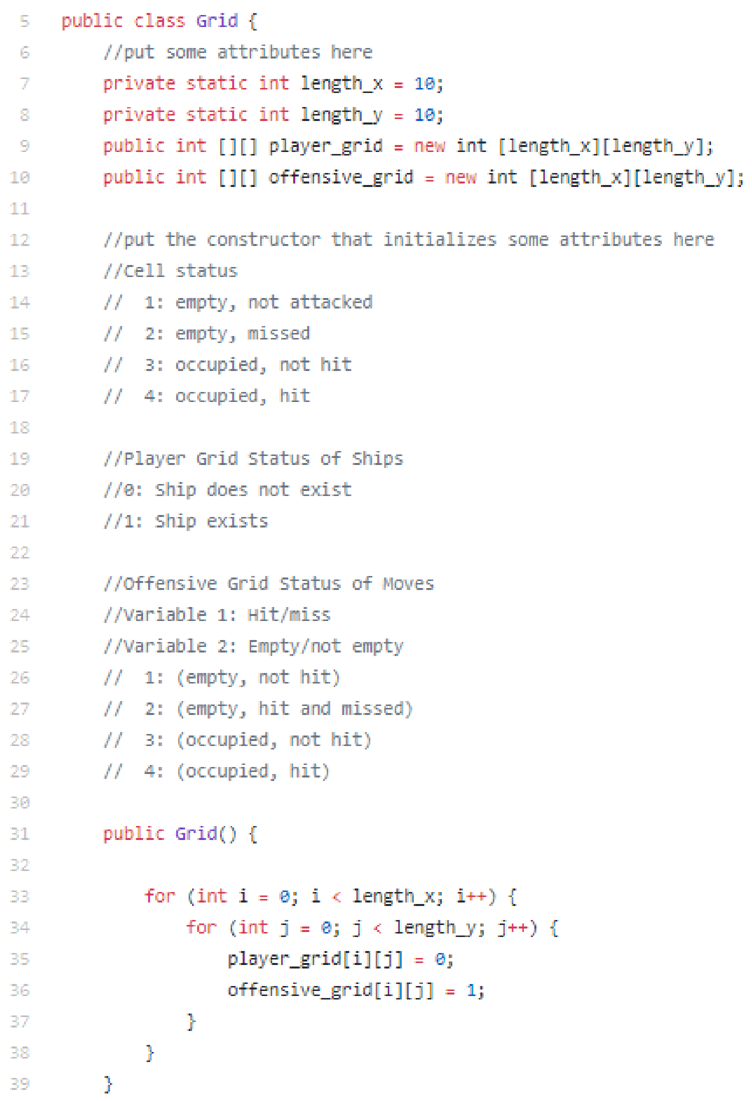
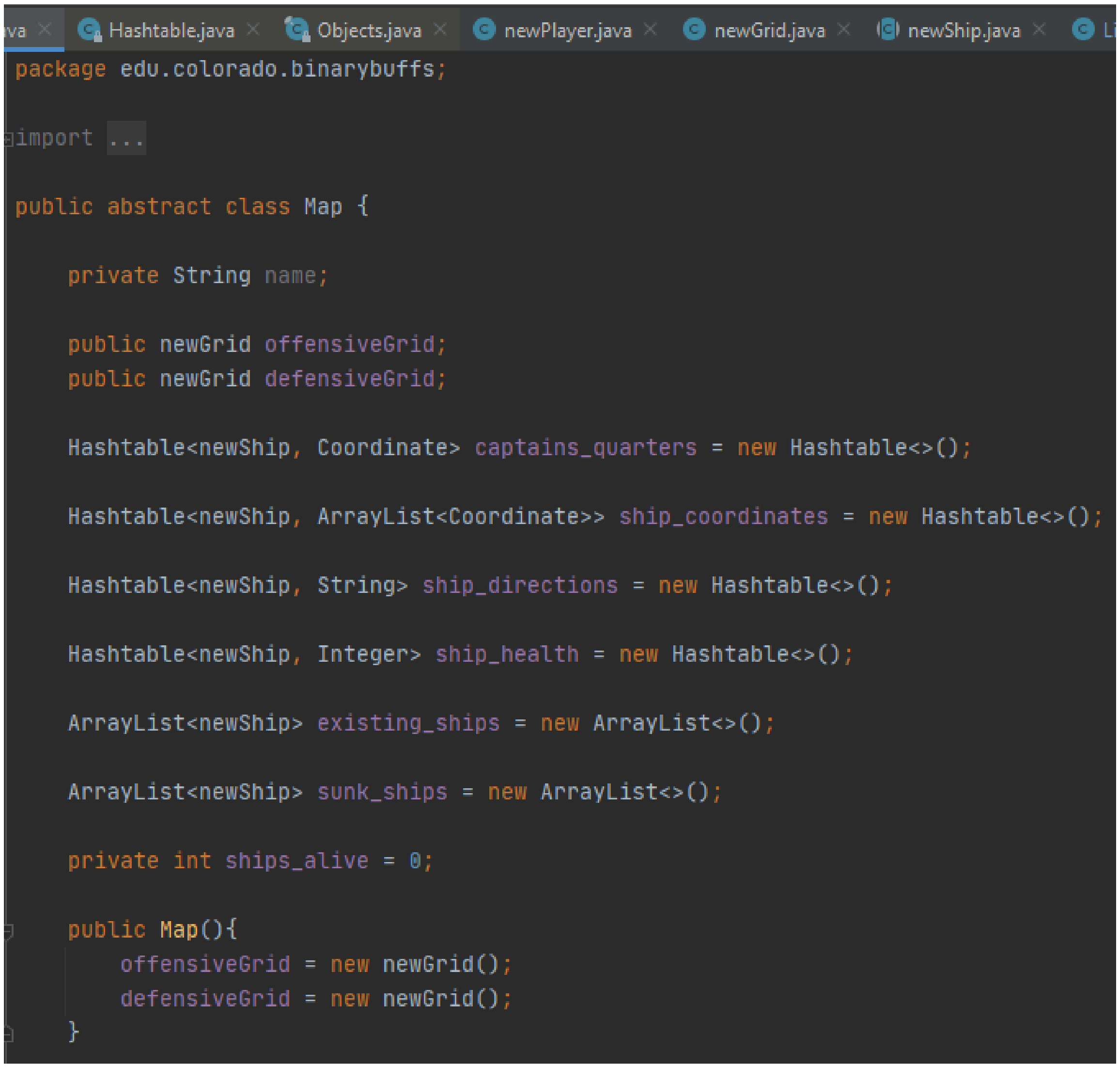
REFACTORING SCREENSHOTS

