1. **Objective of the Project:**

*The objective of the project is to design and develop a ‘Battleships’ game on the computer using the Java software program.*

1. **Statement of Requirements**

We have been assigned the task to use the widely used programming language ‘Java’ to design and create an adaptation of the classic pen-and-paper game "Battleships". The ultimate aim of this project is to develop a game that is reliable, versatile, user-friendly and most important, addictive to entice the player/user to continuously return to repeatedly play the game. In order to achieve this, the game must be entertaining and be able to create excitement and thrill. The target group for this ‘Battleship’ game shall be children between the ages of 5 years to 12 years old using the English language as the medium of instruction.

The expected inputs for the program and the expected outputs from these are shown in the table below:

|  |  |
| --- | --- |
| INPUTS | EXPECTED OUTPUTS |
| Numerical coordinates. | Shot fired based on coordinates inputted by the user. |
| Position of the ship; vertical or horizontal | Displays the ship vertically or horizontally on the grid. |
| Ship location. | Display the ships on the grid. |
| Menu option | Loads different method of gameplay; new game, loads saved game or loads gameplay of user vs the computer. |

Initially, a basic version of Battleships shall be created where only the user gets to "shoot" and the placement of the ships is randomly generated by the AI. The "shooting" would involve guessing the coordinate location of the enemy ships and then inputting it into the program until there are no more ship left. Eventually, the program will be extended to include the following options:

1. To allow the human user to be able to generate his/her own ship grid and for the AI to be able to shoot them.
2. The human and the AI users will shoot at each other's ships and the winner will be determined by the number of ships location hit.
3. **Requirements**

The Project brief has clearly spelt out the required specifications in the following manner:

1. **Functional Requirements:**
   1. A "Menu" page with displays the different game modes available, this will include:

* A new game option where the user will fire torpedoes on a board which has ships randomly in it, the user will be allowed a limited number of torpedoes and will have to try to sink all the ships with the set amount.
* A load game option which loads a previously saved new game.
* A game mode to compete with the program where the user will be allowed to place ships in a grid for the program to fire their torpedoes on and a grid will be created for the user to do the same but in this case the program will place the ships on that board. Both user and the program will try to destroy all the ships or hit as many parts of the ships to determine the winner.
  1. A graphical display window that will show the current state of the board, i.e:
* The board will display whether a hit has been made or a miss on the board.
* The board will display the types of ships which has been placed on the board for the versus computer mode.
* The board will display the positions hit by the program in the ‘versus computer’ mode and the remaining torpedoes left.
  1. A "Fire" option :
* Users shall fire a shot by selecting a grid position.
* The number of remaining shots left for the user and the program in different game modes (remaining torpedoes left).
* The grid shall be updated after every shot fired.
  1. A "Save Game" option to save the entire unfinished game to a text file so that it can be continued later.
  2. A "Load Game" option to load a previously saved game so that the user can continue playing.
  3. Randomly generated positions of the enemy ships.
  4. Labelling the grids numerically for both the horizontal x-axis and the vertical y-axis as the input to fire at specific grids will be based on using coordinates with two number eg. 1 2, 3 7, etc.
  5. The ships may only be generated in vertical or horizontal position.
  6. The game should keep track of game data:
* Number of torpedoes left or number of shots left
* Whether the shot fired is a hit or a miss
  1. The game should check inputs entered are correct and they are handled appropriately in the following manner:
* The coordinates entered are within the grid boundary. The program will request for another set of coordinates if an erroneous input was entered.
* The correct type of input is entered; no character or weird character are entered and any presence of such shall cause the program to request for inputs to be re-entered.
* Ensure that the coordinates of current ship entered is not overlapping with the previous ones.

The Project brief has also included design specification pertaining to the fleet size as shown below:

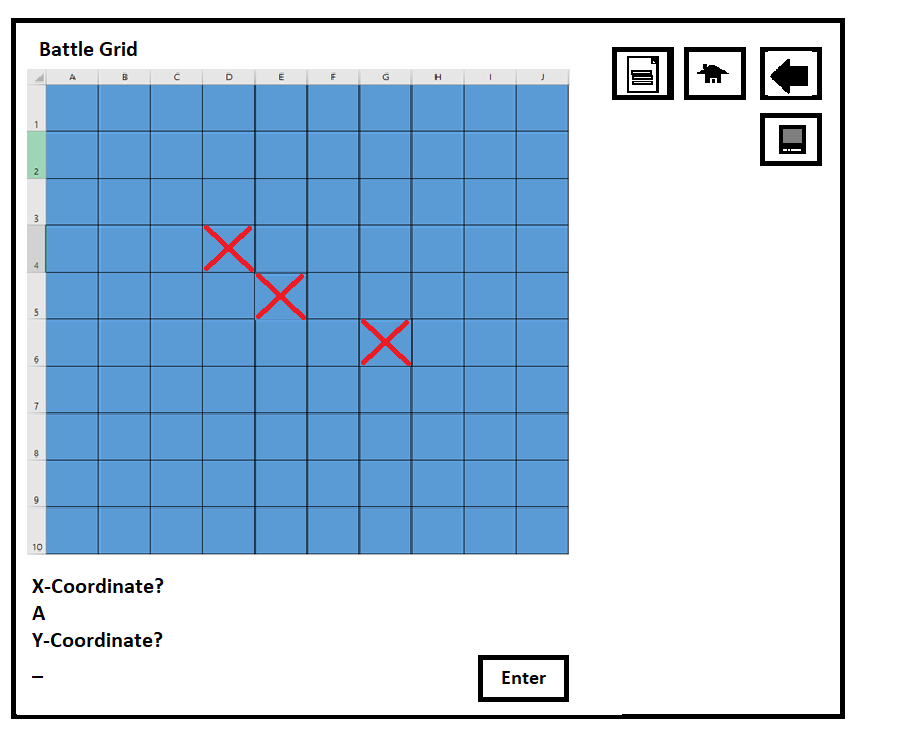
|  |  |  |  |
| --- | --- | --- | --- |
| Type of fleet | Number of units | Grid spaces occupied | Alphabet to be used to denote the type of fleet |
| Battleships | 1 | 4 | ‘B’ |
| Cruisers | 2 | 3 | ‘C’ |
| Destroyers | 3 | 2 | ‘D’ |
| Submarines | 3 | 1 | ‘S’ |

