Battleship

Built by CTRL + V, CTRL + C

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

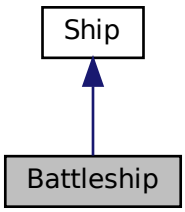
Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

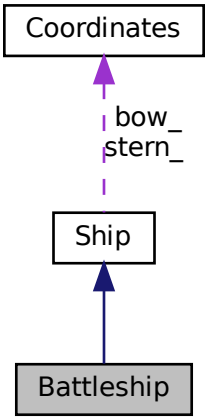
Class Documentation

3.1 Battleship Class Reference

Inheritance diagram for Battleship:



Collaboration diagram for Battleship:



Public Member Functions

- Battleship (Coordinates bow, Coordinates stern)

Static Public Attributes

- static const int **DEFAULT_SIZE** = 5

Additional Inherited Members

3.1.1 Constructor & Destructor Documentation

3.1.1.1 Battleship()

```
Battleship::Battleship (
            Coordinates bow,
            Coordinates stern )
```

Creates a Ship of type BATTLESHIP starting from the bow and stern. If the bow to stern distance does not match the vessel size, throw a std::invalid_argument

Exceptions

std::invali_argument	

Parameters

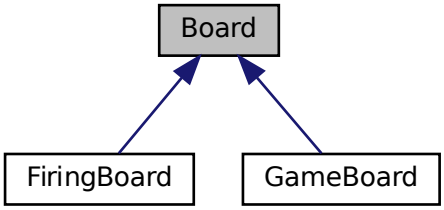
bow	The bow of the ship
stern	The stern of the ship

The documentation for this class was generated from the following file:

- Battleship.h

3.2 Board Class Reference

Inheritance diagram for Board:



Public Member Functions

- Board (int rows, int cols)
- int **GetRows** () const
- void **SetRows** (int rows)
- int **GetCols** () const
- void **SetCols** (int cols)

Private Attributes

- int **rows_**
- int **cols_**

3.2.1 Constructor & Destructor Documentation

3.2.1.1 Board()

```
Board::Board (
            int rows,
            int cols )  [inline]
```

Create a Board object of dimension rows x cols

Parameters

rows	
cols	

The documentation for this class was generated from the following file:

- Board.h

3.3 Coordinates Class Reference

Public Member Functions

- **Coordinates** (std::pair< int, int > coordinates)
- Coordinates (std::string &user_coordinates)
- Coordinates (int row, int col)
- int **GetRow** () const
- int **GetCol** () const
- int CalculateOffsetTo (Coordinates other) const
- std::string ToUserCoordinates () const

  Formats the coordinates in the format: XY, where X is a letter in 'ABCDEFHILMN' and Y is a number between 1 and 12.
- void **SetRow** (int row)
- void **SetCol** (int col)

Static Public Member Functions

- static bool **IsValid** (std::pair< int, int > row_col)
- static bool IsValid (int row, int col)
- static std::vector< Coordinates > GetCoordinatesBetween (Coordinates start, Coordinates end)
- static Coordinates GetRandomCoordinates ()
- static std::set< Coordinates > GetAdjacentStarCoordinates (Coordinates current)

Static Private Member Functions

- static Coordinates ParseCoordinates (std::string &coordinates)

Private Attributes

- std::pair< int, int > **row_col_**

3.3.1 Constructor & Destructor Documentation

3.3.1.1 Coordinates() [1/2]

```
Coordinates::Coordinates (
            std::string & user_coordinates )  [explicit]
```

Starting from the user's coordinates (XY, where X is a letter in 'ABCDEFGHILMN' and Y is a number between 1 and 12), create a Coordinates object. The coordinates are converted according to the following scheme: X -> A number between 0 and 11 Y -> The number itself -1

Parameters

user_coordinates	Coordinates supplied by the user.

3.3.1.2 Coordinates() [2/2]

```
Coordinates::Coordinates (
            int row,
            int col )
```

Parameters

row	number representing the row
col	number representing the col

Exceptions

std::invalid_argument	if the coordinates are not valid.

3.3.2 Member Function Documentation

3.3.2.1 CalculateOffsetTo()

```
int Coordinates::CalculateOffsetTo (
            Coordinates other ) const
```

Calculates the offset between this and other. The offset is calculated only if both coordinates are in the same row or column. If the coordinates are not on the same row/column an exception is thrown std::invalid_argument

Parameters

coordinates	

Exceptions

std::invalid_argument	if the coordinates are not in the same row/column

Returns

The number of cells between this and other, calculated as the difference between rows/columns.

3.3.2.2 GetAdjacentStarCoordinates()

```
static std::set<Coordinates> Coordinates::GetAdjacentStarCoordinates (
            Coordinates current )  [static]
```

Returns a vector composed of the coordinates immediately to the right, left, above and below `current`. Esempio: (X = current)
A B C
D X E
F G H
In this case it returns { B, D, E, G}

3.3.2.3 GetCoordinatesBetween()

```
static std::vector<Coordinates> Coordinates::GetCoordinatesBetween (
            Coordinates start,
            Coordinates end )  [static]
```

Generates all coordinates between `start` and `end`

Parameters

start	
end	

Returns

A vector containing all the coordinates between `start` and `end`

3.3.2.4 GetRandomCoordinates()

```
static Coordinates Coordinates::GetRandomCoordinates ( )  [static]
```

Generates random coordinates

Returns

Random coordinates

3.3.2.5 IsValid()

```
static bool Coordinates::IsValid (
            int row,
            int col )  [static]
```

Returns

Returns true if 0 <= row < 12 and 0 <= col < 12, false otherwise.