1a) Previous method for implementing player/opponent grids:



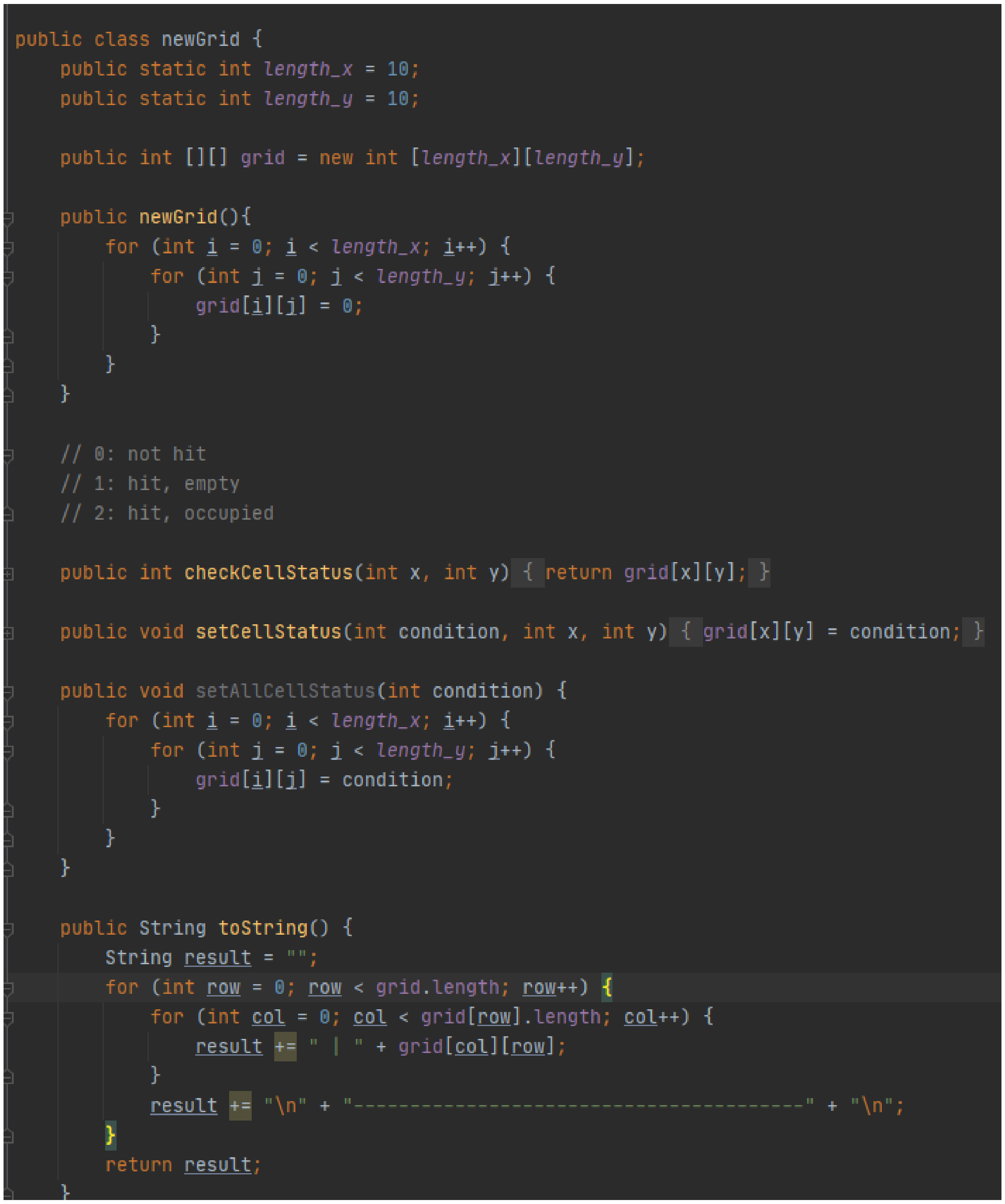
```java
public class Grid {
    //put some attributes here
    private static int length_x = 10;
    private static int length_y = 10;
    public int [][] player_grid = new int [length_x][length_y];
    public int [][] offensive_grid = new int [length_x][length_y];

    //put the constructor that initializes some attributes here
    //Cell status
    //  1: empty, not attacked
    //  2: empty, missed
    //  3: occupied, not hit
    //  4: occupied, hit

    //Player Grid Status of Ships
    //0: Ship does not exist
    //1: Ship exists

    //Offensive Grid Status of Moves
    //Variable 1: Hit/miss
    //Variable 2: Empty/not empty
    //  1: (empty, not hit)
    //  2: (empty, hit and missed)
    //  3: (occupied, not hit)
    //  4: (occupied, hit)

    public Grid() {

        for (int i = 0; i < length_x; i++) {
            for (int j = 0; j < length_y; j++) {
                player_grid[i][j] = 0;
                offensive_grid[i][j] = 1;
            }
        }
    }
```