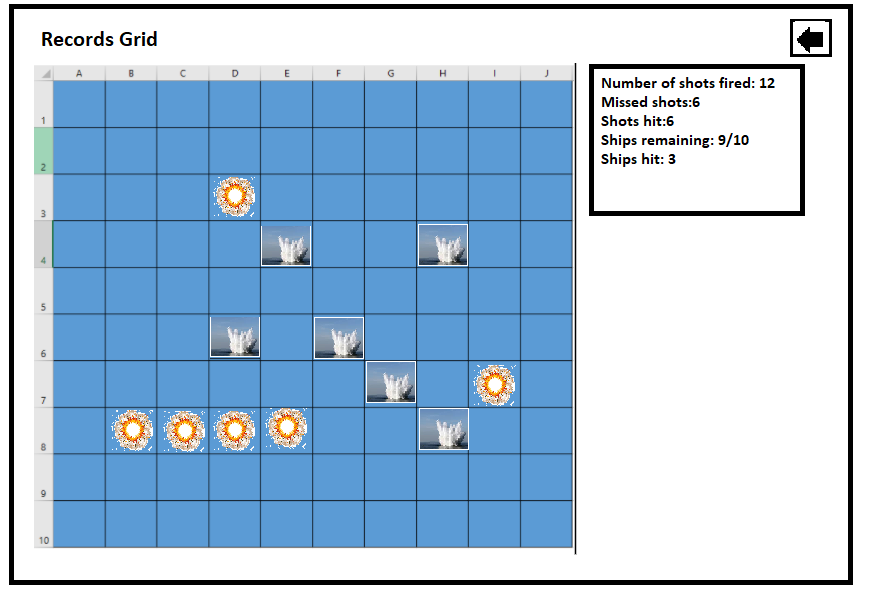
1. **Non-functional Requirements**

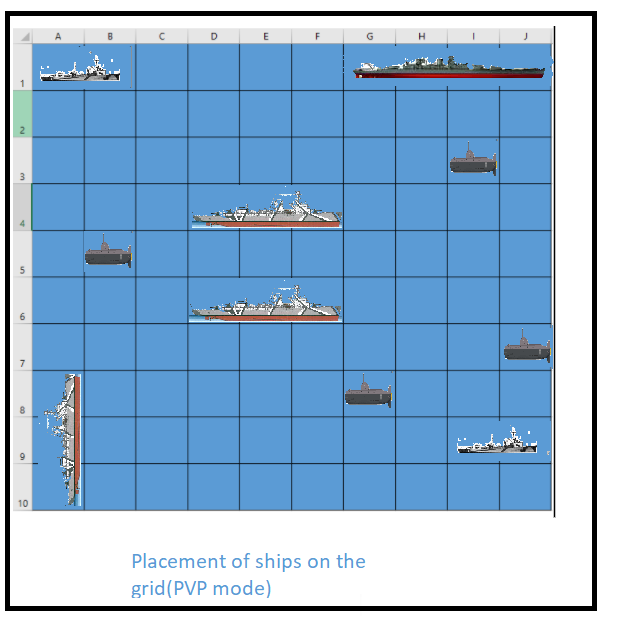
We recognized the importance of non-functional requirements which are as follows:

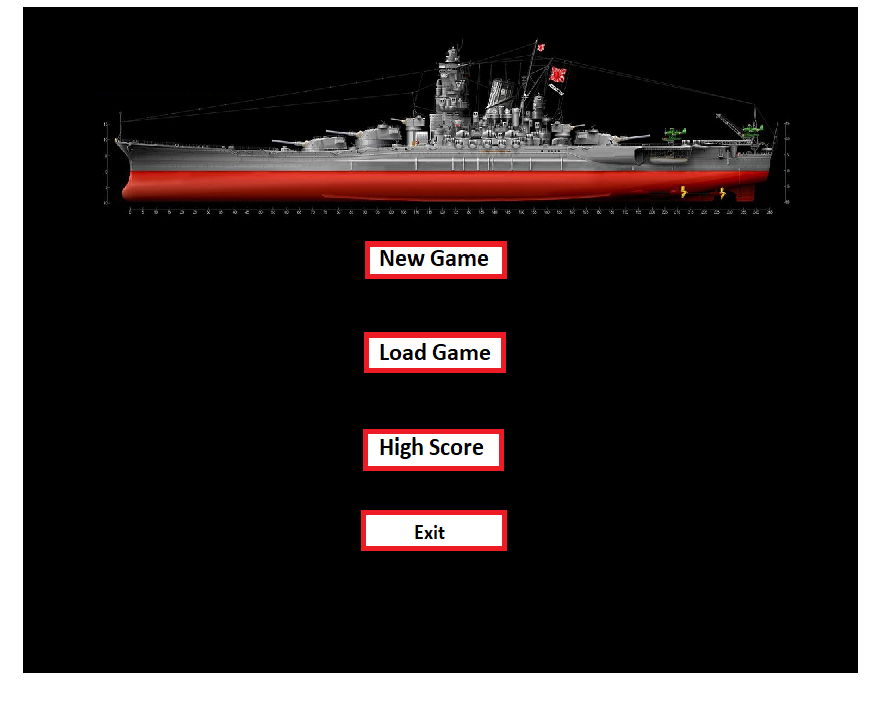
* 1. Clean Code- the code must be written using appropriate structures such as comments and indents.
  2. Aesthetics - the user-interface must be aesthetically pleasing otherwise the user will be deterred from playing the game.
  3. Reliable - the program will run within an acceptable time frame and will not crash.
  4. User-friendly - a simplistic and easily understandable user interface.
  5. G-rated - our target audience is between the ages of 5 and 12 therefore appropriate language and visuals must be used.
  6. To be implemented in Windows 10.
  7. To include visual graphics and use of appropriate colour schemes to help users who have colour blindness.
  8. Having a graphical user interface which will give users a more aesthetically pleasing game.

1. **User Interface Design**









1. **Use Cases**

|  |  |  |
| --- | --- | --- |
| New Game | | |
|  | User | Selects new game option |
|  | System | Creates board of fixed size. |
|  | System | Populates the board with ships which are place randomly. |
|  | System | Displays the board to the user. |
|  | User | Inputs coordinates. |
|  | System | Determines if coordinates entered are accurate; within the grid boundaries and the correct type. |
|  | System | Request for new coordinates to be entered in the event of an erroneous input. |
|  | System | Displays updated board with details of a hit or miss on the board. |
|  | System | Updates the details of remaining shots left. |
|  | System | Prompts user if they want to save the game. |
|  | User | Inputs decision whether to save the game. |
|  | System | Displays a successfully saved message to the user if game has been saved. |
|  | System | Determines that there are remaining shots left for the user to fire. |
|  | System | Prompts for more coordinates or ends game depending on whether there are shots remaining or all the ships have been hit. |
|  | System | Displays message to user when the game has ended and whether they lost or won. |