3.3.2.6 ParseCoordinates()

```
static Coordinates Coordinates::ParseCoordinates (
            std::string & coordinates )  [static], [private]
```

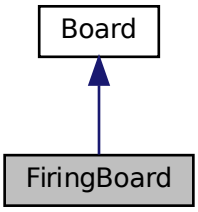
Starting from the user's coordinates (XY, where X is a letter in 'ABCDEFGHILMN' and Y is a number between 1 and 12), create a Coordinates object. The coordinates are converted according to the following scheme: X -> A number between 0 and 11 Y -> The number itself -1

The documentation for this class was generated from the following file:

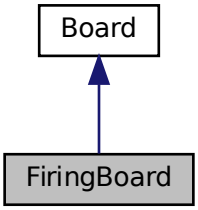
- Coordinates.h

3.4 FiringBoard Class Reference

Inheritance diagram for FiringBoard:



Collaboration diagram for FiringBoard:



Public Member Functions

- void MarkAttack (Coordinates target, bool is_hit)
- void ClearSubmarineSightings ()

  Clears all the submarine sightings marked on the board.
- void **Clear** ()
- std::string ToString () const
- bool HasBeenAttacked (Coordinates coordinates)
- void AddSubmarineSightings (const std::map< Coordinates, OccupationType > &sightings)
- void ClearSuccessfulHits ()

  Clears all the coordinates with status HIT from the board.
- void ClearUnsuccessfulHits ()

  Clears all the coordinates with status MISS from the board.

Private Attributes

- std::map< Coordinates, OccupationType > **tiles_**

3.4.1 Member Function Documentation

3.4.1.1 AddSubmarineSightings()

```
void FiringBoard::AddSubmarineSightings (
            const std::map< Coordinates, OccupationType > & sightings )
```

Adds the sightings from the submarine to the board. The sightings are marked in the grid with OCCUPIED if a ship is present in the opponent board, but not yet hit, or with HIT if it's present and hit.

Parameters

sightings	A map of coordinates and their status in the opponent's board.

3.4.1.2 HasBeenAttacked()

```
bool FiringBoard::HasBeenAttacked (
            Coordinates coordinates )
```

Returns

true if `coordinates` has been attacked (HIT/MISS), false otherwise.

3.4.1.3 MarkAttack()

```
void FiringBoard::MarkAttack (
            Coordinates target,
            bool is_hit )
```

Marks the `target` coordinate as HIT if is_hit is true, or with MISS otherwise.

Parameters

target	The coordinate under attack
is_hit	true: successful attack, false: unsuccessful.

3.4.1.4 ToString()

```
std::string FiringBoard::ToString ( ) const
```

Returns

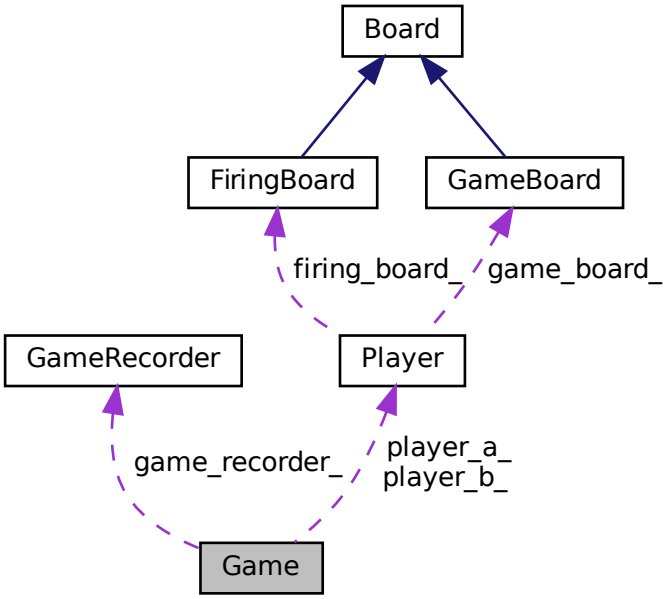
FiringBoard as a string in the form of a grid rows_ x columns_

The documentation for this class was generated from the following file:

- FiringBoard.h

3.5 Game Class Reference

Collaboration diagram for Game:



Public Member Functions

- Game ()

  Creates a game object and asks the user for the game object before initializing the players.
- Game (GameMode mode)

  Creates a game object with the mode specified in `mode`.
- Game (GameRecorder game_recorder)

  Creates a game object with mode GameMode::REPLAY and the game_recorder passed as param.
- void PlayGame ()

  Initializes the main game loop, based on `game_mode_` *starts a game.*
- void PlayComputerVsComputerGame ()
- void PlayComputerVsHumanGame ()
- std::string Replay (bool to_ostream)
- bool PlayMove (Player &attacker, Player &opponent, std::pair< Coordinates, Coordinates > move)
- void PlaceShipsFromUser (Player &player)
- UserCommand GetUserCommand (const std::string &prompt)

