New Features

Invisibility on round a player has the ability to make one ship invisible, meaning you won't be able to attack that ship.

Disable type weapon that prevents ships from moving or going invisible

Wiki Update (4/1/21 3:15 meeting)

Meeting minutes are located after the Revised Project Risks section.

User Stories

- As a developer, I want to communicate effectively while pair programming with my partner to ensure that we are working together to code well.
- As a developer I want to make sure that I am properly using TDD to satisfy the requirements of this Milestone 5.
- As a developer, I want to use the refactoring strategies we learned about in class to refactor the Map class and some of the methods in the class.
- As a developer, I want to refactor the Submarine class to implement some of the more submarine specific features instead of making that a Ship wide change
- As a developer, I want to make sure that my classes are highly cohesive and loosely coupled
- As a player, I want to be able to make one of my ships invisible to the other player.
- As a player, I want to be able to block another player from making his ship invisible or moving.

Planning Game and Time Estimates:

- We will refactor the getAttacked (Map class) system and provide before and after code snapshots.
  - Time Estimated: 2 hours
  - Time Taken: 2 hours
- We will refactor the Submarine class and provide before and after code snapshots.
  - Time Estimated: 3 hours
  - Time Taken: 4 hours
- Invisibility
  - Time Estimated: 2 hours
  - Time Taken: 2 hours
- Disable weapon

        - Time Estimated: 2 hours
        - Time Taken: 2 hours

Revised Project Risks:

- Implementing Refactoring Methods
    - Risk Mitigation: Read through resources given in class, like Code Complete Second Edition and the slides to understand refactoring and when to implement it
- Self set requirements
    - Risk Mitigation: We met more frequently to set goals for ourselves because now we are not being told what features to add

Meeting Minutes

- We read over Milestone 5 requirements as a team and looked over the grading rubric.
- We started working on our wiki and discussed as a team what we wanted in terms of new features, which user stories to tackle before the upcoming meeting, and then split up into two pairs to start pair programming through TDD to work on the user stories.
- We finished up the meeting by collaborating to figure out how long each segment of our coding should take.
- We also reevaluated our project risks, added new risks, and wrote out some risk mitigation strategies as a team.

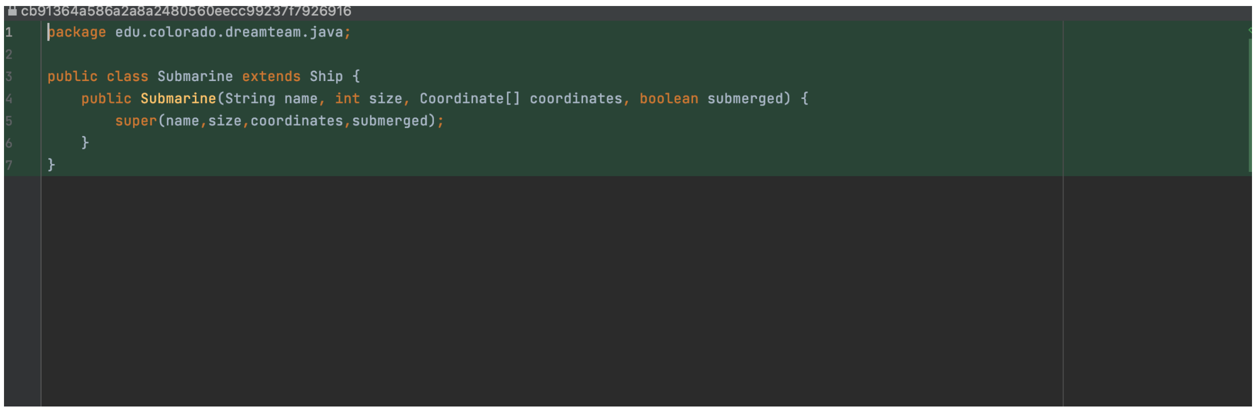
<u>Refactoring & Implementation (Code Complete, Second Edition):</u>