1b) New method for implementing player/opponent grids and various maps:



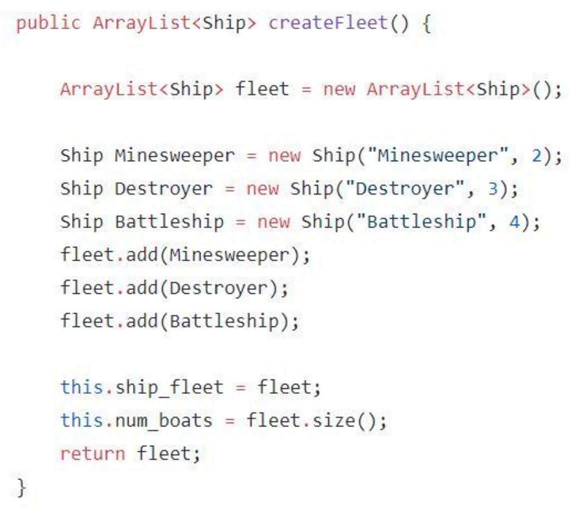
```
package edu.colorado.binarybuffs;

import ...

public abstract class Map {

    private String name;

    public newGrid offensiveGrid;
    public newGrid defensiveGrid;

    Hashtable<newShip, Coordinate> captains_quarters = new Hashtable<>();

    Hashtable<newShip, ArrayList<Coordinate>> ship_coordinates = new Hashtable<>();

    Hashtable<newShip, String> ship_directions = new Hashtable<>();

    Hashtable<newShip, Integer> ship_health = new Hashtable<>();

    ArrayList<newShip> existing_ships = new ArrayList<>();

    ArrayList<newShip> sunk_ships = new ArrayList<>();

    private int ships_alive = 0;

    public Map(){
        offensiveGrid = new newGrid();
        defensiveGrid = new newGrid();
    }
```

1c) Grid class creates a defensive and offensive grid for player

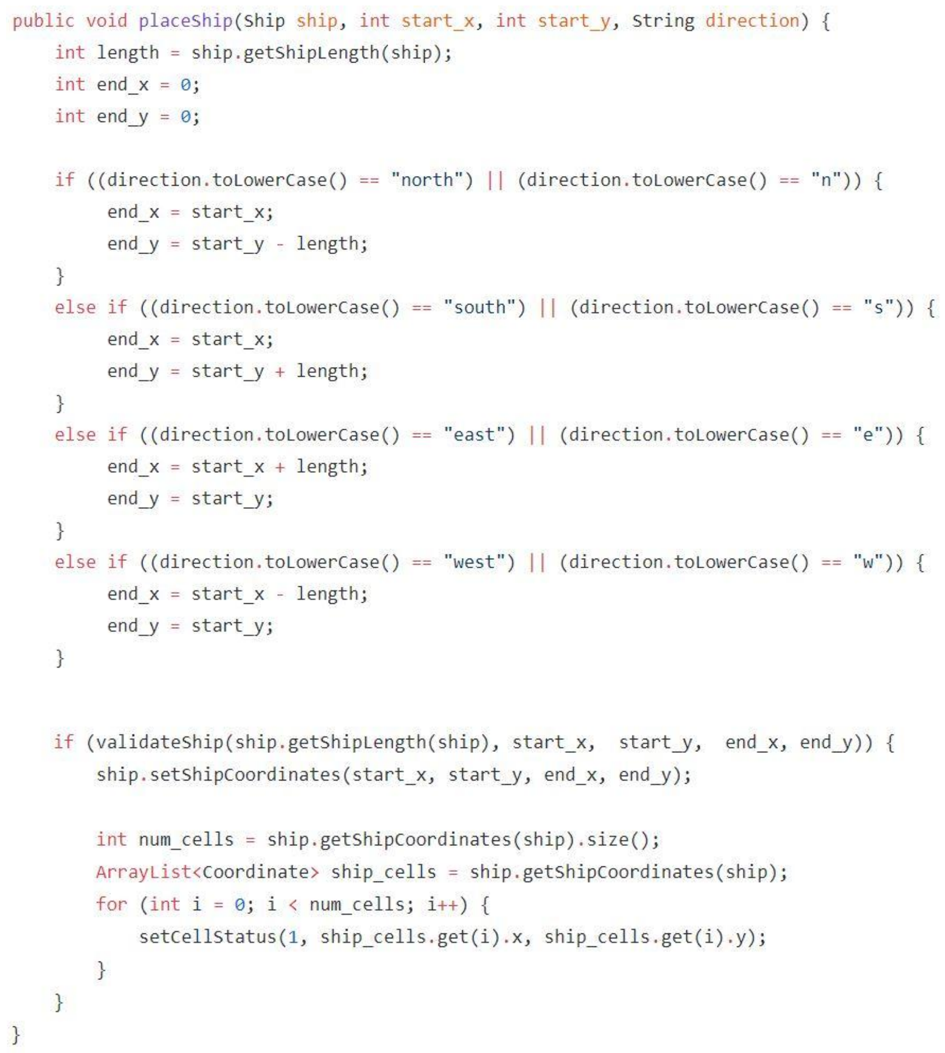


```java
public class newGrid {
    public static int length_x = 10;
    public static int length_y = 10;

    public int [][] grid = new int [length_x][length_y];

    public newGrid(){
        for (int i = 0; i < length_x; i++) {
            for (int j = 0; j < length_y; j++) {
                grid[i][j] = 0;
            }
        }
    }

    // 0: not hit
    // 1: hit, empty
    // 2: hit, occupied

    public int checkCellStatus(int x, int y) { return grid[x][y]; }

    public void setCellStatus(int condition, int x, int y) { grid[x][y] = condition; }

    public void setAllCellStatus(int condition) {
        for (int i = 0; i < length_x; i++) {
            for (int j = 0; j < length_y; j++) {
                grid[i][j] = condition;
            }
        }
    }

    public String toString() {
        String result = "";
        for (int row = 0; row < grid.length; row++) {
            for (int col = 0; col < grid[row].length; col++) {
                result += " | " + grid[col][row];
            }
            result += "\n" + "-----------------------------------------" + "\n";
        }
        return result;
    }
```

2a) Previous methodology for placing ships: including creating a fleet and then placing it on a grid.