  Receives and parses a command from the user. See `UserCommand`.
- bool AttemptToPlaceAShip (Player &player, std::pair< Coordinates, Coordinates > bow_stern, Ship &ship)

Static Public Member Functions

- static int ReadChoiceFromUser (const std::set< int > &available_choices)

Private Attributes

- const int **MAX_ROUNDS** = 200
- const bool **LOGGING_ACTIVE** = true
- Player **player_b_**
- Player **player_a_**
- GameMode **game_mode_**
- GameRecorder **game_recorder_**

3.5.1 Member Function Documentation

3.5.1.1 AttemptToPlaceAShip()

```
bool Game::AttemptToPlaceAShip (
            Player & player,
            std::pair< Coordinates, Coordinates > bow_stern,
            Ship & ship )
```

Attempts to place a ship in `player's` GameBoard. See `Player::PlaceShip`

Returns

true if the has been placed successfully, false otherwise.

3.5.1.2 PlaceShipsFromUser()

```
void Game::PlaceShipsFromUser (
            Player & player )
```

Places the ships on `player` GameBoard, by asking the user for coordinates of bow and stern of each ship to place.

Parameters

player	

3.5.1.3 PlayComputerVsComputerGame()

```
void Game::PlayComputerVsComputerGame ( )
```

Starts a Computer vs Computer Game. The game ends after one of the players loses or after MAX_ROUNDS rounds have been played.

3.5.1.4 PlayComputerVsHumanGame()

```
void Game::PlayComputerVsHumanGame ( )
```

Starts a Computer vs Human game. For the computer rounds the move are generated randomly, for the human's ones the user is asked for a move. The game ends after one of the players loses.

3.5.1.5 PlayMove()

```
bool Game::PlayMove (
            Player & attacker,
            Player & opponent,
            std::pair< Coordinates, Coordinates > move )
```

Performs a `move`.

Parameters

attacker	The current player at this turn
opponent	The opponent.
move	A pair of coordinates (origin, target) representing the center of a ship (origin) and a target, based on the type of ship target can be the square to attack or the square to move to.

Returns

if the move has been performed successfully.

3.5.1.6 ReadChoiceFromUser()

```
static int Game::ReadChoiceFromUser (
            const std::set< int > & available_choices )  [static]
```

Utility function to read a choice from the user between a set of available choices. It keeps asking for a valid choice until the user types a valid choice.

Parameters

available_choices	

Returns

3.5.1.7 Replay()

```
std::string Game::Replay (
            bool to_ostream )
```

Replays a game from GameRecorder If `to_ostream` is set to true the replay of each round gets printed to std::cout, otherwise a string will be returned with the with the replay of each round

Parameters

to_ostream	bool to control the output, true: the output is std::cout, false the output is the return value.

Returns

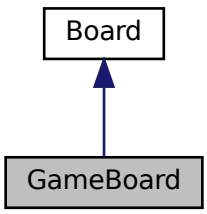
if to_stream is true, returns a string with the replay, otherwise an empty string

The documentation for this class was generated from the following file:

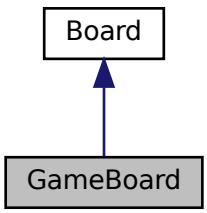
- Game.h

3.6 GameBoard Class Reference

Inheritance diagram for GameBoard:



Collaboration diagram for GameBoard:



Public Member Functions

- bool PlaceShip (Coordinates bow, Coordinates stern, const Ship &ship)
- std::shared_ptr< Ship > GetShipAt (Coordinates location)
- bool MoveShip (Coordinates origin, Coordinates target)
- bool ReceiveAttack (Coordinates target)
- bool CanPlaceShip (const Ship &ship) const
- bool IsInsideBoard (const Ship &ship) const
- std::string **ToString** () const
- const std::map< Coordinates, std::shared_ptr< Ship > > & GetOccupiedLocations () const
- void RemoveShip (Coordinates coordinates)
- std::pair< Coordinates, Coordinates > GetBowAndSternFromCenter (const Coordinates &center, const std::shared_ptr< Ship > &ship) const
- int GetAvailableBattleships () const

  Returns the number of active battleships in the board.

Private Attributes

- std::map< Coordinates, std::shared_ptr< Ship > > **occupied_locations_**
- std::set< Coordinates > **protected_coordinates_**
- int **available_battleships** = 3

3.6.1 Member Function Documentation

3.6.1.1 CanPlaceShip()

```
bool GameBoard::CanPlaceShip (
            const Ship & ship ) const
```

Checks that the `ship` does not overlaps any other ships in the board, and it's fully inside the board.

Parameters

ship	

Returns

true: if ship doesn't overlap any other ship and it's inside the board , false otherwise.

3.6.1.2 GetBowAndSternFromCenter()

```
std::pair<Coordinates, Coordinates> GameBoard::GetBowAndSternFromCenter (
            const Coordinates & center,
            const std::shared_ptr< Ship > & ship ) const
```

Returns the coordinates of the bow and stern starting from those of the center. Ship is needed to get the width and orientation of the ship. \paramCenter

Parameters

ship	

Returns

3.6.1.3 GetOccupiedLocations()

```
const std::map<Coordinates, std::shared_ptr<Ship> >& GameBoard::GetOccupiedLocations ( )
const
```

Returns all the occupied locations any ship

Returns

A map of pairs of coordinates and the ship at that position.

