University of New Brunswick

Faculty of Computer Science

Course: CS2043 – Software Engineering I Deliverable #: ____4_____

Instructor: Natalie Webber Date: _____2017/02/28_____

Project group members:

Student #1: _____3413735_____ / _____Andrew Hampton_____

Student Number / Student Name

Student #2: _____3516474_____ / _____Shane Palletier_____

Student Number / Student Name

Table of Contents

Source Code

BGSetupParser class

```
import java.util.ArrayList;

public class BGSetupParser {
  public static Ship parseMessage(String message) {
    ArrayList<String> parts = new ArrayList<String>();
    String[] tokens = message.split(" ");

    if (tokens[0] == "AC") {
      if (tokens.length != 6) {
        return null;
      } else {
        for (int i = 1; i < tokens.length; i++) {
          parts.add(tokens[i]);
        }
      }
    } else if (tokens[0] == "CR") {
      if (tokens.length != 5) {
        return null;
      } else {
        for (int i = 1; i < tokens.length; i++) {
          parts.add(tokens[i]);
        }
      }
    } else if (tokens[0] == "SB") {
      if (tokens.length != 4) {
        return null;
      } else {
        for (int i = 1; i < tokens.length; i++) {
          parts.add(tokens[i]);
        }
      }
    } else if (tokens[0] == "FR") {
      if (tokens.length != 3) {
        return null;
      } else {
        for (int i = 1; i < tokens.length; i++) {
          parts.add(tokens[i]);
        }
      }
    } else {
      return null;
    }

    Ship temp = new Ship(tokens[0], parts);
    if (temp.isValid()) {
      return temp;
    } else {
      return null;
```

```
    }
  }
}
```

GameBoard class

```
import java.util.ArrayList;

public class GameBoard {
  private ArrayList<Ship> playerOneShips;
  private ArrayList<Ship> playerTwoShips;

  public GameBoard() {
      playerOneShips = new ArrayList<Ship>();
      playerTwoShips = new ArrayList<Ship>();
  }

  public Boolean addShip(int playerNumber, Ship ship) {
    if (playerNumber == 1) {
      return playerOneShips.add(ship);
    } else if (playerNumber == 2) {
      return playerTwoShips.add(ship);
    } else {
      return false;
    }
  }

  public Boolean removeShip(int playerNumber, Ship ship) {
    if (playerNumber == 1) {
      return playerOneShips.remove(ship);
    } else if (playerNumber == 2) {
      return playerTwoShips.remove(ship);
    } else {
      return false;
    }
  }

  public String attack(int playerNumber, String location) {
    if (playerNumber == 1) {
      for (int i = 0; i < playerOneShips.size(); i++) {
        if (playerOneShips.get(i).partAt(location)) {
          playerOneShips.get(i).removePart(location);
          if (playerOneShips.get(i).isValid()) {
            return "sunk";
          } else {
            return "hit";
          }
        } else {
          return "miss";
        }
      }
    } else if (playerNumber == 2) {
      for (int i = 0; i < playerTwoShips.size(); i++) {
```

```
        if (playerTwoShips.get(i).partAt(location)) {
          playerTwoShips.get(i).removePart(location);
          if (playerTwoShips.get(i).isValid()) {
            return "sunk";
          } else {
            return "hit";
          }
        } else {
          return "miss";
        }
      }
    } else {
      return "err";
    }
    return "err";
  }
}
```

GameManager class

```
public class GameManager {
  private static Boolean isInSetup;
  private static GameBoard board;
  private static Server server;

  public static void main(String[] args) {
    // Port defaults to 314159 if a port isn't passed in on the
command line

   // TODO: main method
    isInSetup = true;
    board = new GameBoard();
           server = new Server(args.length != 0 ? args[0] : "31415");

  }

  public Boolean receiveMessage(int playerNumber, String message) {
    if (isInSetup == true) {
      String[] unparsedShips = message.split(",");

      boolean success = false;

      for (int i = 0; i < unparsedShips.length; i++) {
        success = false;
        if (!board.addShip(playerNumber,
BGSetupParser.parseMessage(unparsedShips[i]))) {
          break;
        }
        success = true;
      }

      if (success) {
        if (server != null) {
```

```
          server.sendMessage(playerNumber, "ack," + playerNumber);
        }
      } else {
        if (server != null) {
          server.sendMessage(playerNumber, "err");
        }
      }

    } else {
      String result = board.attack(playerNumber, message);
      server.sendMessage(playerNumber, result);
      server.sendMessage(playerNumber == 1 ? 2 : 1, message + ',' +
result);
    }
    System.out.println("It works");
    return true;
  }
}
```

