46 src/test/java/edu/colorado/fourdimensionalonedgames/ShipTest.java

```
@@ -179,7 +179,7 @@ void testSubmarineGenerateCoordinates(){
            //for the weird submarine protrusion tile
            expected.add(new Point3D(8,7,0));

-           for(int i = 0; i < player2Submarine.size; i++){
+           for(int i = 0; i < player2Submarine.getSize(); i++){
                assertEquals(expected.get(i), results.get(i));
            }

@@ -194,7 +194,7 @@ void testSubmarineGenerateCoordinates(){
            //damage Submarine's CQ (sub at 2,2 down) (CQ at 2,5)
            player1.addWeapon(spaceLaser);
            fireInput1 = new PlayerFireInput("Space Laser", "2", "5");
            results = player1.attack(player2.getBoard(), fireInput1);
            result = results.get(0);
-           captainsQ = (CaptainsQuartersTile)result.ship.getShipTiles().get(3);
+           captainsQ = (CaptainsQuartersTile)result.getShip().getShipTiles().get(3);
            assertEquals(captainsQ.getHp(), 1);
-           assertFalse(result.ship.destroyed());
-           assertSame(result.type, AttackResultType.MISS);
+           assertFalse(result.getShip().destroyed());
+           assertSame(result.getType(), AttackResultType.MISS);

            //destroy entire Submarine after hitting CQ one more time
            results = player1.attack(player2.getBoard(), fireInput1);
            result = results.get(0);
-           assertTrue(result.ship.destroyed());
+           assertTrue(result.getShip().destroyed());
            assertEquals(captainsQ.getHp(), 0);
-           assertSame(result.type, AttackResultType.SUNK);
+           assertSame(result.getType(), AttackResultType.SUNK);
        }

    }

-       for(int i = 0; i < player2Submarine.size; i++){
+       for(int i = 0; i < player2Submarine.getSize(); i++){
            assertEquals(expected.get(i), results.get(i));
        }

@@ -244,7 +244,7 @@ void testLinearShipsGenerateCoordinates() {
            expected.add(new Point3D(9, y, 0));
        }

-       for (int i = 0; i < player2Battleship.size; i++) {
+       for (int i = 0; i < player2Battleship.getSize(); i++) {
            assertEquals(expected.get(i), results.get(i));
        }
    }
@@ -266,61 +266,61 @@ void testCaptainQuarters(){
        fireInput1 = new PlayerFireInput("Single Shot", "1", "1");
        List<AttackResult> results = player1.attack(player2.getBoard(), fireInput1);
        AttackResult result = results.get(0);
-       assertTrue(result.ship.destroyed());
-       CaptainsQuartersTile captainsQ = (CaptainsQuartersTile)result.ship.getShipTiles().get(0);
+       assertTrue(result.getShip().destroyed());
+       CaptainsQuartersTile captainsQ = (CaptainsQuartersTile)result.getShip().getShipTiles().get(0);
        assertEquals(captainsQ.getHp(), 0);
-       assertSame(result.type, AttackResultType.SUNK);


        //damage Destroyer's CQ
        fireInput1 = new PlayerFireInput("Single Shot", "4", "5");
        results = player1.attack(player2.getBoard(), fireInput1);
        result = results.get(0);
-       assertFalse(result.ship.destroyed());
-       captainsQ = (CaptainsQuartersTile)result.ship.getShipTiles().get(1);
+       assertFalse(result.getShip().destroyed());
+       captainsQ = (CaptainsQuartersTile)result.getShip().getShipTiles().get(1);
        assertEquals(captainsQ.getHp(), 1);
-       assertSame(result.type, AttackResultType.MISS);

        //destroy entire Destroyer after hitting CQ one more time
        results = player1.attack(player2.getBoard(), fireInput1);
        result = results.get(0);
-       assertTrue(result.ship.destroyed());
+       assertTrue(result.getShip().destroyed());
        assertEquals(captainsQ.getHp(), 0);
-       assertSame(result.type, AttackResultType.SUNK);


        //damage Battleship's CQ
        fireInput1 = new PlayerFireInput("Single Shot", "5", "7");
        results = player1.attack(player2.getBoard(), fireInput1);
        result = results.get(0);
-       captainsQ = (CaptainsQuartersTile)result.ship.getShipTiles().get(2);
+       captainsQ = (CaptainsQuartersTile)result.getShip().getShipTiles().get(2);
        assertEquals(captainsQ.getHp(), 1);
-       assertFalse(result.ship.destroyed());
-       assertSame(result.type, AttackResultType.MISS);

        //destroy entire Battleship after hitting CQ one more time
        results = player1.attack(player2.getBoard(), fireInput1);
        result = results.get(0);
-       assertTrue(result.ship.destroyed());
+       assertTrue(result.getShip().destroyed());
        assertEquals(captainsQ.getHp(), 0);
-       assertSame(result.type, AttackResultType.SUNK);


        //damage Submarine's CQ (sub at 2,2 down) (CQ at 2,5)
        player1.addWeapon(spaceLaser);
        fireInput1 = new PlayerFireInput("Space Laser", "2", "5");
        results = player1.attack(player2.getBoard(), fireInput1);
        result = results.get(0);
-       captainsQ = (CaptainsQuartersTile)result.ship.getShipTiles().get(3);
+       captainsQ = (CaptainsQuartersTile)result.getShip().getShipTiles().get(3);
        assertEquals(captainsQ.getHp(), 1);
-       assertFalse(result.ship.destroyed());
-       assertSame(result.type, AttackResultType.MISS);

        //destroy entire Submarine after hitting CQ one more time
        results = player1.attack(player2.getBoard(), fireInput1);
        result = results.get(0);
-       assertTrue(result.ship.destroyed());
+       assertTrue(result.getShip().destroyed());
        assertEquals(captainsQ.getHp(), 0);
-       assertSame(result.type, AttackResultType.SUNK);
    }

}
```