3.6.1.4 GetShipAt()

```
std::shared_ptr<Ship> GameBoard::GetShipAt (
            Coordinates location )
```

Parameters

location	

Returns

Returns a pointer to the ship contained in the location `location`, or *null_ptr* if the cell is empty

3.6.1.5 IsInsideBoard()

```
bool GameBoard::IsInsideBoard (
            const Ship & ship ) const
```

Checks that ship is within the bounds of board. Basically check that both ship.GetBow() and ship.GetStern() are inside borders.

Parameters

ship	

Returns

true: if the ship is within the margins, false: otherwise

3.6.1.6 MoveShip()

```
bool GameBoard::MoveShip (
            Coordinates origin,
            Coordinates target )
```

Changes this ship's bow and stern. Bow and stern are calculated using target as the ship's center cell.

Parameters

origin	The central cell where the ship is contained
target	The center cell of the ship's new position.

Returns

true if it was possible to change the ship's position, otherwise false. Returns false if the ship goes off the edge of board, or one of the destination cells are already occupied.

3.6.1.7 PlaceShip()

```
bool GameBoard::PlaceShip (
            Coordinates bow,
            Coordinates stern,
            const Ship & ship )
```

Places a ship inside this board.

Parameters

bow	The bow of the ship
stern	The stern of the ship
ship	The ship to place on the grid

Returns

true if the ship was placed correctly, false if the ship could not be placed. May return false if the cells where the ship should be placed are already occupied or outside the board.

3.6.1.8 ReceiveAttack()

```
bool GameBoard::ReceiveAttack (
            Coordinates target )
```

Handles an attack by an enemy ship. If the attack is successful returns true and marks the affected ship cell as attacked. Otherwise it returns false.

Parameters

target	

Returns

true: attack successful, false: attack unsuccessful.

3.6.1.9 RemoveShip()

```
void GameBoard::RemoveShip (
            Coordinates coordinates )
```

Remove the vessel from the board.. To get the coordinates where the ship is, it call the ship's GetLocations() method and eliminate them one by one.

Parameters

coordinates	The center of the ship.

The documentation for this class was generated from the following file:

- GameBoard.h

3.7 GameRecorder Class Reference

Public Member Functions

- GameRecorder (GameMode game_mode)

  Creates a GameRecorder object base on the GameMode `game_mode` *passed.*

- void RecordShipPlacement (Coordinates bow, Coordinates stern, int ship_with)
- void RecordMove (std::pair< Coordinates, Coordinates > move)
- void LoadGameFromLog (const std::string &log_path)
- void PersistGameToLog ()
- void SetIsPlayerATurn (bool is_player_a_turn)
- int GetStartingPlayer () const
- void SetStartingPlayer (int starting_player)
- std::string ToString () const
- const std::map< std::pair< Coordinates, Coordinates >, int > & GetPlayerAShipPlacement () const
- const std::map< std::pair< Coordinates, Coordinates >, int > & GetPlayerBShipPlacement () const
- const std::vector< std::pair< Coordinates, Coordinates > > & GetMoves () const

Static Public Attributes

- static const std::string **LOG_PATH**

Private Attributes

- std::map< std::pair< Coordinates, Coordinates >, int > **player_a_ship_placement_**
- std::map< std::pair< Coordinates, Coordinates >, int > **player_b_ship_placement_**
- std::vector< std::pair< Coordinates, Coordinates > > **moves_**
- bool **player_a_turn_**
- int **starting_player_**
- GameMode **game_mode_**

Friends

- std::ostream & operator<< (std::ostream &os, const GameRecorder &recorder)

  Prints the string returned by `ToString()` to the output stream `os`.

3.7.1 Member Function Documentation

3.7.1.1 GetMoves()

```
const std::vector<std::pair<Coordinates, Coordinates> >& GameRecorder::GetMoves ( ) const
```

Returns the moves played during the game

Returns

a vector of pairs of coordinates (origin, target)

3.7.1.2 GetPlayerAShipPlacement()

```
const std::map<std::pair<Coordinates, Coordinates>, int>& GameRecorder::GetPlayerAShip↩
Placement ( ) const
```

Returns the Player A's ship placement

Returns

a map of a pair of coordinates (bow, stern) and a int (width)

3.7.1.3 GetPlayerBShipPlacement()

```
const std::map<std::pair<Coordinates, Coordinates>, int>& GameRecorder::GetPlayerBShip↩
Placement ( ) const
```

Returns the Player A's ship placement

Returns

a map of a pairs of coordinates (bow, stern) and a int (width)

3.7.1.4 GetStartingPlayer()

```
int GameRecorder::GetStartingPlayer ( ) const
```

Returns which player start the game.

Returns

1: Player A, 2: Player B

3.7.1.5 LoadGameFromLog()

```
void GameRecorder::LoadGameFromLog (
            const std::string & log_path )
```

Given a plain text file path `log_path`, this method parses it's content and reads the starting player, player A's ship placement, player B's ship placement and the moves played.

Parameters

log_path	The plain text file path from the executable root.