- Increasing modularity
    - Cleaner routine by separating getAttacked() and getAttackedBelow into different classes
- Shortening routine
    - getAttacked() was separated into different functions for each type of weapon to reduce the amount of work one function was doing
    - This also increased modularity, as we have more well-defined, well-named routines that only do one thing
- Increasing cohesion

- ○ getAttackedBelow was moved from the Ship class to the Submarine class, this made a more cohesive and related set of responsibilities for those classes

Adding functionality to Submarine class:

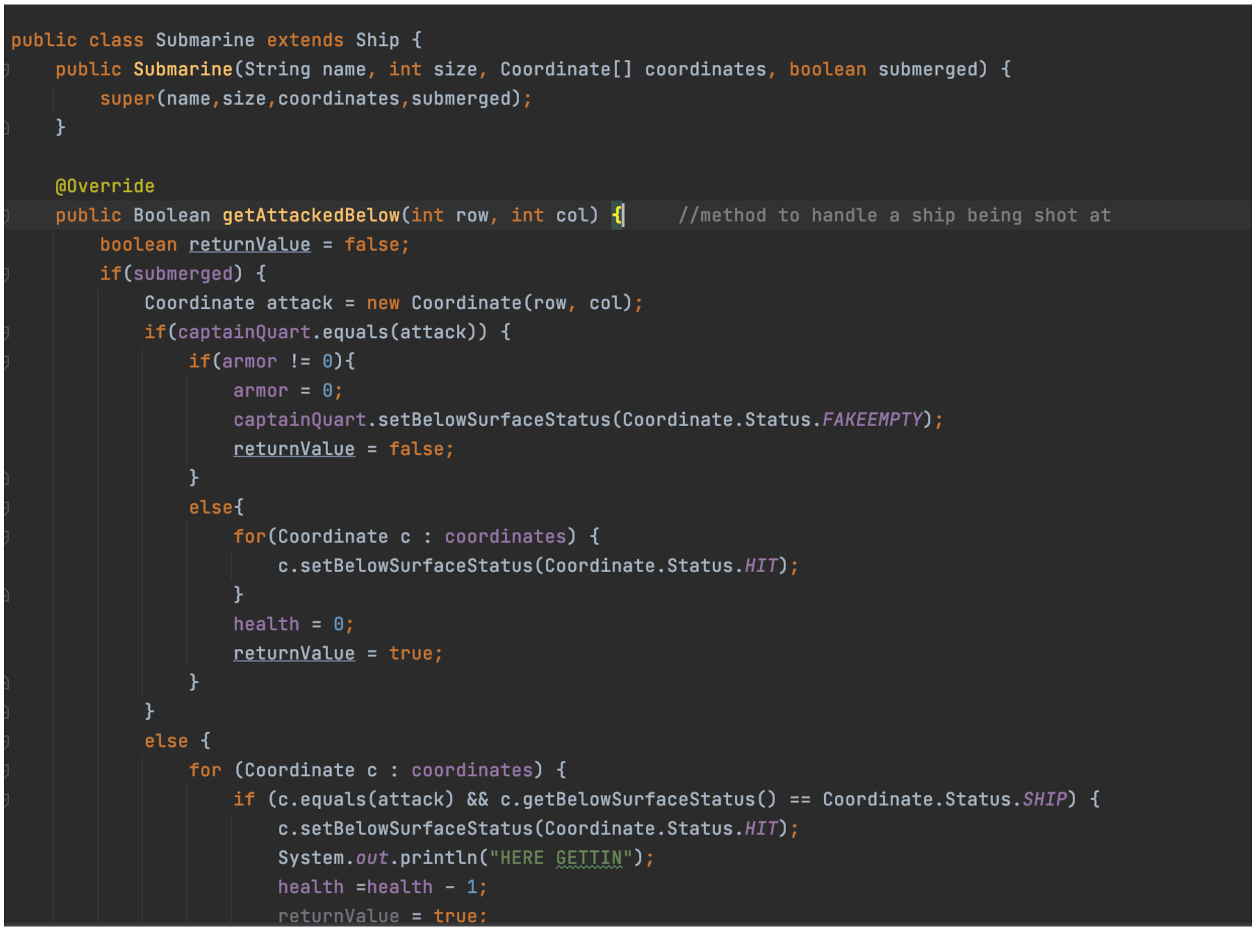
BEFORE:



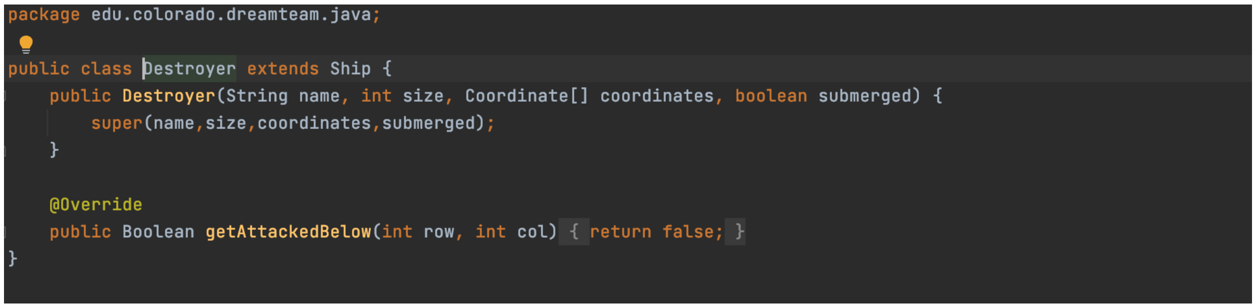
cb91364a586a2a8a2480560eecc99237f7926916

```
package edu.colorado.dreamteam.java;

public class Submarine extends Ship {
    public Submarine(String name, int size, Coordinate[] coordinates, boolean submerged) {
        super(name,size,coordinates,submerged);
    }
}
```

AFTER:



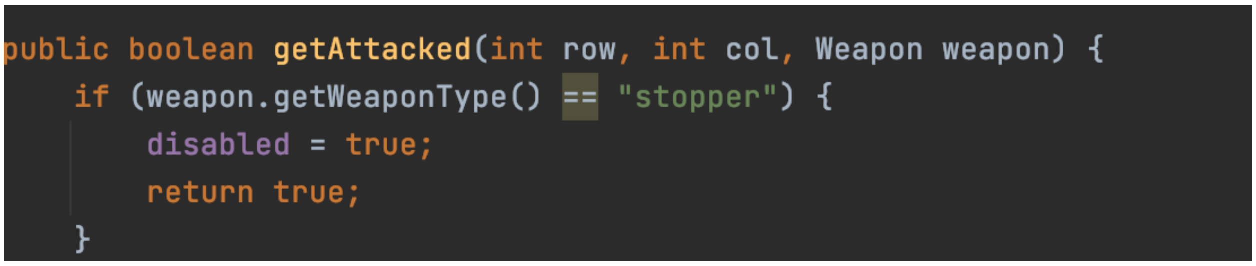
```java
public class Submarine extends Ship {
    public Submarine(String name, int size, Coordinate[] coordinates, boolean submerged) {
        super(name,size,coordinates,submerged);
    }

    @Override
    public Boolean getAttackedBelow(int row, int col) {     //method to handle a ship being shot at
        boolean returnValue = false;
        if(submerged) {
            Coordinate attack = new Coordinate(row, col);
            if(captainQuart.equals(attack)) {
                if(armor != 0){
                    armor = 0;
                    captainQuart.setBelowSurfaceStatus(Coordinate.Status.FAKEEMPTY);
                    returnValue = false;
                }
                else{
                    for(Coordinate c : coordinates) {
                        c.setBelowSurfaceStatus(Coordinate.Status.HIT);
                    }
                    health = 0;
                    returnValue = true;
                }
            }
            else {
                for (Coordinate c : coordinates) {
                    if (c.equals(attack) && c.getBelowSurfaceStatus() == Coordinate.Status.SHIP) {
                        c.setBelowSurfaceStatus(Coordinate.Status.HIT);
                        System.out.println("HERE GETTIN");
                        health =health - 1;
                        returnValue = true;
```



