

Informatics Institute of Technology

Formal Methods

6SENG005C.1

Course Work

Report

Pasindu Anuradha Godakanda

Uow Number – w1871482

IIT Number – 20212026

# Machine Structure Diagram

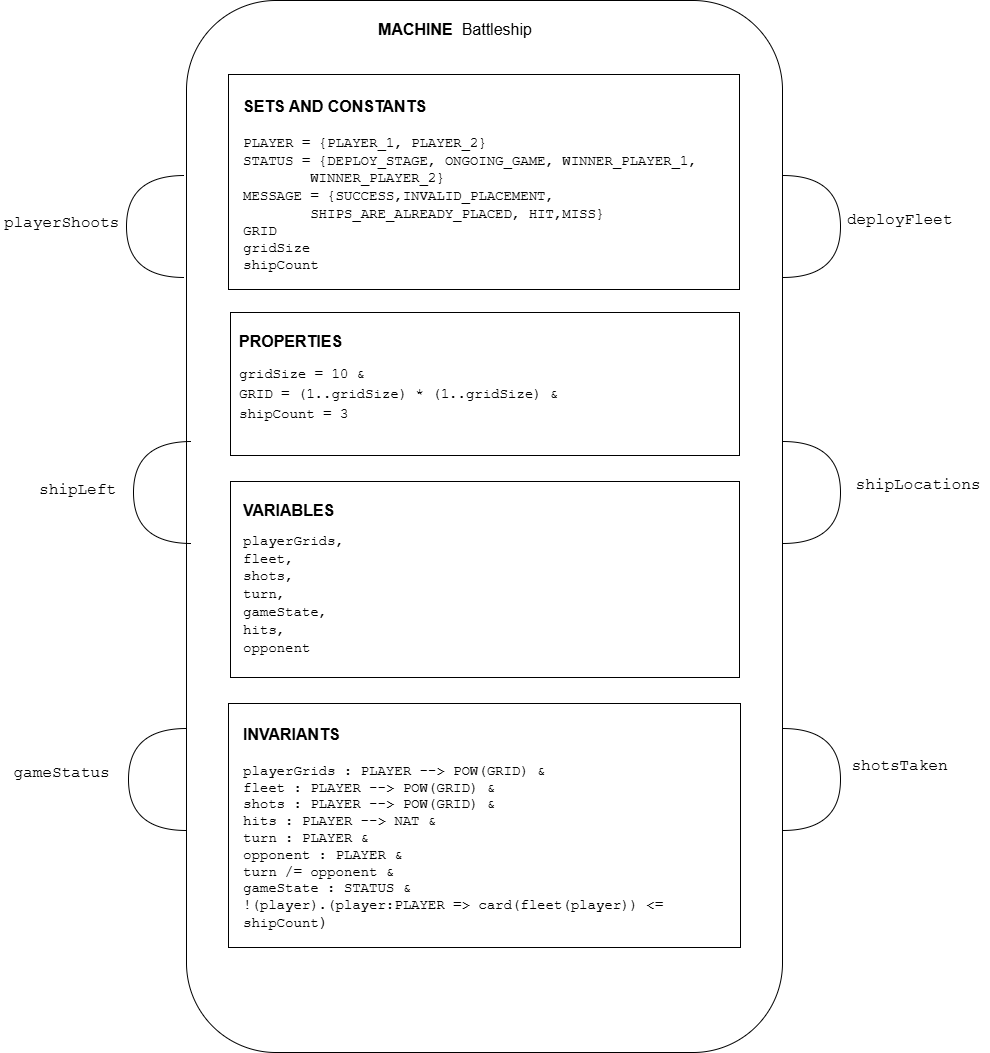


Figure : Machine Structure Diagram

# Justifications:

Detailed explanations for the assigned and utilized SETS, VARIABLES and INVARIENTS are provided below.