Server class

```
import java.net.*;
import java.io.*;

public class Server {
  private ServerSocket socket;
  private Socket playerOneSocket;
  private Socket playerTwoSocket;
  private BufferedReader playerOneIn;
  private BufferedReader playerTwoIn;
  private PrintWriter playerOneOut;
  private PrintWriter playerTwoOut;

  public Server(String port) {
    // TODO: server connection stuff
    try {
      socket = new ServerSocket(Integer.parseInt(port));
    } catch (IOException e) {
      System.err.println("Couldn't listen on: " + port + ". " +
e.getMessage());
    }

    System.out.println("Server listening on port " + port);

    try {
      playerOneSocket = socket.accept();
    } catch (IOException e) {
      System.err.println("Accepting player one failed: " +
e.getMessage());
    }


    try {
```

```
      playerOneIn = new BufferedReader
                    (new
InputStreamReader(playerOneSocket.getInputStream()));
      playerOneOut = new PrintWriter
                     (playerOneSocket.getOutputStream());

    } catch (IOException e) {
      System.err.println("Cannot read or write player one: " +
e.getMessage());
    }

    System.out.println("Player one connected");

    try {
      playerTwoSocket = socket.accept();

    } catch (IOException e) {
      System.err.println("Accepting player two failed: " +
e.getMessage());
    }

    try {
      playerTwoIn = new BufferedReader
                    (new
InputStreamReader(playerTwoSocket.getInputStream()));
      playerTwoOut = new PrintWriter
                     (playerTwoSocket.getOutputStream());
           String player2Input;


    } catch (IOException e) {
      System.err.println("Cannot read or write player two: " +
e.getMessage());

    }


    System.out.println("Player two connected");
     try{

           boolean done = false;
           GameManager gm = new GameManager();
           while(!done)
           {
               String player1Input = playerOneIn.readLine();
                String player2Input = playerTwoIn.readLine();

                gm.receiveMessage(1,player1Input);
                 gm.receiveMessage(2,player2Input);

                  if (player1Input.trim().equals("BYE")
||player2Input.trim().equals("BYE") ) {
```

```
                    done = true;
                }
            }

        }
      catch (IOException e) {
         System.err.println("Unable to read from or write to the
client: "
                            + e.getMessage());
      }
      try {
         playerOneOut.close();
         playerOneIn.close();
         playerOneSocket.close();
         socket.close();
      }
      catch (IOException e) {
         System.err.println("Unable to close player one's writer,
reader, or socket: "
                            + e.getMessage());
      }
     try {
         playerTwoOut.close();
         playerTwoIn.close();
         playerTwoSocket.close();

      }
      catch (IOException e) {
         System.err.println("Unable to close player two's writer,
reader, or socket: "
                            + e.getMessage());
      }
     }

  public Boolean sendMessage(int playerNumber, String message) {
    if (playerNumber == 1) {
      playerOneOut.println(message);
    } else if (playerNumber == 2) {
      playerTwoOut.println(message);
    } else {
      return false;
    }
    return false;
  }
}
```