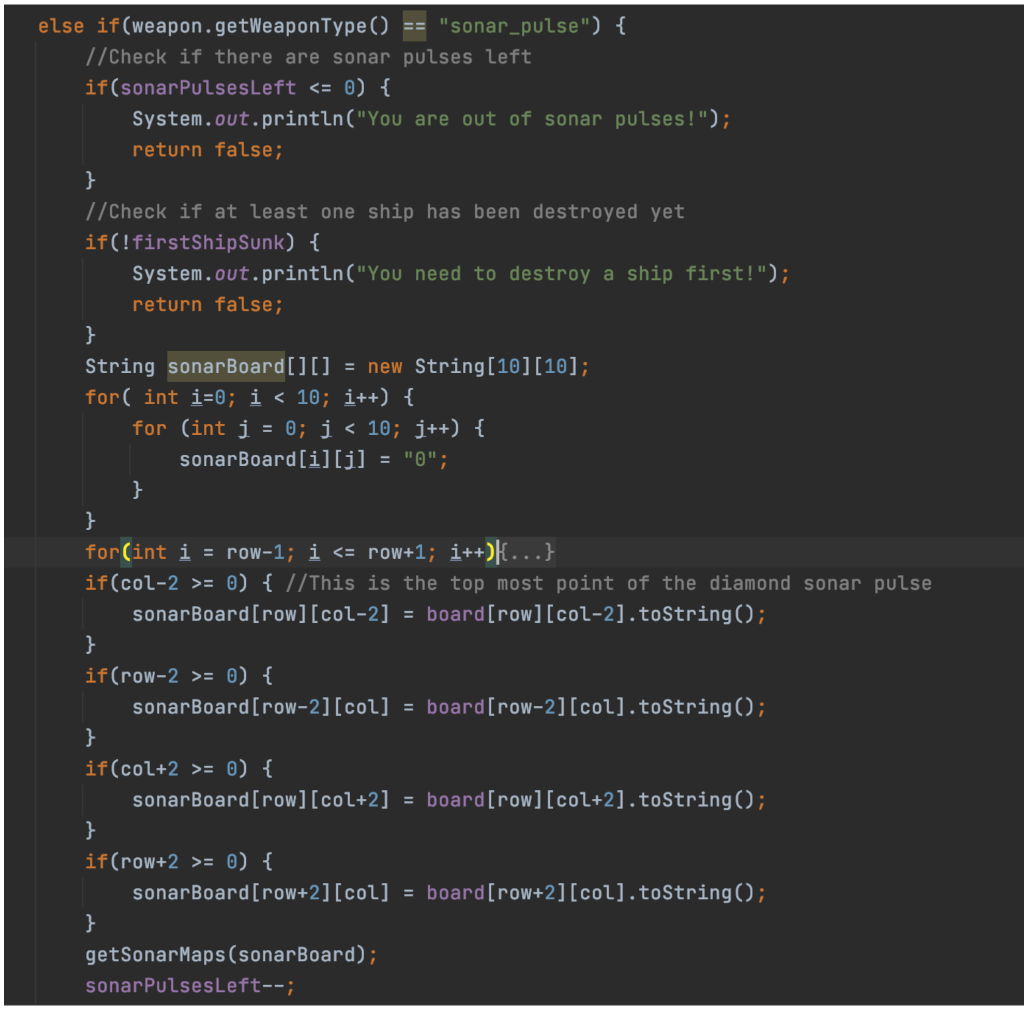
```java
package edu.colorado.dreamteam.java;

public class Destroyer extends Ship {
    public Destroyer(String name, int size, Coordinate[] coordinates, boolean submerged) {
        super(name,size,coordinates,submerged);
    }

    @Override
    public Boolean getAttackedBelow(int row, int col) { return false; }
}
```

Separating getAttacked() into shorter routines:

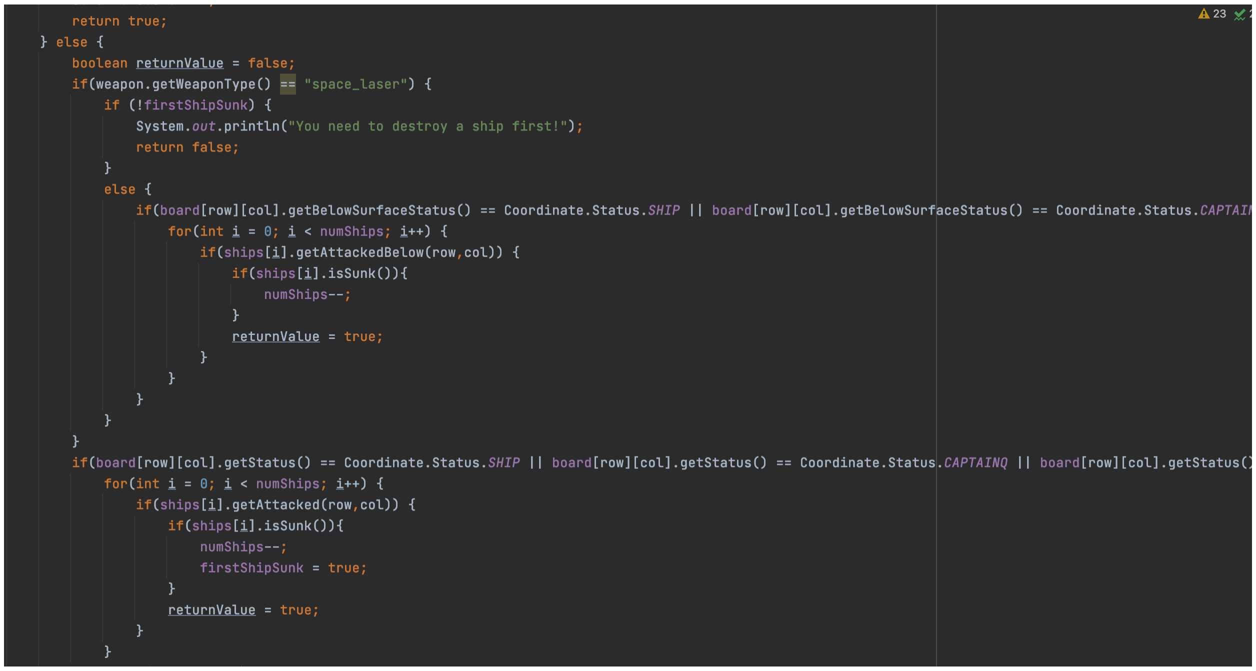
BEFORE:



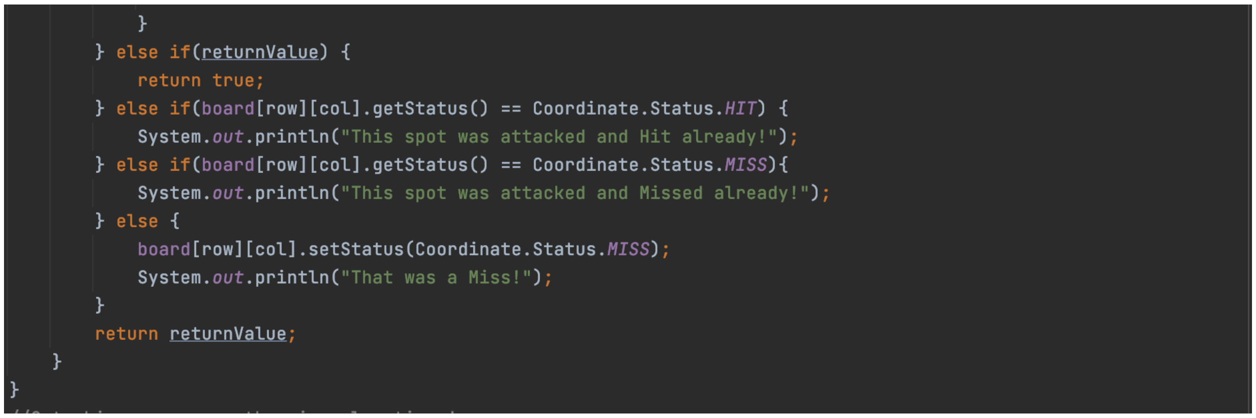
```java
public boolean getAttacked(int row, int col, Weapon weapon) {
    if (weapon.getWeaponType() == "stopper") {
        disabled = true;
        return true;
    }
```



```java
else if(weapon.getWeaponType() == "sonar_pulse") {
    //Check if there are sonar pulses left
    if(sonarPulsesLeft <= 0) {
        System.out.println("You are out of sonar pulses!");
        return false;
    }
    //Check if at least one ship has been destroyed yet
    if(!firstShipSunk) {
        System.out.println("You need to destroy a ship first!");
        return false;
    }
    String sonarBoard[][] = new String[10][10];
    for( int i=0; i < 10; i++) {
        for (int j = 0; j < 10; j++) {
            sonarBoard[i][j] = "0";
        }
    }
    for(int i = row-1; i <= row+1; i++){...}
    if(col-2 >= 0) { //This is the top most point of the diamond sonar pulse
        sonarBoard[row][col-2] = board[row][col-2].toString();
    }
    if(row-2 >= 0) {
        sonarBoard[row-2][col] = board[row-2][col].toString();
    }
    if(col+2 >= 0) {
        sonarBoard[row][col+2] = board[row][col+2].toString();
    }
    if(row+2 >= 0) {
        sonarBoard[row+2][col] = board[row+2][col].toString();
    }
    getSonarMaps(sonarBoard);
    sonarPulsesLeft--;
```