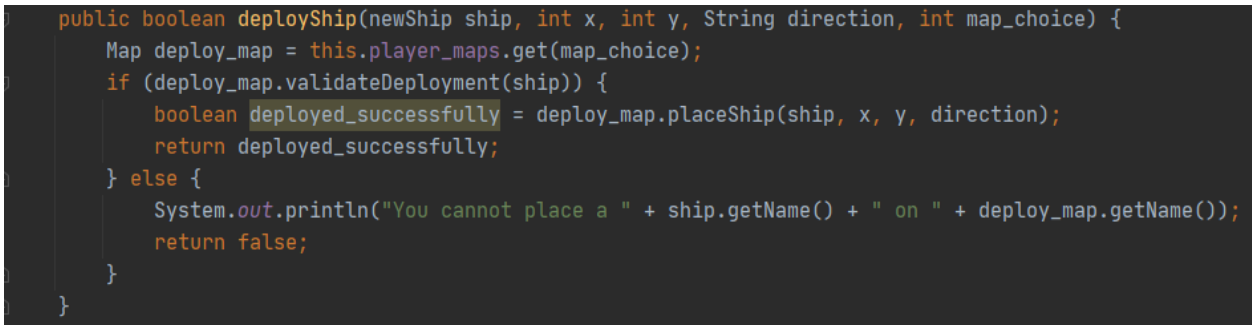


```java
public ArrayList<Ship> createFleet() {

    ArrayList<Ship> fleet = new ArrayList<Ship>();

    Ship Minesweeper = new Ship("Minesweeper", 2);
    Ship Destroyer = new Ship("Destroyer", 3);
    Ship Battleship = new Ship("Battleship", 4);
    fleet.add(Minesweeper);
    fleet.add(Destroyer);
    fleet.add(Battleship);

    this.ship_fleet = fleet;
    this.num_boats = fleet.size();
    return fleet;
}
```

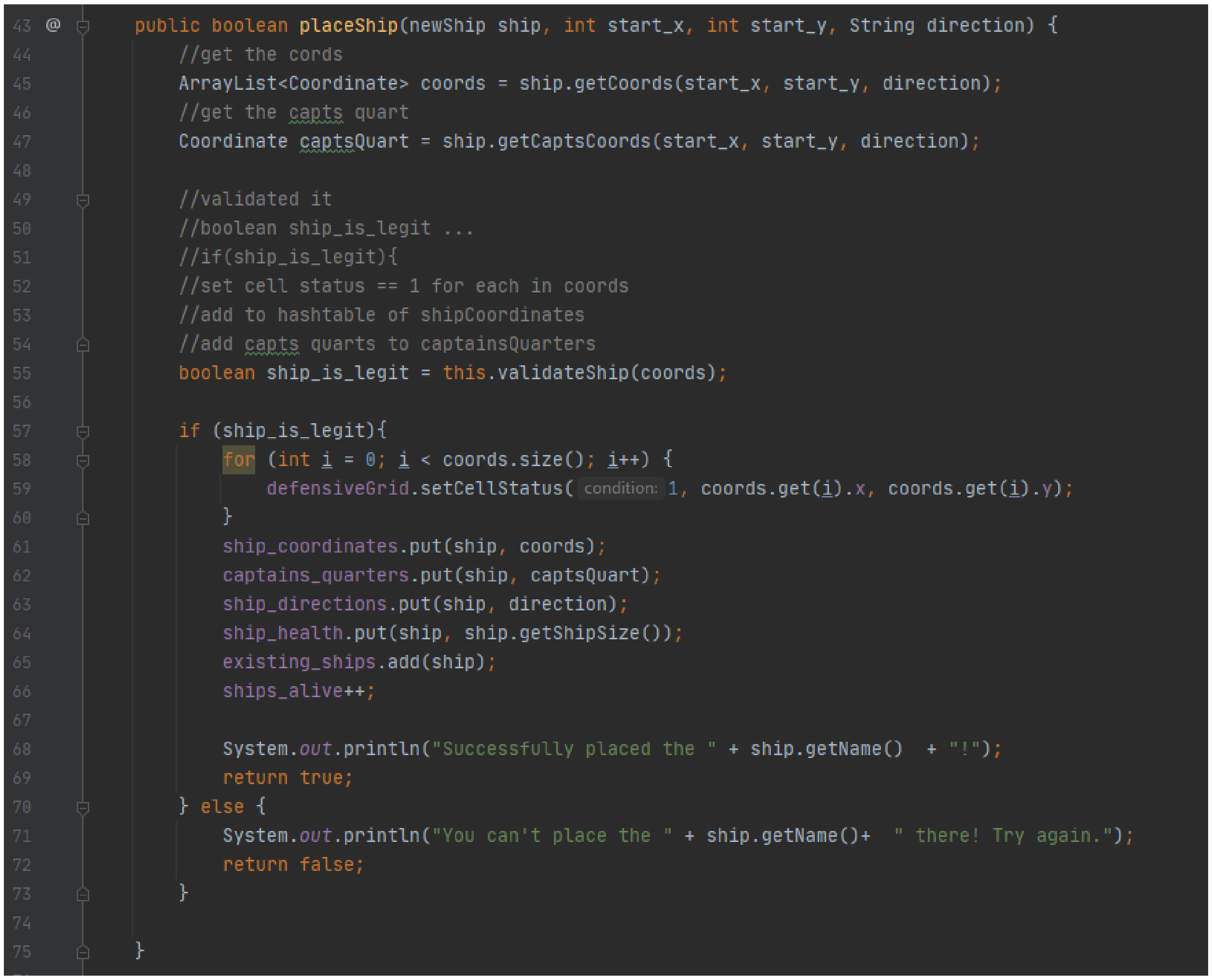


```java
public void placeShip(Ship ship, int start_x, int start_y, String direction) {
    int length = ship.getShipLength(ship);
    int end_x = 0;
    int end_y = 0;

    if ((direction.toLowerCase() == "north") || (direction.toLowerCase() == "n")) {
         end_x = start_x;
         end_y = start_y - length;
    }
    else if ((direction.toLowerCase() == "south") || (direction.toLowerCase() == "s")) {
         end_x = start_x;
         end_y = start_y + length;
    }
    else if ((direction.toLowerCase() == "east") || (direction.toLowerCase() == "e")) {
         end_x = start_x + length;
         end_y = start_y;
    }
    else if ((direction.toLowerCase() == "west") || (direction.toLowerCase() == "w")) {
         end_x = start_x - length;
         end_y = start_y;
    }


    if (validateShip(ship.getShipLength(ship), start_x,  start_y,  end_x, end_y)) {
        ship.setShipCoordinates(start_x, start_y, end_x, end_y);

        int num_cells = ship.getShipCoordinates(ship).size();
        ArrayList<Coordinate> ship_cells = ship.getShipCoordinates(ship);
        for (int i = 0; i < num_cells; i++) {
            setCellStatus(1, ship_cells.get(i).x, ship_cells.get(i).y);
        }
    }
}
```

