Operating Systems - 2021

Project 1 – Clash of Ships

Implemented in [Java] using [IntelliJ IDEA]

Alireza Khorshidi (9735413)

Link to the project on GitHub
Click

Table of Contents

Introduction

This documentation contains information on an implementation of the first project of the course *Operating Systems* in Spring 2021.

The goal is to practice declaring and running multiple threads that run simultaneously yet bound together via some shared information. Java is used to implement the following solution.

More details on the project are specified in the assignment file & the following Project overview section.

Project overview

The goal is to simulate a number of ships attacking each other. There are no teams, meaning that all ships are enemies. The game ends whenever there are either no ships or only one ship left alive[1]. The last standing ship is considered to be the winner.

At the beginning, the user specifies the number of Countries at war (number of ships) and selects to run the game in either the PRESENT setting (each ship dies with one bullet) or the FUTURE setting (each ship dies with 2 bullets[2]).

Each battle is simulated as an object of the *Game* class. It is tasked with processing the game, tracking ships' status, assigning turns to them, and providing a valid target for each ship to shoot at. Therefore, it contains multiple instances of the *Ship* class.

Each ship has a dedicated thread (By inheriting the built-in *Thread* class) and can attack other ships, however, none of the ships' threads stop unless the game ends or the ship dies[3] (A ship dies when it runs out of health).

Each ship will take a random number of seconds between 1 & 3 to reload and get ready to shoot. A simple state machine is dedicated to each ship to help with tracking the ships & assigning turns to them.

The *Main* class is tasked to receive input data from terminal and to run the game as an instance of the *Game* class. The output will be displayed on terminal in real-time.

[1] Under certain circumstances, there will be no ships alive at the end of the game.
[2] The values can further be customized by directly specifying them at instantiation or modifying the code.
[3] As the project rules state, a thread can only stop itself but it should not stop other threads. This has been taken into consideration when implementing the design.

Classes

Main

Start the application from here.

- public void Main(String args[])
  Gets input from the user, initializes the game according to that data, and runs it.

Game

Responsible for processing the game, tracking ships' status, assigning turns to them, and fetching proper targets for them to shoot at.

- private final int numberOfCountries
  The number of countries which is the total number of ships in the game.

- private final List<Ship> allShips
  List of all of the ships in the game.

- private int shipTurn
  An integer index between 0 and numberOfCountries determining which ship has the turn to make a move, that is to shoot.

- private GameType gameType
  Determines the game type. Affects ships' starting health value. The PRESENT setting means that ships die with only one shot, while the FUTURE setting means that they are to be shot twice to die.

- public Game(int numberOfCountries)
  Constructor. Initializes the attributes and sets the gameType field to PRESENT by default.

- public Game(int numberOfCountries, GameType gameType)
  Constructor. Initializes the attributes and sets the gameType field to value of the argument.

- public void initializeGame()
  Creates ships and adds them to allShips according to the gameType. But <u>Does not</u> start their threads.

- public void startGame()
  Starts every ships' thread. Then waits until the game ends. Finally announces the winner.

- public boolean gameHasEnded()
  Returns true if the game has ended, false if not. The game ends when zero or only one ship is alive.

- public Ship getTargetShipForAttacker(Ship attacker)
  returns a proper ship for the attacker to shoot at. A ship is considered valid to be shot at by attacker if it is alive & is not the same as the attacker ship. Returns null if no ships are found.

- private void updateShipTurn()
  Updates the ship turn so that every ship can make a move. If the current ship is not shooting or is dead, then the turn should be given to the next ship.

- public GameType getGameType()
  Returns the gameType.

- public int getShipTurn()
  Return the index of the ship which has the turn.

Ship

A model of every ship. It also is a thread as well. The thread lives as long as the game hasn't ended or the ship is still alive.

- private int id
  The id of the ship. It also is the same as the ship's index in Game.allShips.

- private String name
  The name of the ship. Used for printing to terminal just so it wouldn't be as boring as numbers floating on the screen.