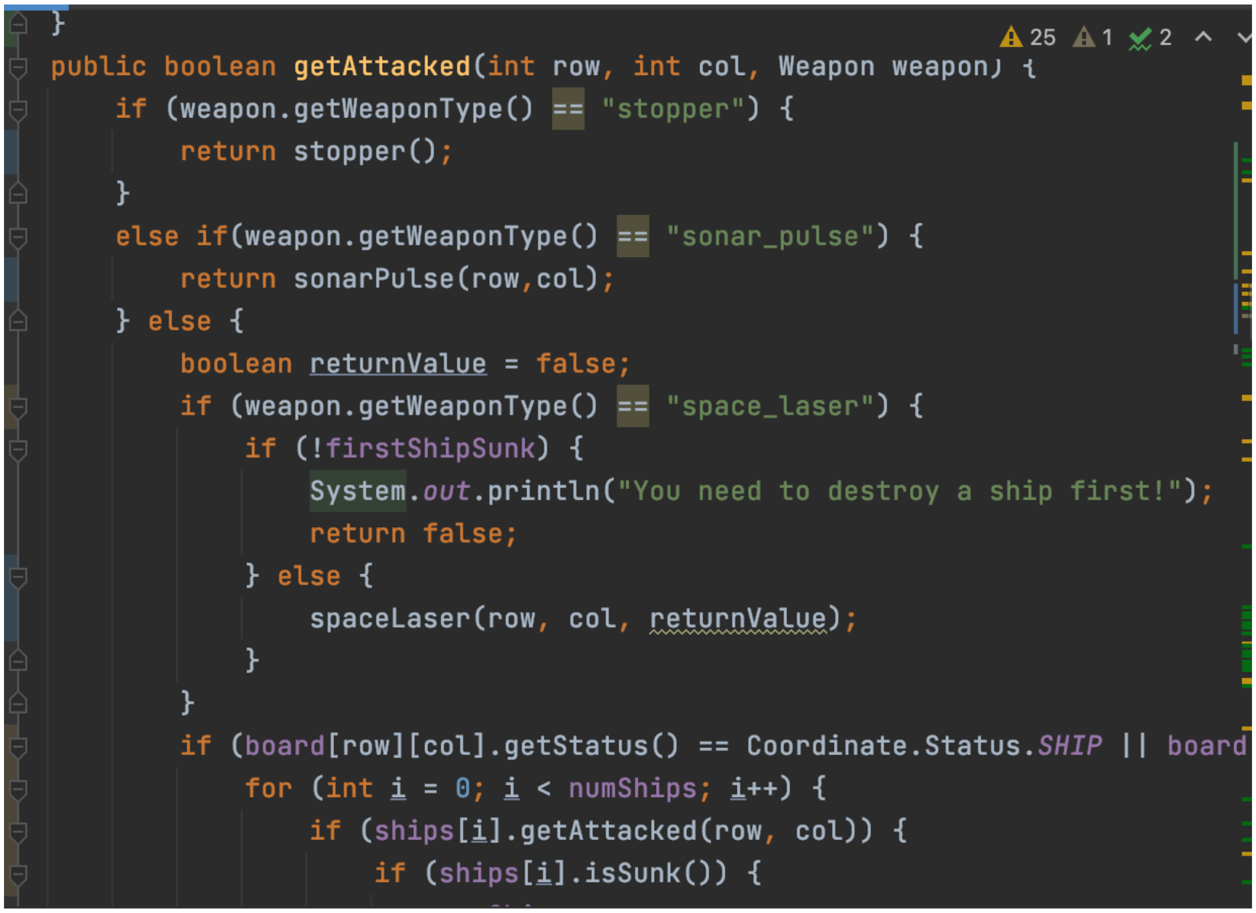


```
        return true;
    } else {
        boolean returnValue = false;
        if(weapon.getWeaponType() == "space_laser") {
            if (!firstShipSunk) {
                System.out.println("You need to destroy a ship first!");
                return false;
            }
            else {
                if(board[row][col].getBelowSurfaceStatus() == Coordinate.Status.SHIP || board[row][col].getBelowSurfaceStatus() == Coordinate.Status.CAPTAI
                    for(int i = 0; i < numShips; i++) {
                        if(ships[i].getAttackedBelow(row,col)) {
                            if(ships[i].isSunk()){
                                numShips--;
                            }
                            returnValue = true;
                        }
                    }
                }
            }
        }
        if(board[row][col].getStatus() == Coordinate.Status.SHIP || board[row][col].getStatus() == Coordinate.Status.CAPTAINQ || board[row][col].getStatus()
            for(int i = 0; i < numShips; i++) {
                if(ships[i].getAttacked(row,col)) {
                    if(ships[i].isSunk()){
                        numShips--;
                        firstShipSunk = true;
                    }
                    returnValue = true;
                }
            }
```



```
            }
        } else if(returnValue) {
            return true;
        } else if(board[row][col].getStatus() == Coordinate.Status.HIT) {
            System.out.println("This spot was attacked and Hit already!");
        } else if(board[row][col].getStatus() == Coordinate.Status.MISS){
            System.out.println("This spot was attacked and Missed already!");
        } else {
            board[row][col].setStatus(Coordinate.Status.MISS);
            System.out.println("That was a Miss!");
        }
        return returnValue;
    }
}
```

AFTER REFACTORING



25 1 2

```
}
public boolean getAttacked(int row, int col, Weapon weapon) {
    if (weapon.getWeaponType() == "stopper") {
        return stopper();
    }
    else if(weapon.getWeaponType() == "sonar_pulse") {
        return sonarPulse(row,col);
    } else {
        boolean returnValue = false;
        if (weapon.getWeaponType() == "space_laser") {
            if (!firstShipSunk) {
                System.out.println("You need to destroy a ship first!");
                return false;
            } else {
                spaceLaser(row, col, returnValue);
            }
        }
        if (board[row][col].getStatus() == Coordinate.Status.SHIP || board
            for (int i = 0; i < numShips; i++) {
                if (ships[i].getAttacked(row, col)) {
                    if (ships[i].isSunk()) {
```