2b) New methodology for deploying ships: takes in a ship object and places it on map.

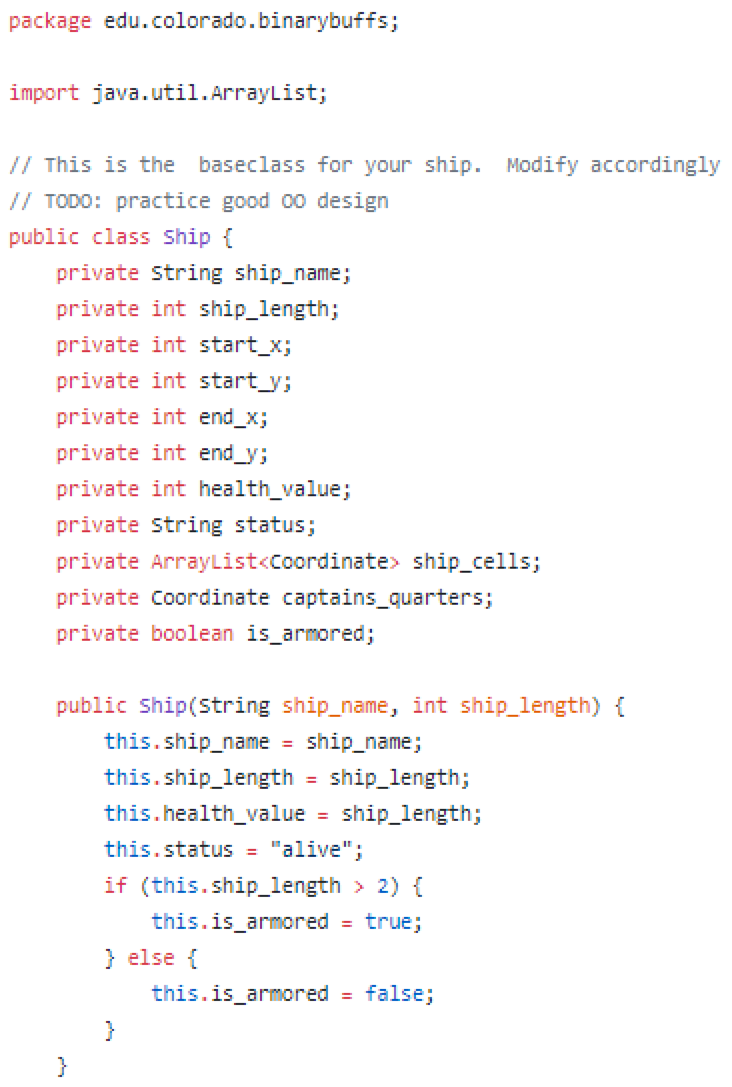


```java
public boolean deployShip(newShip ship, int x, int y, String direction, int map_choice) {
    Map deploy_map = this.player_maps.get(map_choice);
    if (deploy_map.validateDeployment(ship)) {
        boolean deployed_successfully = deploy_map.placeShip(ship, x, y, direction);
        return deployed_successfully;
    } else {
        System.out.println("You cannot place a " + ship.getName() + " on " + deploy_map.getName());
        return false;
    }
}
```