3.7.1.6 PersistGameToLog()

```
void GameRecorder::PersistGameToLog ( )
```

Persists the contents of the internal variables to a plain text file. The generated file is named GAMEMODE_game↩_TIMESTAMP.txt GAMEMODE is either 'cc' or 'pc' based on `game_mode_`, 'cc' is computer vs computer, 'pc' is player vs computer TIMESTAMP is a timestamp in the format YYYYMMDDHHMMSS . See `Utility::Get↩Timestamp()`. The directory where the file is saved is specified in `GameRecorder::LOG_PATH`

3.7.1.7 RecordMove()

```
void GameRecorder::RecordMove (
            std::pair< Coordinates, Coordinates > move )
```

Records a move to the moves vector. There is no need to use `SetIsPlayerATurn` each time before recording a new move, the players takes turn making one move a time.

Parameters

move	the move to record

3.7.1.8 RecordShipPlacement()

```
void GameRecorder::RecordShipPlacement (
            Coordinates bow,
            Coordinates stern,
            int ship_with )
```

Records a ship placement to either `player_a_ship_placement_` or `player_a_ship_placement↩_` based on the value of player_a_turn. To record a ship placement to `player_a_ship_placement_` set player_a_turn to true by calling `SetIsPlayerATurn()`, or set it to false to record the move to the other player.

Parameters

bow	The bow of the ship placed
stern	The stern of the ship placed
ship_with	The ship's width

3.7.1.9 SetIsPlayerATurn()

```
void GameRecorder::SetIsPlayerATurn (
            bool is_player_a_turn )
```

Changes the turn

Parameters

is_player_a_turn	if true it's the player A's turn, false it's the player B's turn. See `GameRecord::RecordShipPlacement()` for it's usage.

3.7.1.10 SetStartingPlayer()

```
void GameRecorder::SetStartingPlayer (
            int starting_player )
```

Sets the starting player

Parameters

starting_player	1: Player A starts, 2: Player B starts.

3.7.1.11 ToString()

```
std::string GameRecorder::ToString ( ) const
```

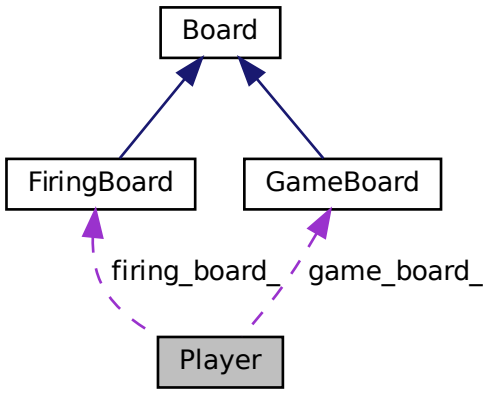
Returns the string representation of this object. This method to generate the contents of the log file to persist. The generated string follows the pattern: STARTING_PLAYER\n\nPLAYER_A_SHIP_PLACEMENT\n\nPLAYER_B_↩ SHIP_PLACEMENT\n\nMOVES

The documentation for this class was generated from the following file:

- GameRecorder.h

3.8 Player Class Reference

Collaboration diagram for Player:



Public Member Functions

- **Player** (std::string name, bool is_human=false)
- bool PlaceShip (Coordinates bow, Coordinates stern, Ship &ship)
- bool PlayMove (Coordinates origin, Coordinates target)
- void PlaceShipsRandomly (GameRecorder &game_recorder)
- std::string ToString () const

  Returns a string containing a string representation of GameBoard and FiringBoard side by side.
- std::pair< Coordinates, Coordinates > GetRandomShipPlacement (int ship_width, bool is_horizontal) const
- std::pair< Coordinates, Coordinates > GenerateRandomMove ()
- std::shared_ptr< Ship > GetShipAt (Coordinates location)

  This is a helper method to call GameBoard.GetShipAt without exposing the object GameBoard.
- bool ReceiveAttack (Coordinates coordinates)

  This is a helper method to call GameBoard.ReceiveAttack without exposing the object GameBoard.
- void MarkAttack (Coordinates coordinates, bool b)

  This is a helper method to call FiringBoard.ReceiveAttack without exposing the object FiringBoard.
- bool MoveShip (Coordinates origin, Coordinates target)

  This is a helper method to call GameBoard.PlaceShip without exposing the object GameBoard.
- const std::string & GetName () const
- Coordinates GetNextTarget ()
- void AddNextTargets (Coordinates coordinates)
- OccupationType InquireState (Coordinates target)
- void AddSubmarineSightings (const std::map< Coordinates, OccupationType > &scan_from_submarine)
- void ClearSubmarineSightings ()

  This a helper method to avoid exposing the internal FiringBoard to the caller. It calls the ClearSubmarineSightings() method of FiringBoard.
- void ClearSuccessfulHits ()

  This a helper method to avoid exposing the internal FiringBoard to the caller. It calls the ClearSuccessfulHits() method of FiringBoard.