- private int health
  The health of the ship. A ship dies whenever its health reaches zero.

- private int bulletDamage
  The amount of damage the ship deals when attacking another ship.

- private Game parentGame
  A reference to the game which the ship belongs to. The ship learns of the ships' turn and requests targets via this object.

- private ShipState shipState
  The current state of the machine. RELOADING is for when the thread is sleeping indicating that the ship is reloading its canons. WAITING is for when the ship has reloaded but is waiting for its turn to arrive and shoot. SHOOTING is for when the ship is shooting another ship. IDLE is for when the ship has taken its shot.

- public Ship(int id, String name, Game parentGame)
  Constructor. Assigns the arguments' values to their corresponding class fields. The health and bulletDamage are both set to one by default. The shipState is set to IDLE by default.

- public Ship(int id, String name, int health, int bulletDamage, Game parentGame)
  Constructor. Assigns the arguments' values to their corresponding class fields. The shipState is set to IDLE by default.

- public String getShipName()
  Returns the name of the ship.

- public boolean isShipAlive()
  Returns true if the ship is alive, false if not. A ship is dead whenever its health reaches zero.

- public ShipState getShipState()
  Returns the state of the ship.

- public void takeDamage(int damageAmount)
  Decreases the damageAmount from the ship's health & kills its thread if the ship dies.

- public void shootAt(Ship target)
  Shoots at the target ship & damages it by the bulletDamage value. Prints the event of shooting to terminal. If the Game.gameType is set to FUTURE, then checks if the target is dead after taking the shot. If so, outputs the event of termination to terminal.

- public boolean equals(Ship other)
  Returns true if the ship is the same as the other. Two ships are equal if they have the same name.

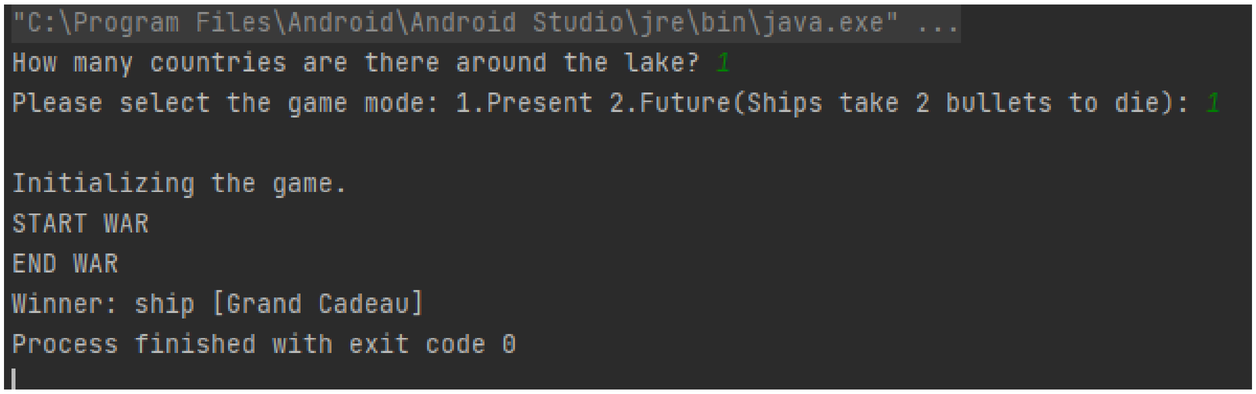
- public void run()
  An implementation of Thread.run() for the *Ship* class. The method contains a while loop which runs as long as the game doesn't end or the ship is alive. As long as in the loop, the ship reloads, waits for its turn, and then if it hasn't died by then, shoots at a target. The target is chosen by Game.getTargetShipForAttacker. Also, the state machine of the ship updates according to each of the steps described above.

NameUtils

Used for providing names for ships. If it runs out of names, it uses an integer index in a string as the name for the ship.