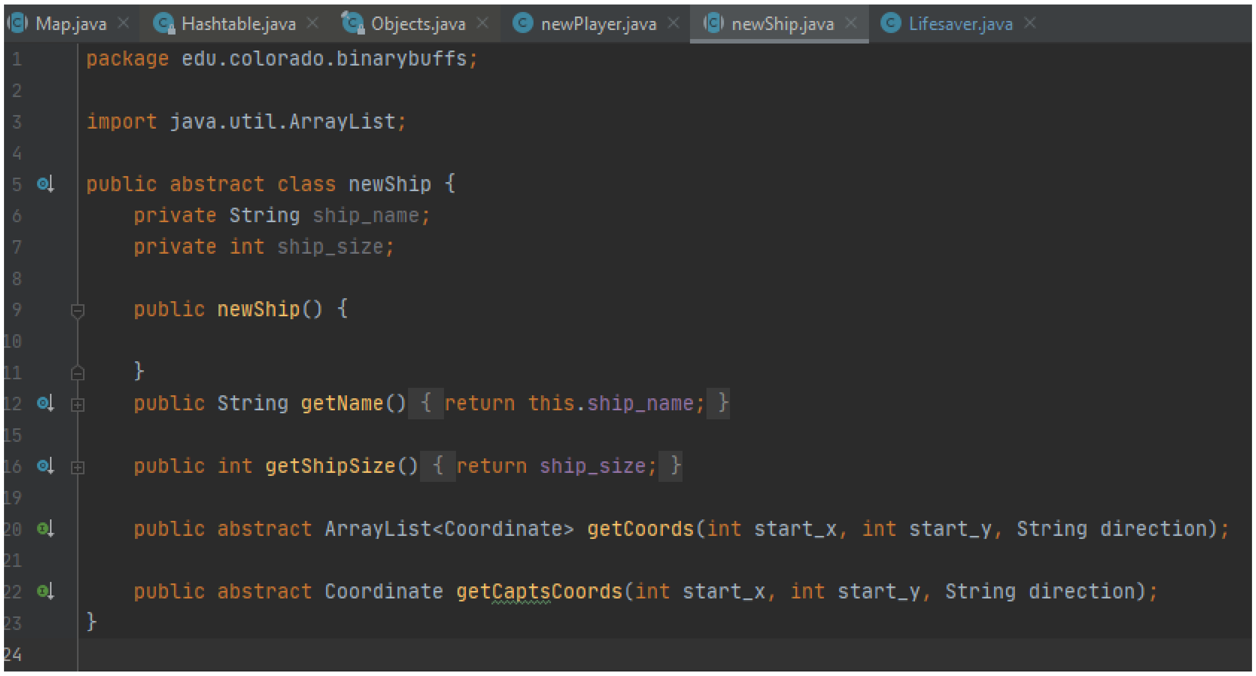
```java
    public boolean placeShip(newShip ship, int start_x, int start_y, String direction) {
        //get the cords
        ArrayList<Coordinate> coords = ship.getCoords(start_x, start_y, direction);
        //get the capts quart
        Coordinate captsQuart = ship.getCaptsCoords(start_x, start_y, direction);

        //validated it
        //boolean ship_is_legit ...
        //if(ship_is_legit){
        //set cell status == 1 for each in coords
        //add to hashtable of shipCoordinates
        //add capts quarts to captainsQuarters
        boolean ship_is_legit = this.validateShip(coords);

        if (ship_is_legit){
            for (int i = 0; i < coords.size(); i++) {
                defensiveGrid.setCellStatus( condition: 1, coords.get(i).x, coords.get(i).y);
            }
            ship_coordinates.put(ship, coords);
            captains_quarters.put(ship, captsQuart);
            ship_directions.put(ship, direction);
            ship_health.put(ship, ship.getShipSize());
            existing_ships.add(ship);
            ships_alive++;

            System.out.println("Successfully placed the " + ship.getName()  + "!");
            return true;
        } else {
            System.out.println("You can't place the " + ship.getName()+  " there! Try again.");
            return false;
        }

    }
```

3a) Previous method for implementing various ships for player fleet:



```java
package edu.colorado.binarybuffs;

import java.util.ArrayList;

// This is the  baseclass for your ship.  Modify accordingly
// TODO: practice good OO design
public class Ship {
    private String ship_name;
    private int ship_length;
    private int start_x;
    private int start_y;
    private int end_x;
    private int end_y;
    private int health_value;
    private String status;
    private ArrayList<Coordinate> ship_cells;
    private Coordinate captains_quarters;
    private boolean is_armored;

    public Ship(String ship_name, int ship_length) {
        this.ship_name = ship_name;
        this.ship_length = ship_length;
        this.health_value = ship_length;
        this.status = "alive";
        if (this.ship_length > 2) {
            this.is_armored = true;
        } else {
            this.is_armored = false;
        }
    }
```

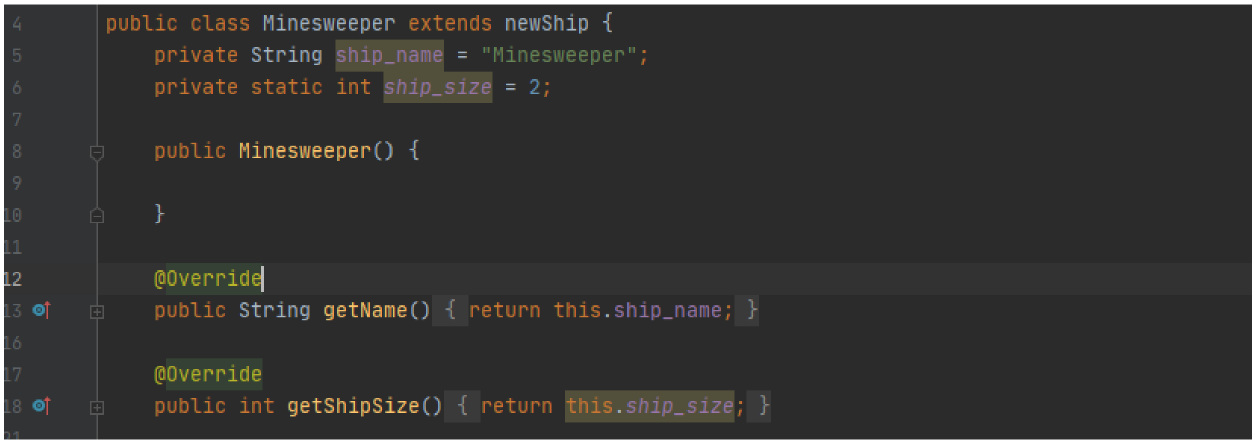
3b) New method for implementing ships - individual ship classes are extended from abstract Ship class:



Map.java × Hashtable.java × Objects.java × newPlayer.java × newShip.java × Lifesaver.java ×

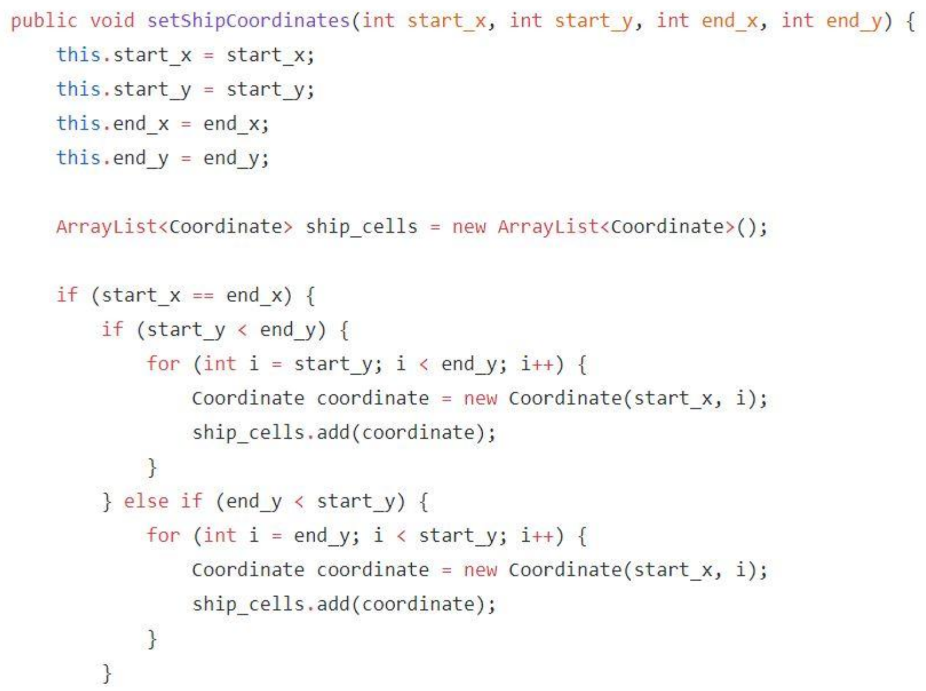
```java
package edu.colorado.binarybuffs;

import java.util.ArrayList;

public abstract class newShip {
    private String ship_name;
    private int ship_size;

    public newShip() {

    }
    public String getName() { return this.ship_name; }

    public int getShipSize() { return ship_size; }

    public abstract ArrayList<Coordinate> getCoords(int start_x, int start_y, String direction);

    public abstract Coordinate getCaptsCoords(int start_x, int start_y, String direction);
}
```

3c) Minesweeper is an individual ship class extended from Ship class

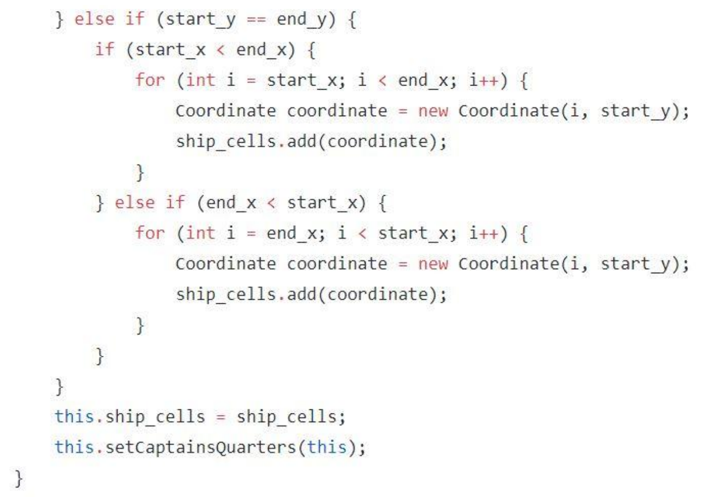


```java
public class Minesweeper extends newShip {
    private String ship_name = "Minesweeper";
    private static int ship_size = 2;

    public Minesweeper() {

    }

    @Override
    public String getName() { return this.ship_name; }

    @Override
    public int getShipSize() { return this.ship_size; }
```

4a) Previous method for setting and getting the ship coordinates:

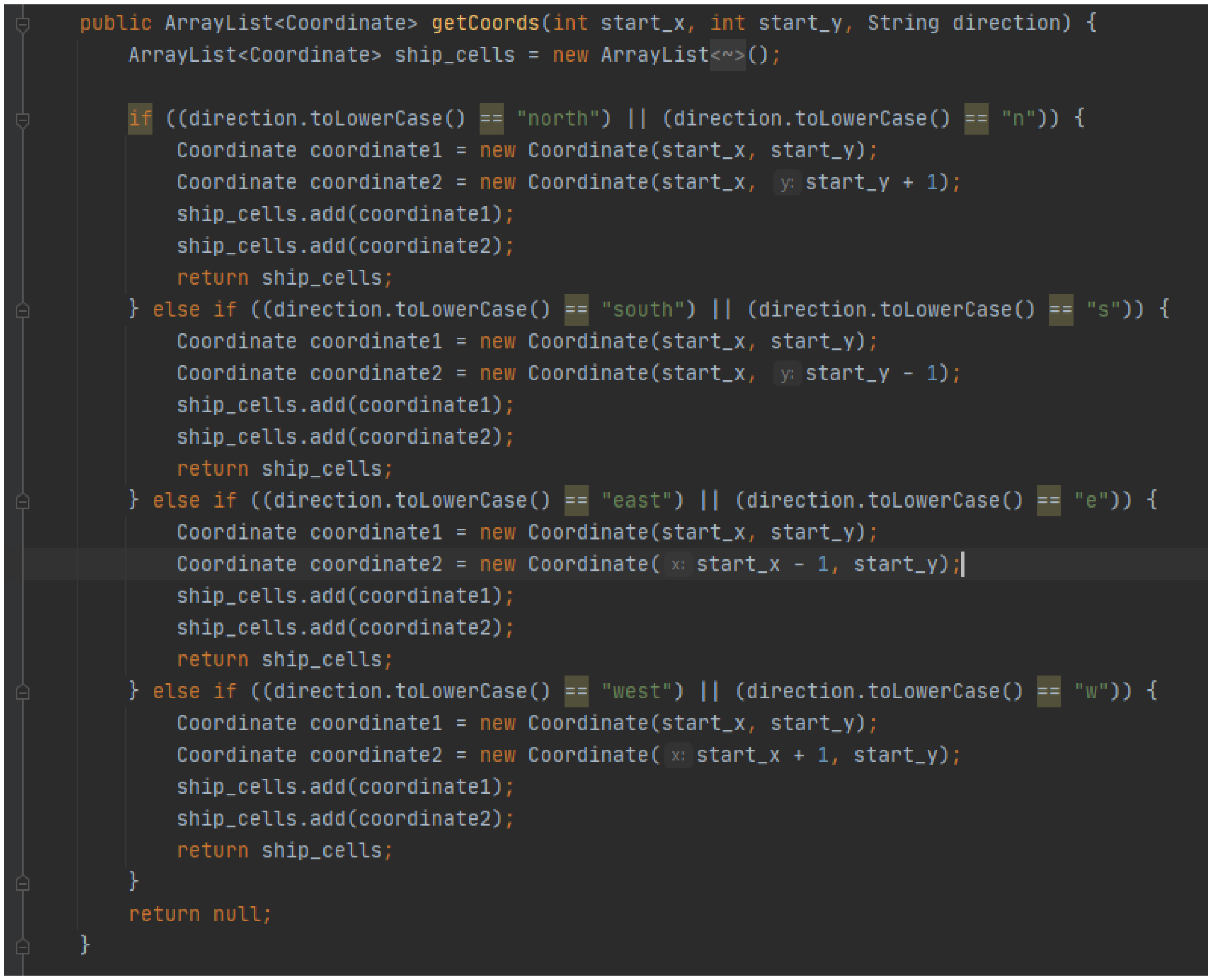


```java
public void setShipCoordinates(int start_x, int start_y, int end_x, int end_y) {
    this.start_x = start_x;
    this.start_y = start_y;
    this.end_x = end_x;
    this.end_y = end_y;

    ArrayList<Coordinate> ship_cells = new ArrayList<Coordinate>();

    if (start_x == end_x) {
        if (start_y < end_y) {
            for (int i = start_y; i < end_y; i++) {
                Coordinate coordinate = new Coordinate(start_x, i);
                ship_cells.add(coordinate);
            }
        } else if (end_y < start_y) {
            for (int i = end_y; i < start_y; i++) {
                Coordinate coordinate = new Coordinate(start_x, i);
                ship_cells.add(coordinate);
            }
        }
```



```java
    } else if (start_y == end_y) {
        if (start_x < end_x) {
            for (int i = start_x; i < end_x; i++) {
                Coordinate coordinate = new Coordinate(i, start_y);
                ship_cells.add(coordinate);
            }
        } else if (end_x < start_x) {
            for (int i = end_x; i < start_x; i++) {
                Coordinate coordinate = new Coordinate(i, start_y);
                ship_cells.add(coordinate);
            }
        }
    }
    this.ship_cells = ship_cells;
    this.setCaptainsQuarters(this);
}
```

4b) New method for setting and getting ship coordinates: example from Minesweeper (simply provides the offset for each ship)



```java
public ArrayList<Coordinate> getCoords(int start_x, int start_y, String direction) {
    ArrayList<Coordinate> ship_cells = new ArrayList<~>();

    if ((direction.toLowerCase() == "north") || (direction.toLowerCase() == "n")) {
        Coordinate coordinate1 = new Coordinate(start_x, start_y);
        Coordinate coordinate2 = new Coordinate(start_x,  y: start_y + 1);
        ship_cells.add(coordinate1);
        ship_cells.add(coordinate2);
        return ship_cells;
    } else if ((direction.toLowerCase() == "south") || (direction.toLowerCase() == "s")) {
        Coordinate coordinate1 = new Coordinate(start_x, start_y);
        Coordinate coordinate2 = new Coordinate(start_x,  y: start_y - 1);
        ship_cells.add(coordinate1);
        ship_cells.add(coordinate2);
        return ship_cells;
    } else if ((direction.toLowerCase() == "east") || (direction.toLowerCase() == "e")) {
        Coordinate coordinate1 = new Coordinate(start_x, start_y);
        Coordinate coordinate2 = new Coordinate( x: start_x - 1, start_y);
        ship_cells.add(coordinate1);
        ship_cells.add(coordinate2);
        return ship_cells;
    } else if ((direction.toLowerCase() == "west") || (direction.toLowerCase() == "w")) {
        Coordinate coordinate1 = new Coordinate(start_x, start_y);
        Coordinate coordinate2 = new Coordinate( x: start_x + 1, start_y);
        ship_cells.add(coordinate1);
        ship_cells.add(coordinate2);
        return ship_cells;
    }
    return null;
}
```