- void ClearUnsuccessfulHits ()

  This a helper method to avoid exposing the internal FiringBoard to the caller. It calls the ClearUnsuccessfulHits() method of FiringBoard.
- void ClearAllHits ()

  This a helper method to avoid exposing the internal FiringBoard to the caller. It calls the ClearAllHits() method of FiringBoard.
- void AddNextTargets (const std::map< Coordinates, OccupationType > &submarine_sightings)
- bool HasLost ()

  A player loses when all of his Battleships are destroyed.
- bool **IsHuman** () const
- std::string GameBoardToString () const
- void RepairShipAt (Coordinates coordinates)

  Clears all the hits received by the ship at.
- std::string GetAttackMessage (bool is_successful)

  Helper methods to print messages during the game.
- std::pair< Coordinates, Coordinates > **GetRandomShipPlacement** (int ship_width) const

Private Attributes

- std::string **name_**
- GameBoard **game_board_**
- FiringBoard **firing_board_**
- bool **is_human_**
- std::set< Coordinates > **next_targets_**

3.8.1 Member Function Documentation

3.8.1.1 AddNextTargets() [1/2]

```
void Player::AddNextTargets (
            const std::map< Coordinates, OccupationType > & submarine_sightings )
```

This methods add all coordinates present in `submarine_sightings` to the set of next targets. The idea behind this is that every time a submarine detects an enemy ship at some coordinates there is a high chance that the ship has not moved yet by the time GetNextTarget() is called, so it makes sense to try to shoot there.

3.8.1.2 AddNextTargets() [2/2]

```
void Player::AddNextTargets (
            Coordinates coordinates )
```

Adds the Coordinates.GetAdjacentStartCoordinates to the set of next targets. The idea behind this is that every time we hit an enemy ship there is a high chance that the remaining parts of that ship are either above, below, right or left from the hit coordinate.

Parameters

coordinates	

3.8.1.3 AddSubmarineSightings()

```
void Player::AddSubmarineSightings (
            const std::map< Coordinates, OccupationType > & scan_from_submarine )
```

This is a helper method to avoid exposing the internal FiringBoard to the caller. It calls FiringBoard.AddSubmarineSightings

3.8.1.4 GameBoardToString()

```
std::string Player::GameBoardToString ( ) const  [inline]
```

A helper method that returns the string representation of the internal GameBoard, obtained by calling the Game↩ Board.ToString(). It is used during the placement of the various ships.

Returns

a string representation of the internal GameBoard.

3.8.1.5 GenerateRandomMove()

```
std::pair<Coordinates, Coordinates> Player::GenerateRandomMove ( )
```

Generates a random move composed of a pair of coordinates (origin, target). Origin is the central cell of one of the ships placed in GameBoard. Target is generated randomly if the ship at origin is not a Battleship, or using GetNextTarget() if it's a battleship.

Returns

3.8.1.6 GetName()

```
const std::string& Player::GetName ( ) const
```

Returns the name of the current player.

Returns

3.8.1.7 GetNextTarget()

```
Coordinates Player::GetNextTarget ( )
```

Returns the coordinates for the next target for the Battleship to shoot at. The next target is picked between a set of next targets that gets updated each time a successful hit is made with the adjacent coordinates. The set of next targets gets also updated each a time a submarine detects enemy ships. If the set is empty, it returns a random target.

3.8.1.8 GetRandomShipPlacement()

```
std::pair<Coordinates, Coordinates> Player::GetRandomShipPlacement (
            int ship_width,
            bool is_horizontal ) const
```

Generates a random ship placement for a ship with width `ship_width`

Parameters

ship_width	
true	if the placement should be horizontal, false otherwise

Returns

Pair of coordinates (bow, stern)

3.8.1.9 InquireState()

```
OccupationType Player::InquireState (
            Coordinates target )
```

This is a helper method to avoid exposing the internal GameBoard to the caller (usually an enemy submarine).

Parameters

coordinates	

Returns

It returns the occupation type of the cell `target`,`OCCUPIED` at `target` a part of a ship is present but not hit, `HIT` if a part of a ship is present and hit or `EMPTY` if nothing is present there.

3.8.1.10 PlaceShip()

```
bool Player::PlaceShip (
            Coordinates bow,
```

```
            Coordinates stern,
            Ship & ship )
```

Positions a ship given the bow and stern coordinates

Parameters

bow	The coordinates of the bow
stern	The coordinates of the stern
ship	The ship to position.

Returns

true if the ship has been placed successfully

false if the ship has not been placed successfully

3.8.1.11 PlaceShipsRandomly()

```
void Player::PlaceShipsRandomly (
            GameRecorder & game_recorder )
```

Places the ships randomly on on the board.

Parameters

game_recorder	The game recorder used to record the placements of the ships.

3.8.1.12 PlayMove()

```
bool Player::PlayMove (
            Coordinates origin,
            Coordinates target )
```

Makes a game move.

Parameters

origin	The central coordinates of the ship .
target	The target coordinates to hit if the ship is a Battleship, or where to move and search/cover in the case of other ships.

Returns

true if the move could be played

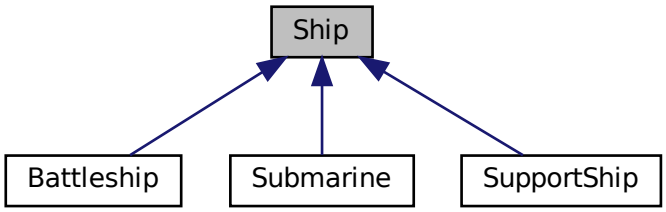
false otherwise.