Sample Tests

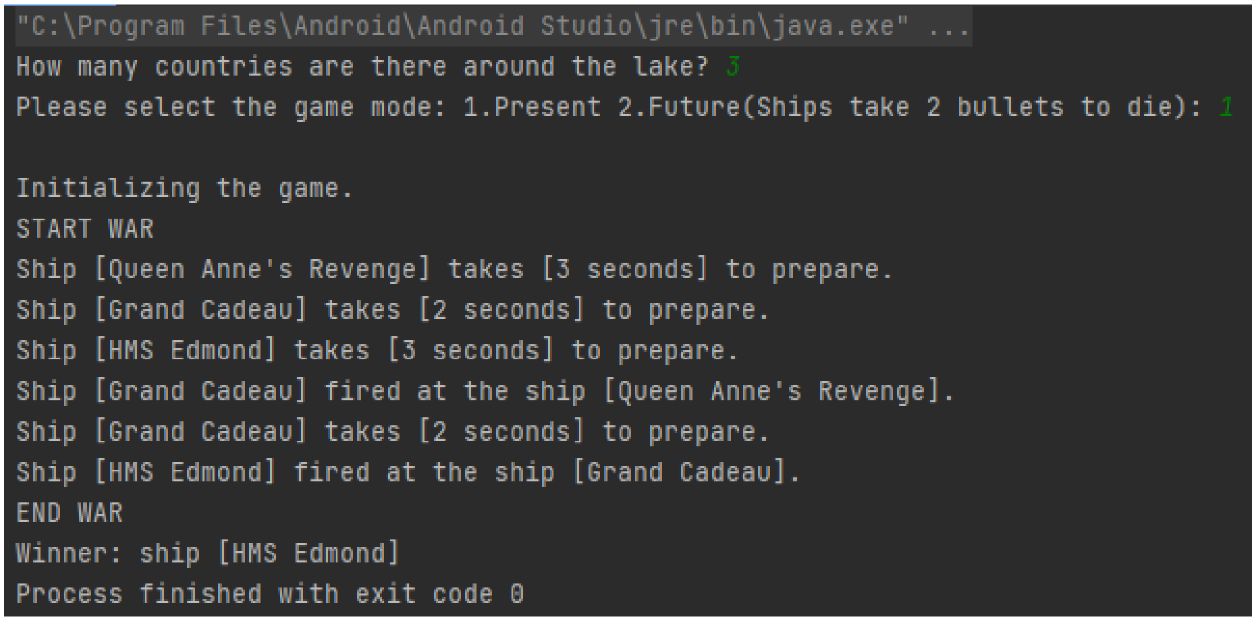
With one ship set in PRESENT timeline:



```
"C:\Program Files\Android\Android Studio\jre\bin\java.exe" ...
How many countries are there around the lake? 1
Please select the game mode: 1.Present 2.Future(Ships take 2 bullets to die): 1

Initializing the game.
START WAR
END WAR
Winner: ship [Grand Cadeau]
Process finished with exit code 0
```

