ACTION
            -- Cursor points to this user operation
        require
                on_item

    on_item: BOOLEAN
            -- cursor points to a valid operation
            -- cursor is not before or after

end -- class HISTORY
```

----------SHIP_ALPHABET----------

```
note
    description: "Summary description for {SHIP_ALPHABET}."
    author: ""
    date: "$Date$"
    revision: "$Revision$"

class interface
    SHIP_ALPHABET

create
    make

feature -- Attributes

    item: CHARACTER_8

feature -- Commands

    make (a_char: CHARACTER_8)

feature -- output

    out: STRING_8
            -- Return string representation of alphabet.

invariant
    allowable_symbols: item = '_' or item = 'h' or item = 'v' or item = 'O' or item = 'X'

end -- class SHIP_ALPHABET
```

----------RANDOM_GENERATOR----------

```
note
    description: "Summary description for {RANDOM_GENERATOR}."
    author: ""
    date: "$Date$"
    revision: "$Revision$"

class interface
    RANDOM_GENERATOR

create
    make_debug,
    make_random

feature -- commands

    forth
            -- sets the row, column and direction variables forward
            -- should be called for a new ship or if there is a collision

feature -- queries

    column: INTEGER_32
            -- returns a random variable used to generate column coordinates

    direction: INTEGER_32
            -- returns a random variable used to generate direction

    row: INTEGER_32
            -- returns a random variable used to generate row coordinates

end -- class RANDOM_GENERATOR
```