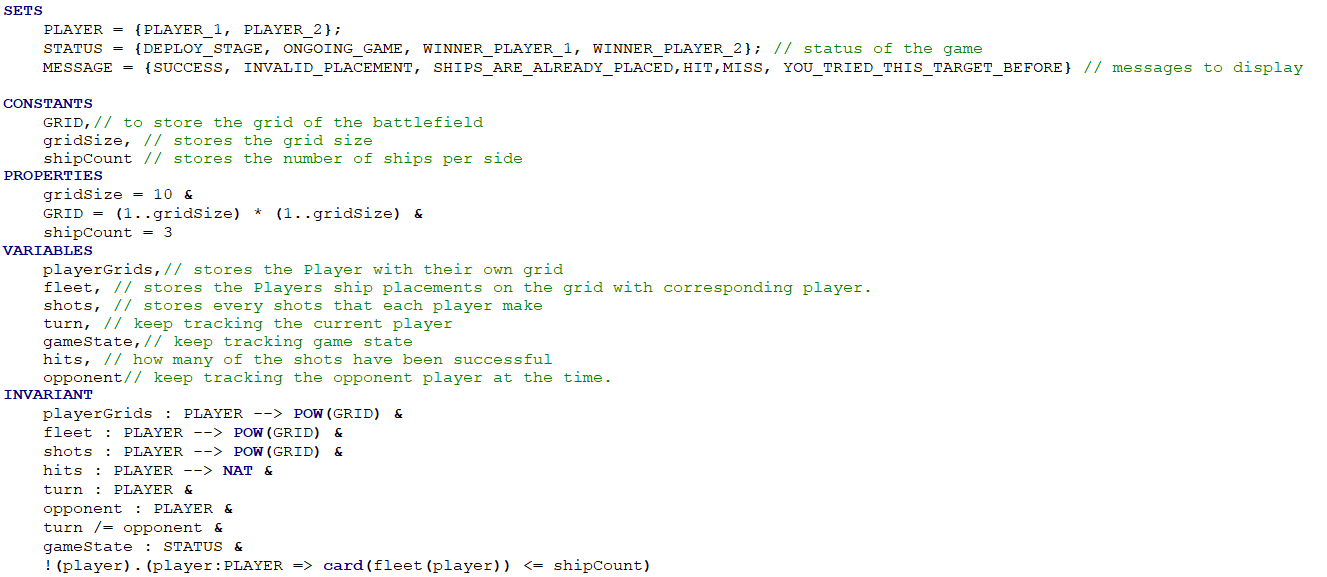


Figure : SETS, VARIABLES and INVARIENTS

## SETS & CONSTANTS

|  |  |
| --- | --- |
| **SET & CONSTANTS** | **Explanation** |
| PLAYER = {PLAYER\_1, PLAYER\_2} | This Represents the two participants in the game. Battleship game involves two players, requiring this set to distinguish between them. |
| STATUS = {DEPLOY\_STAGE, ONGOING\_GAME, WINNER\_PLAYER\_1, WINNER\_PLAYER\_2} | This Represents different game status. This is requires to keep track the game progression throughout the different stages of the game. |
| MESSAGE = {SUCCESS, INVALID\_PLACEMENT, SHIPS\_ARE\_ALREADY\_PLACED,HIT,MISS, YOU\_TRIED\_THIS\_TARGET\_BEFORE} | This Represents the feedback messages. This Requires to provide relevant game notifications during the play. |
| GRID | This is the game board coordinates which defines the playable area for ship placement and targeting. |
| gridSize | This constant used for set the grid dimensions. As for this specification it uses 10 x 10 grid, hence the grid size would be 10. |
| shipCount | This is the number of ships that each player can deploy. This fixes ship count for balanced gameplay. |