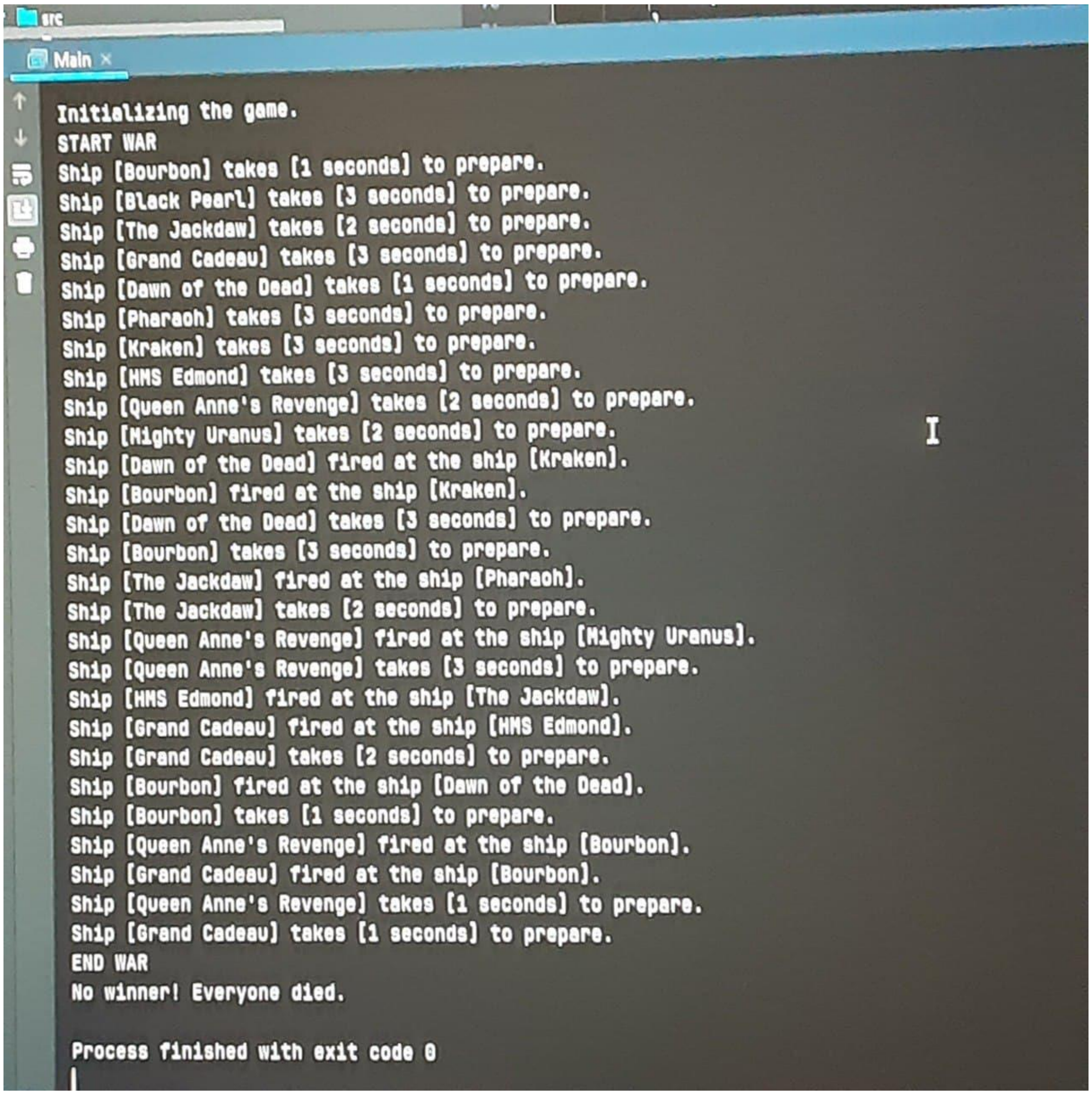
With 3 ships set in PRESENT timeline:



```
"C:\Program Files\Android\Android Studio\jre\bin\java.exe" ...
How many countries are there around the lake? 3
Please select the game mode: 1.Present 2.Future(Ships take 2 bullets to die): 1

Initializing the game.
START WAR
Ship [Queen Anne's Revenge] takes [3 seconds] to prepare.
Ship [Grand Cadeau] takes [2 seconds] to prepare.
Ship [HMS Edmond] takes [3 seconds] to prepare.
Ship [Grand Cadeau] fired at the ship [Queen Anne's Revenge].
Ship [Grand Cadeau] takes [2 seconds] to prepare.
Ship [HMS Edmond] fired at the ship [Grand Cadeau].
END WAR
Winner: ship [HMS Edmond]
Process finished with exit code 0
```