|  |  |  |
| --- | --- | --- |
| Save Game | | |
|  | User | Selects Save Game during the game. |
|  | System | Stores the grid of the game with details of where each ship has been placed. |
|  | System | Stores the game state into a text file with a predetermined name and directory. |
|  | System | Informs the user that the game has been saved successfully. |
|  | System | Exits the game. |

|  |  |  |
| --- | --- | --- |
| Load Game | | |
|  | User | Selects Load Game option |
|  | System | Loads each line of the text file into an array list. |
|  | System | Places each character in the text file into the board and displays it. |
|  | System | Updates the remaining shots available . |
|  | System | Displays board with details of shots missed, successful hits and remaining shots. |

|  |  |  |
| --- | --- | --- |
| VS Computer | | |
|  | User | Select vs computer option |
|  | System | Prompts user for coordinates within a certain range. |
|  | User | Enter coordinates. |
|  | System | Authenticates the inputs made by the user. |
|  | System | Request the user for new coordinates if the inputs are erroneous. |
|  | System | Prompts user for position in which the ship is to be placed; horizontally or vertically. |
|  | System | Places ship at the coordinate on the board if the position entered is valid and the ship size fits the board. |
|  | System | Displays the grid with the ship placed in the position entered. |
|  | System | Prompts user for more ships coordinates and position to be entered. |
|  | User | Enter new ship coordinates and position of each ship. |
|  | System | Updates grid with each subsequent ship if the coordinates entered are entered correctly. |
|  | System | Creates |
|  | System | Creates enemy board of fixed size. |
|  | System | Populates enemy board with enemy ships which are place randomly. |
|  | System | Displays the board to the user. |
|  | System | Prompts user for coordinates to be fired upon |
|  | User | Inputs coordinates. |
|  | System | Determines if coordinates entered are accurate; within the grid boundaries and the correct type. |
|  | System | Request for new coordinates to be entered in the event of an erroneous input. |
|  | System | Displays updated board with details of a hit or miss on the board. |
|  | System | Increments users score whenever a hit has been made. |
|  | System | Updates the details of remaining shots left. |
|  | System | Prompts for more coordinates depending on whether there are shots remaining or all the ships have been hit. |
|  | System | Displays message to user when all shot has been fired or all ships have been hit. |
|  | System | Program fires a torpedo randomly onto coordinates of the user board. |
|  | System | Displays the updated board with details of hit or miss. |
|  | System | Displays details of remaining shots and coordinates in which was fired upon. |
|  | System | Increments programs score if a hit was made. |
|  | System | Program continues to fire a torpedo randomly depending on whether there are any shots remaining or all the ships have been hit. |
|  | System | Compares the user score and the programs score. |
|  | System | Display a text of the winner. |

1. Classes

##### Candidate Classes

|  |  |  |
| --- | --- | --- |
| Candidate Classes | Accept / Reject | Reason for rejection or acceptance |
| Ship | Accept |  |
| Ships | Reject | No plurals |
| SetUpBoard | Reject | Rejected as board is made of cells and SetUpBoard  is a method |
| MButton | Reject | Not descriptive enough |
| MenuButton | Reject | There are more button types |
| Buttons | Reject | Plural |
| Button | Accept |  |
| GameManager | Accept | Needs to manage the game |
| MainMenu | Accept |  |
| Board | Accept |  |
| Player | Reject | GameManager will have methods for players |
| Graphical Display Window | Accept |  |
| files | Accept |  |
| LoadGame | Reject | Files classes will have a load game method |
| SaveGame | Reject | Files classes will have a load game method |