Ship class

```
import java.util.ArrayList;

public class Ship {
  private String type;
  private ArrayList<String> parts;
```

```
  public Ship(String type, ArrayList<String> parts) {
    this.type = type;
    this.parts = parts;
  }

  public Boolean removePart(String location) {
    return parts.remove(location);
  }

  public Boolean isValid() {
    if (parts.size() == 0) {
      return false;
    }

    if (type != "AC" && type != "CR" && type != "SB" && type != "FR")
{
      return false;
    }

    if (type == "AC" && parts.size() != 5) {
      return false;
    }

    if (type == "CR" && parts.size() != 4) {
      return false;
    }

    if (type == "SB" && parts.size() != 3) {
      return false;
    }

    if (type == "FR" && parts.size() != 2) {
      return false;
    }

    for (int i = 0; i < parts.size(); i++) {
      char row = parts.get(i).charAt(0);

      if (row != 'A' && row != 'B' && row != 'C' && row != 'D' && row
!= 'E' &&
          row != 'F' && row != 'G' && row != 'H' && row != 'I' && row
!= 'J') {
        return false;
      }

      char col = parts.get(i).charAt(1);

      if (col != '1' && col != '2' && col != '3' && col != '4' && col
!= '5' &&
          col != '6' && col != '7' && col != '8' && col != '9') {
        return false;
```

```
      }
    }

    for (int i = 0; i < parts.size(); i++) {
      if (parts.get(0).charAt(0) != parts.get(i).charAt(0) &&
parts.get(0).charAt(1) != parts.get(i).charAt(1)) {
        return false;
      }
    }

    return true;
  }

  public Boolean partAt(String location) {
    return parts.contains(location);
  }
}
```

Test Report

BGSetupParserTest class

```
public class BGSetupParserTest {
  public static void main(String[] args) {
    Ship ship = BGSetupParser.parseMessage("AC A1 A2 A3 A4 A5");
    System.out.println(ship.isValid());

    ship = BGSetupParser.parseMessage("AC A1");
    System.out.println(ship);

    ship = BGSetupParser.parseMessage("AC A1 A2 A3 A4 A5 A6");
    System.out.println(ship);

    ship = BGSetupParser.parseMessage("AC A11 A2 A3 A4 A5");
    System.out.println(ship);
}
  }
```

GameBoardTest class

```
import java.util.ArrayList;

public class GameBoardTest {
  public static void main(String[] args) {
    GameBoard board = new GameBoard();

    ArrayList<String> parts = new ArrayList<String>();
    parts.add("A1");
    parts.add("A2");
```

```
    Ship ship = new Ship("FR", parts);
    board.addShip(1, ship);

    parts.clear();
    parts.add("A3");
    parts.add("A4");
    ship = new Ship("FR", parts);
    System.out.println(board.addShip(1, ship));

    GameBoard board2 = new GameBoard();

    parts.clear();
    parts.add("A1");
    parts.add("A2");
    Ship ship2 = new Ship("FR", parts);
    board2.addShip(1, ship2);

    parts.clear();
    parts.add("A2");
    parts.add("A3");
    Ship ship3 = new Ship("FR", parts);
    System.out.println(board2.addShip(1, ship3));
  }
}
```

ShipTest class

```
import java.util.ArrayList;

public class ShipTest {
  public static void main(String[] args) {
    ArrayList<String> parts = new ArrayList<String>();
    parts.add("A1");
    Ship ship = new Ship("FR", parts);
    System.out.println(ship.isValid());

    parts.clear();
    parts.add("A1");
    parts.add("A2");
    parts.add("A3");
    ship = new Ship("FR", parts);
    System.out.println(ship.isValid());

    parts.clear();
    parts.add("A1");
    parts.add("B2");
    ship = new Ship("FR", parts);
    System.out.println(ship.isValid());

    parts.clear();
    parts.add("A11");
    parts.add("A12");
    ship = new Ship("FR", parts);
```

```
    System.out.println(ship.isValid());

    parts.clear();
    parts.add("A1");
    parts.add("A2");
    ship = new Ship("FR", parts);
    System.out.println(ship.isValid());

    parts.clear();
    parts.add("A1");
    parts.add("A2");
    ship = new Ship("FR", parts);
    System.out.println(ship.partAt("A3"));

    parts.clear();
    parts.add("A1");
    parts.add("A2");
    ship = new Ship("FR", parts);
    System.out.println(ship.partAt("A2"));

    parts.clear();
    parts.add("A1");
    parts.add("A2");
    ship = new Ship("FR", parts);
    ship.removePart("A2");
    System.out.println(ship.partAt("A2"));
  }
}
```

Time Sheet

Activity	Start	Completion	Number hours Shane Pelletier	Number hours Andrew Hampton
Prepare Project Plan	2017-01-13	2017-01-24	1.5	1.5
Prepare UML diagram and documentation	2017-01-24	2017-02-02	1.5	1.3
Prepare Test plan	2017-02-02	2017-02-09	1.5	1.5
battleship server implementation and unit testing	2017-02-14	2017-02-28	2	2