The documentation for this class was generated from the following file:

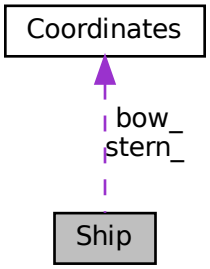
- Player.h

3.9 Ship Class Reference

Inheritance diagram for Ship:



Collaboration diagram for Ship:



Public Member Functions

- Ship (char icon, int width)
- Ship (char icon, int width, int icon_color)
- const std::set< int > & GetHitLocationsOffset () const
- void **SetHitLocationsOffset** (const std::set< int > &hit_locations_offset)
- const std::string & **GetIcon** () const
- void **SetIcon** (const std::string &icon)
- const std::string & **GetHitIcon** () const
- void **SetHitIcon** (const std::string &hit_icon)

- bool IsSunk ()

  Check whether a ship has been sunk. A ship is considered sunk when the number of hits received is >= it's width.
- void HitLocation (Coordinates location)
- int **GetWidth** () const
- void **SetWidth** (int width)
- ShipType **GetShipType** () const
- const Coordinates & **GetBow** () const
- void **SetBow** (const Coordinates &bow)
- const Coordinates & **GetStern** () const
- void **SetStern** (const Coordinates &stern)
- std::string ToString (Coordinates location)
- Coordinates GetShipCenter ()
- std::vector< Coordinates > GetLocations ()
- bool IsHorizontal () const
- bool IsHit (Coordinates target)
- void Repair ()

  Clears all hits received by this ship restoring the health.

Protected Attributes

- std::string icon_

  Icon of this ship, this is the icon that gets returned by the `ToString()` method.
- int icon_color_ = 2

  This is the main color of the icon of the ship.
- std::string hit_icon_

  When a unit of a ship is hit this is the icon that get returned by the ToString() method.
- int hit_icon_color_ = 1

  The color of a hit unit's icon.
- int width_

  The width of this ship.
- ShipType ship_type_ = UNSET

  The ship type.
- Coordinates **bow_**
- Coordinates **stern_**
- std::set< int > hit_locations_offset_

3.9.1 Constructor & Destructor Documentation

3.9.1.1 Ship() [1/2]

```
Ship::Ship (
            char icon,
            int width )  [inline]
```

Parameters

icon	The icon of the ship
width	The width of the ship

3.9.1.2 Ship() [2/2]

```
Ship::Ship (
            char icon,
            int width,
            int icon_color )  [inline]
```

Parameters

icon	The icon of the ship
width	The width of the ship
icon_color	The color of the icon of the ship

3.9.2 Member Function Documentation

3.9.2.1 GetHitLocationsOffset()

```
const std::set<int>& Ship::GetHitLocationsOffset ( ) const
```

Returns the offsets from the bow that have been hit in this ship.

Returns

A set of the offsets hit in this ship.

3.9.2.2 GetLocations()

```
std::vector<Coordinates> Ship::GetLocations ( )
```

Uses the Coordinates::GetCoordinateBetween(bow, stern) to generate all the location occupied by this ship.

Returns

The a vector of Coordinates representing the locations where units of this ship are placed.

3.9.2.3 GetShipCenter()

```
Coordinates Ship::GetShipCenter ( )
```

Returns the center of this ship. The center is calculated as the middle unit between bow and stern.

Returns

3.9.2.4 HitLocation()

```
void Ship::HitLocation (
            Coordinates location )
```

If `location` is part of this ship, the unit of this ship in it get marked as hit.

Parameters

location	

3.9.2.5 IsHit()

```
bool Ship::IsHit (
            Coordinates target )
```

Checks whether the unit of this ship contained at `target` has been hit or not.

Parameters

target	

Returns

true: the unit at `target` has been hit

false: the unit at `target` has not been hit.

3.9.2.6 IsHorizontal()

```
bool Ship::IsHorizontal ( ) const
```

Returns

true: if the ship is horizontal

false: if the ship is vertical

3.9.2.7 ToString()

```
std::string Ship::ToString (
            Coordinates location )
```

Returns the icon of this ship based on the location provided. If `location` contains a unit from this ship, and that unit is hit, it returns hit_icon Otherwise it returns icon.

Parameters

location	The location where a unit of ship is present.

3.9.3 Member Data Documentation

3.9.3.1 hit_locations_offset_

```
std::set<int> Ship::hit_locations_offset_  [protected]
```

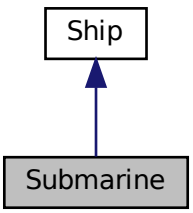
Each time a unit of a ship gets hit, it's offset from `bow_` gets added to this set This set is used to keep track of which units of a ship have been hit during the movement of a ship.

The documentation for this class was generated from the following file:

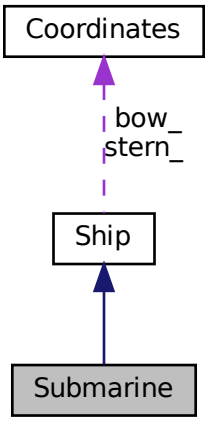
- Ship.h

3.10 Submarine Class Reference

Inheritance diagram for Submarine:



Collaboration diagram for Submarine:



Public Member Functions

- Submarine (Coordinates bow, Coordinates stern)
- Submarine ()

  Creates a ship of type Submarine.