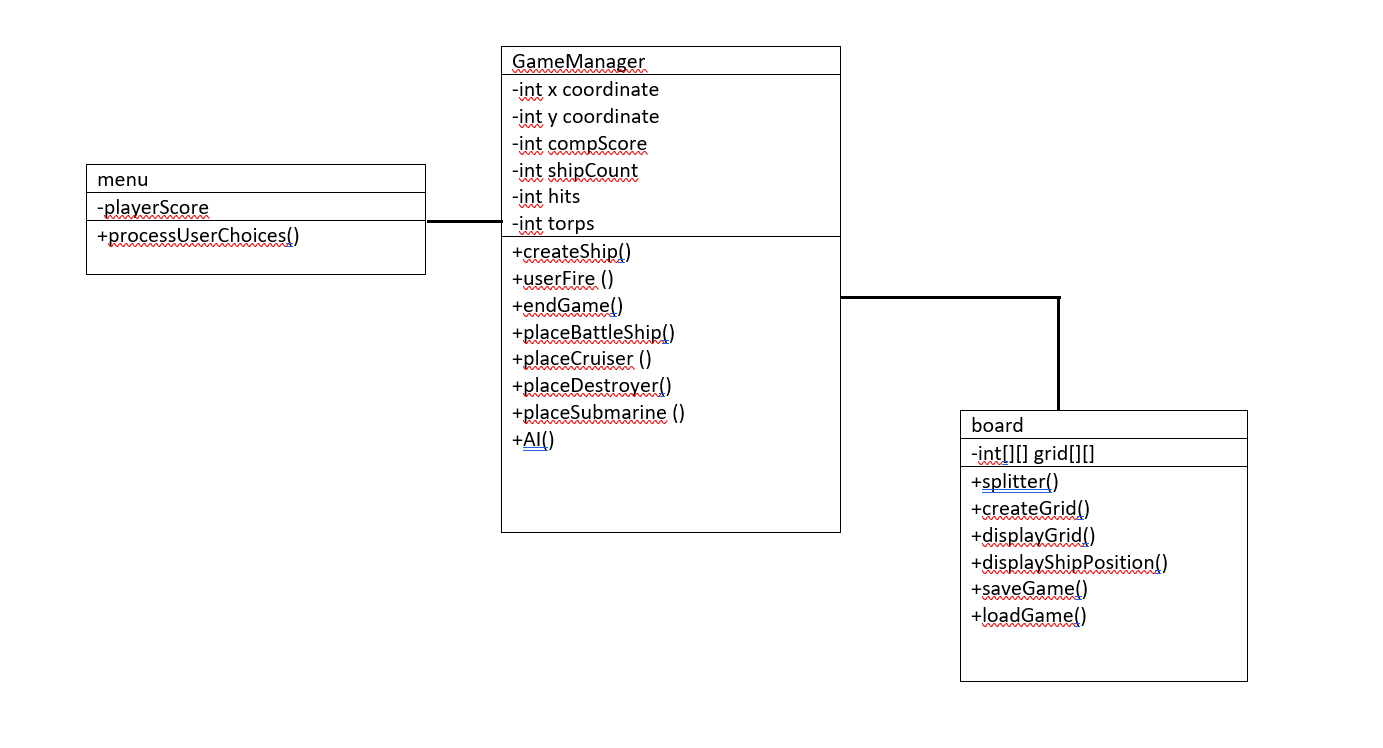
##### Class Descriptions including Responsibilities, Fields and Methods

|  |
| --- |
| menu |
| -playerScore |
| +processUserChoices() |

|  |
| --- |
| board |
| -int[][] grid[][] |
| +splitter()  +createGrid()  +displayGrid()  +displayShipPosition()  +saveGame()  +loadGame() |

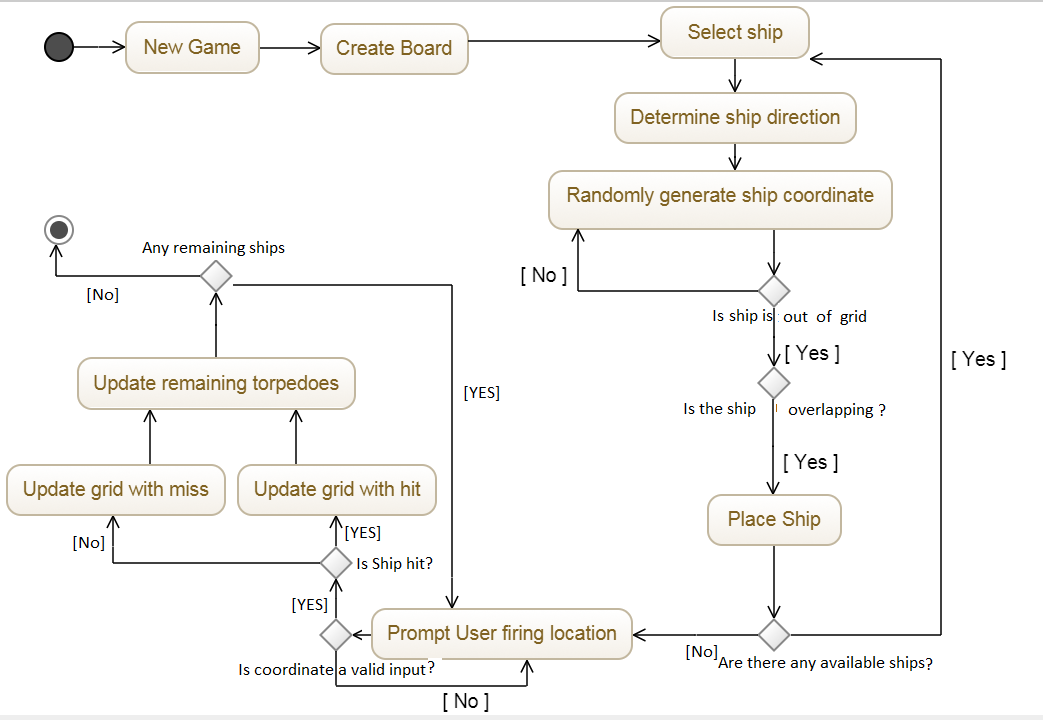
|  |
| --- |
| GameManager |
| -int x coordinate  -int y coordinate  -int compScore  -int shipCount  -int hits  -int torps |
| +createShip()  +userFire ()  +endGame()  +placeBattleShip()  +placeCruiser ()  +placeDestroyer()  +placeSubmarine ()  +AI() |