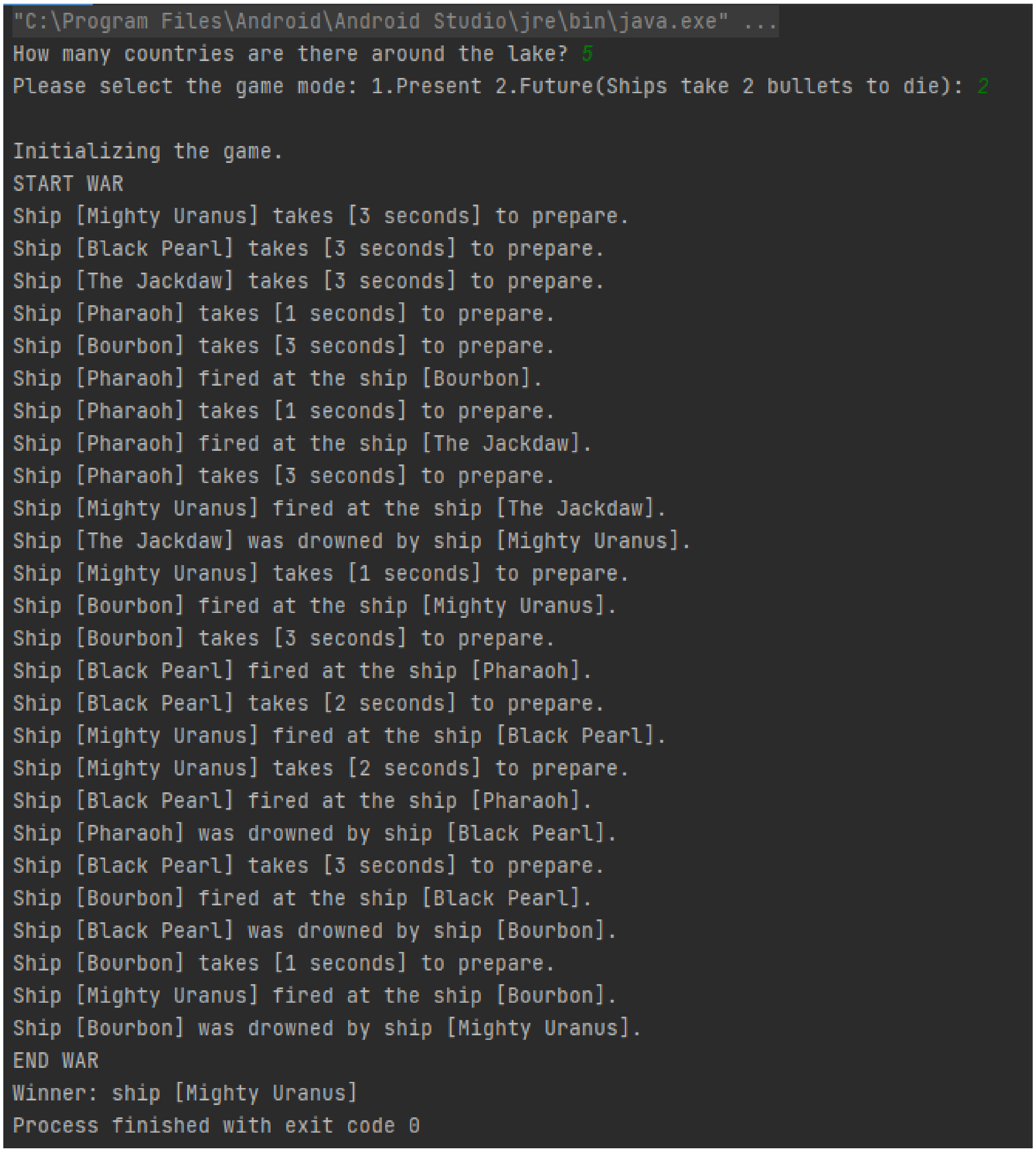
A <u>no winner</u> situation with 10 ships in PRESENT timeline:



```
Initializing the game.
START WAR
Ship [Bourbon] takes [1 seconds] to prepare.
Ship [Black Pearl] takes [3 seconds] to prepare.
Ship [The Jackdaw] takes [2 seconds] to prepare.
Ship [Grand Cadeau] takes [3 seconds] to prepare.
Ship [Dawn of the Dead] takes [1 seconds] to prepare.
Ship [Pharaoh] takes [3 seconds] to prepare.
Ship [Kraken] takes [3 seconds] to prepare.
Ship [HMS Edmond] takes [3 seconds] to prepare.
Ship [Queen Anne's Revenge] takes [2 seconds] to prepare.
Ship [Mighty Uranus] takes [2 seconds] to prepare.
Ship [Dawn of the Dead] fired at the ship [Kraken].
Ship [Bourbon] fired at the ship [Kraken].
Ship [Dawn of the Dead] takes [3 seconds] to prepare.
Ship [Bourbon] takes [3 seconds] to prepare.
Ship [The Jackdaw] fired at the ship [Pharaoh].
Ship [The Jackdaw] takes [2 seconds] to prepare.
Ship [Queen Anne's Revenge] fired at the ship [Mighty Uranus].
Ship [Queen Anne's Revenge] takes [3 seconds] to prepare.
Ship [HMS Edmond] fired at the ship [The Jackdaw].
Ship [Grand Cadeau] fired at the ship [HMS Edmond].
Ship [Grand Cadeau] takes [2 seconds] to prepare.
Ship [Bourbon] fired at the ship [Dawn of the Dead].
Ship [Bourbon] takes [1 seconds] to prepare.
Ship [Queen Anne's Revenge] fired at the ship [Bourbon].
Ship [Grand Cadeau] fired at the ship [Bourbon].
Ship [Queen Anne's Revenge] takes [1 seconds] to prepare.
Ship [Grand Cadeau] takes [1 seconds] to prepare.
END WAR
No winner! Everyone died.

Process finished with exit code 0
```