Static Public Member Functions

- static std::map< Coordinates, OccupationType > ScanSurroundings (Player &opponent, Coordinates current_position)

Static Public Attributes

- static const int **DEFAULT_SIZE** = 1

Additional Inherited Members

3.10.1 Constructor & Destructor Documentation

3.10.1.1 Submarine()

```
Submarine::Submarine (
            Coordinates bow,
            Coordinates stern )  [inline]
```

Creates a Ship of type Submarine starting from the bow and stern. If the bow to stern distance does not match the vessel size, throw a std::invalid_argument

Exceptions

std::invali_argument	

Parameters

bow	The bow of the ship
stern	The stern of the ship

3.10.2 Member Function Documentation

3.10.2.1 ScanSurroundings()

```
static std::map<Coordinates, OccupationType> Submarine::ScanSurroundings (
            Player & opponent,
            Coordinates current_position )  [static]
```

Scans the the cells of the opponent's GameBoard the are in a 5 by 5 matrix with the `current_position` at the center. First it generates the needed surrounding coordinates and the users the `InquireState()` method from `Player` to get the state of the cell.

Returns

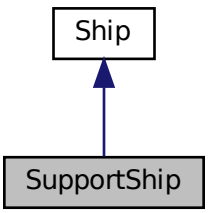
a map of the scanned opponent's GameBoard.

The documentation for this class was generated from the following file:

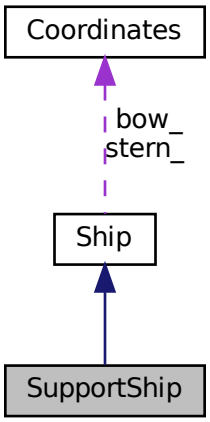
- Submarine.h

3.11 SupportShip Class Reference

Inheritance diagram for SupportShip:



Collaboration diagram for SupportShip:



Public Member Functions

- SupportShip (Coordinates bow, Coordinates stern)
- SupportShip ()

Static Public Member Functions

- static void RepairNearbyShips (Player &player, Coordinates current_position)

  Repairs all the `player's` hips in a range of 3 from the current position.

Static Public Attributes

- static const int **DEFAULT_SIZE** = 3

Additional Inherited Members

3.11.1 Constructor & Destructor Documentation

3.11.1.1 SupportShip() [1/2]

```
SupportShip::SupportShip (
            Coordinates bow,
            Coordinates stern )  [inline]
```

Creates a Ship of type SupportShip starting from the bow and stern. If the bow to stern distance does not match the vessel size, throw a std::invalid_argument

Exceptions

std::invali_argument	

Parameters

bow	The bow of the ship
stern	The stern of the ship

3.11.1.2 SupportShip() [2/2]

```
SupportShip::SupportShip ( )  [inline]
```

Creates a ship of type SupportShip.

Exceptions

std::invali_argument	

Parameters

bow	The bow of the ship
stern	The stern of the ship

The documentation for this class was generated from the following file:

- SupportShip.h

3.12 UserCommand Class Reference

Public Member Functions

- UserCommand (const std::pair< Coordinates, Coordinates > &move, CommandType command_type)
- UserCommand (CommandType command_type)
- UserCommand (const std::string &command)
- bool IsSpecial () const
- const std::pair< Coordinates, Coordinates > & **GetMove** () const
- void **SetMove** (const std::pair< Coordinates, Coordinates > &move)
- CommandType **GetCommandType** () const
- void **SetCommandType** (CommandType command_type)

Static Public Member Functions

- static bool IsSpecial (const std::string &command)

Private Attributes

- std::pair< Coordinates, Coordinates > **move_**
- CommandType **command_type_**

Static Private Attributes

- static const std::map< std::string, CommandType > **SPECIAL_COMMANDS_**

3.12.1 Constructor & Destructor Documentation

3.12.1.1 UserCommand() [1/3]

```
UserCommand::UserCommand (
            const std::pair< Coordinates, Coordinates > & move,
            CommandType command_type )
```

Creates a UserCommand from a pair of Coordinates (origin, target) and CommandType

Parameters

move	A pair of Coordinates representing the origin and the target of a move.
command_type	The CommandType

3.12.1.2 UserCommand() [2/3]

```
UserCommand::UserCommand (
            CommandType command_type )  [explicit]
```

Create a UserCommand from a CommandType

Parameters

command_type	

3.12.1.3 UserCommand() [3/3]

```
UserCommand::UserCommand (
            const std::string & command )  [explicit]
```

Create a UserCommand from a string representing the user's command. If the `command` is one of te special commands (`UserCommand::SPECIAL_COMMANDS_`) it creates a UserCommand simply by setting the command_type, otherwise it tries to parse the string `command` and extract a pair of coordinates from it. Uses `Coordinates(std::string)` constructor for the parsing.

Exceptions

std::invalid_argument	if the provided string is not a valid command.

3.12.2 Member Function Documentation

3.12.2.1 IsSpecial() [1/2]

```
bool UserCommand::IsSpecial ( ) const  [inline]
```

Checks whether this is a special command or not.

Returns

true if it's a special command

false if it's not

3.12.2.2 IsSpecial() [2/2]

```
static bool UserCommand::IsSpecial (
            const std::string & command )  [static]
```

Checks where the string passed is a special command.

Parameters

command	

Returns

true if `command` is one of the special commands defined in `UserCommand::SPECIAL_COMMANDS_`

false if it's not a special command.

The documentation for this class was generated from the following file:

- UserCommand.h

The problem with being faster than light is that you can only live in darkness

Index