##### Class Diagram

**

1. **Activity Diagrams / Pseudocode**

Activity Diagram for the base game



1. **Descriptions of your classes and methods**

|  |  |  |  |
| --- | --- | --- | --- |
| Class | Description | Method | Description |
| 1. menu | Class is used to hold all methods of menu which includes main menu choices of the game modes and method to exit the game. | 1. +processUserChoices() | 1. Selects users game option and to determine which game method to load. |
| 1. board | Class is an object reference of the game board which contains variables such as the fixed grid size and methods to display board, create board, etc. | * 1. +splitter()   2. +createGrid()   3. +displayGrid()   4. +displayShipPosition() | 1. Creates a line that separates the board for a more visual appearance   ii. Creates the grid size  iii. Displays the grid while hiding the ships  iv.Displays the grid with the ships |
| 1. GameManager | Class contains methods in relation to the game play; it determines whether the game has ended, firing of the torpedoes, the program ai for the versus computer mode,etc. | * 1. createShip()   II. userFire ()  III.endGame()  IV.placeBattleShip()  V.placeCruiser ()  VI.placeDestroyer()  VII.placeSubmarine ()  VIII.AI() | i.Creates all the ship methods  ii. Updates whether the fired torpedo is a miss or hit.  iii. Displays the end game message.  iv. Places the Battleship in the versus computer game mode.  v. Places the Cruiser in the versus computer game mode.  vi. Places the Destroyer in the versus computer game mode.  vii. Places the Submarine in the versus computer game mode.  viii. Randomly selects places in the user grid to be fired upon. |

*References and credits*

# Dr. Dan Saranga. 2016. Light Cruiser

Image from: <https://www.the-blueprints.com/blueprints/ships/cruisers-germany/65864/view/dkm_nurnberg_1941_%5Blight_cruiser%5D/> [Accessed: 26th of September]

Tatjana Rittner. Explosion  
Image from: <https://www.123rf.com/clipart-vector/explosion_fire.html?mediapopup=11937093>[Accessed:26th of September]

S.Baldkich 2008. Ship  
Image from: <http://statehistory.ru/books/A--M--Vasilev---A--B--Morin_Superlinkory-Stalina--Sovetskiy-Soyuz--Kronshtadt--Stalingrad/14> [Accessed: 26th of September]

Pascal Pochard-Casabianca/AFP/Getty Images. 2015. Water Explosion  
Image from: <http://science.howstuffworks.com/explosion-land-water1.htm> [Accessed: 26th of September]

Dr. Dan Saranga. 2006. Destroyer  
Image from: <https://www.the-blueprints.com/blueprints/ships/destroyers-germany/14786/view/dkm_t5_%28destroyer%29_%281938%29/> [Accessed: 26th of September]

# PaintFan08. 2010. Submarine

Image from: <https://paintfan08.deviantart.com/art/Type-XXX-Submarine-160647436> [Accessed: 26th of September]

Bsyamato. 2001. Battleship Yamato  
Image from: <https://commons.wikimedia.org/wiki/Japanese_battleship_Yamato#/media/File:Battleship_Yamato_on_Apr_1945.jpg> [Accessed: 26th of September]