With 5 ships set in FUTURE timeline:



```
"C:\Program Files\Android\Android Studio\jre\bin\java.exe" ...
How many countries are there around the lake? 5
Please select the game mode: 1.Present 2.Future(Ships take 2 bullets to die): 2

Initializing the game.
START WAR
Ship [Mighty Uranus] takes [3 seconds] to prepare.
Ship [Black Pearl] takes [3 seconds] to prepare.
Ship [The Jackdaw] takes [3 seconds] to prepare.
Ship [Pharaoh] takes [1 seconds] to prepare.
Ship [Bourbon] takes [3 seconds] to prepare.
Ship [Pharaoh] fired at the ship [Bourbon].
Ship [Pharaoh] takes [1 seconds] to prepare.
Ship [Pharaoh] fired at the ship [The Jackdaw].
Ship [Pharaoh] takes [3 seconds] to prepare.
Ship [Mighty Uranus] fired at the ship [The Jackdaw].
Ship [The Jackdaw] was drowned by ship [Mighty Uranus].
Ship [Mighty Uranus] takes [1 seconds] to prepare.
Ship [Bourbon] fired at the ship [Mighty Uranus].
Ship [Bourbon] takes [3 seconds] to prepare.
Ship [Black Pearl] fired at the ship [Pharaoh].
Ship [Black Pearl] takes [2 seconds] to prepare.
Ship [Mighty Uranus] fired at the ship [Black Pearl].
Ship [Mighty Uranus] takes [2 seconds] to prepare.
Ship [Black Pearl] fired at the ship [Pharaoh].
Ship [Pharaoh] was drowned by ship [Black Pearl].
Ship [Black Pearl] takes [3 seconds] to prepare.
Ship [Bourbon] fired at the ship [Black Pearl].
Ship [Black Pearl] was drowned by ship [Bourbon].
Ship [Bourbon] takes [1 seconds] to prepare.
Ship [Mighty Uranus] fired at the ship [Bourbon].
Ship [Bourbon] was drowned by ship [Mighty Uranus].
END WAR
Winner: ship [Mighty Uranus]
Process finished with exit code 0
```