## VARIABLES & INVARIENTS

|  |  |
| --- | --- |
| **VARIABLES & INVARIENTS** | **Explanation** |
| playerGrids  playerGrids : PLAYER --> POW(GRID) | playerGrids uses to map players to their own grids and this keeps track of each player’s board setup. Each player must have a grid of valid coordinates. This ensures proper grid assignment. |
| Fleet  fleet : PLAYER --> POW(GRID)  !(player).(player:PLAYER => card(fleet(player)) <= shipCount) | Stores ships’ positions of each player for ship placement validation and hit detection.  Each player’s fleet must consist of valid coordinates within the grid.  A player’s fleet cannot exceed the allowed ship count. |
| Shots  shots : PLAYER --> POW(GRID) | Tracks all shots fired by each player for prevent duplicate shots and helps determine game progress.  All the shots must be valid grid coordinates. |
| Hits  hits : PLAYER --> NAT | Tracks successful hits per player. This requires for determine when a player wins the game.  Since negative hits are logically impossible and there can be an event that a player don’t have any successful hit. Hence this tracks natural numbers including 0. |
| Turn  turn : PLAYER | The game is alternates between player’s turn, hence the turn variable tracks the current player who’s attacking at the moment.  The current turn must belong to a valid player and it must be always assigned. |
| opponent  opponent : PLAYER  turn /= opponent | This is the opposing player during the current turn. This requires for determine whose ships are targeted.  Opponent also should be a valid player and since players cannot play against themselves, opponent cannot be the turn at the same time. |
| gameState  gameState : STATUS | This Represents the current state of the game. This is requires to manage game flow from deployment to conclusion.  gameState should be always a valid STATUS. |

# Machine Testing results:

## Type Check Result

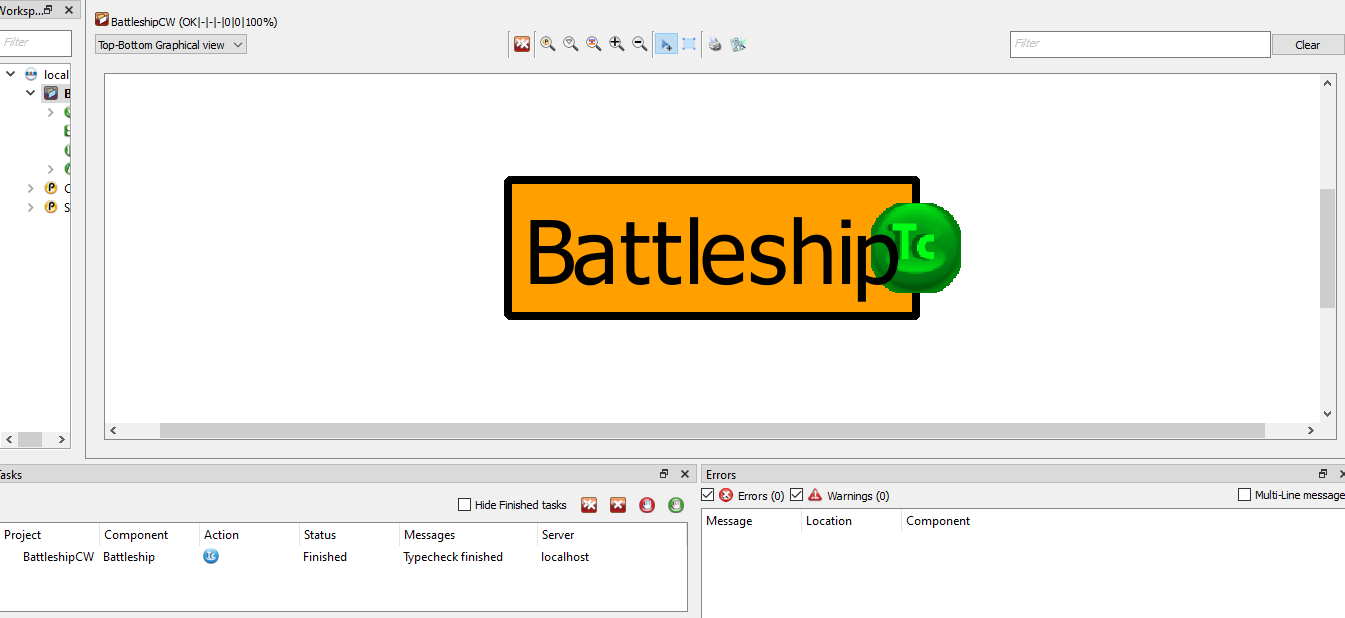


Figure : Type Check result

# Graphical Visualization

|  |  |  |  |
| --- | --- | --- | --- |
|  | Opposing Player (who takes the attack) |  | Current Player (who attacks) |
|  | Missed shot |  | Unharmed Ship |
|  | Successful Hit (sinking Ship) |  |  |