3 src/main/java/edu/colorado/fourdimensionalonedgames/game/ship/Ship.java

```
@@ -11,7 +11,7 @@
 
 public abstract class Ship implements IRenderable {
 
-    public int size;
     protected List<ShipTile> shipTiles;
     private boolean destroyed;
 
@@ -23,6 +23,7 @@ public Ship() {
     public void addTile(ShipTile tile) {
         shipTiles.add(tile);
     }
+    public int getSize(){return size;}
 
     /**
      * Calculate the damage done to the ship
```

```
 
 public abstract class Ship implements IRenderable {
 
+    protected int size;
     protected List<ShipTile> shipTiles;
     private boolean destroyed;
 
     public void addTile(ShipTile tile) {
         shipTiles.add(tile);
     }
+    public int getSize(){return size;}
 
     /**
      * Calculate the damage done to the ship
```

2 src/test/java/edu/colorado/fourdimensionalonedgames/BasicAttacksTest.java

```
@@ -118,7 +118,7 @@ void attackHit() {
         AttackResult expected = new AttackResult(AttackResultType.HIT, player2Minesweeper);
         // assertions after one hit
         assertEquals(expected, result);
-        assertEquals(1, result.ship.damage());
 
         // sink minesweeper ship
         fireInput1 = new PlayerFireInput("Single Shot", "1", "1");
```

```
         AttackResult expected = new AttackResult(AttackResultType.HIT, player2Minesweeper);
         // assertions after one hit
         assertEquals(expected, result);
+        assertEquals(1, result.getShip().damage());
 
         // sink minesweeper ship
         fireInput1 = new PlayerFireInput("Single Shot", "1", "1");
```

4 src/main/java/edu/colorado/fourdimensionalonedgames/game/ship/FleetControl.java

```
@@ -6,8 +6,8 @@
 
 //every turn, we will spawn a new FleetControl object which will have a "fresh" empty stack
 public class FleetControl {
-    Stack<MoveFleetCommand> fleetCommandStack;
-    Player player;
 
     public FleetControl(Player player){
         this.player = player;
```

```
 
 //every turn, we will spawn a new FleetControl object which will have a "fresh" empty stack
 public class FleetControl {
+    private Stack<MoveFleetCommand> fleetCommandStack;
+    private Player player;
 
     public FleetControl(Player player){
         this.player = player;
```

4 src/main/java/edu/colorado/fourdimensionalonedgames/game/ship/MoveFleetCommand.java

```
@@ -5,8 +5,8 @@
 public class MoveFleetCommand implements Command{
 
 
-    Orientation moveDirection;
-    Player player;
 
     public MoveFleetCommand(Player player, Orientation direction){
         this.player = player;
```

```
 public class MoveFleetCommand implements Command{
 
 
+    private Orientation moveDirection;
+    private Player player;
 
     public MoveFleetCommand(Player player, Orientation direction){
         this.player = player;
```

2 src/main/java/edu/colorado/fourdimensionalonedgames/game/Player.java

```
@@ -101,7 +101,7 @@ private void generateWeapons(){
         enemyBoardGui.updateObservers();
 
         for(AttackResult attackResult : results){
-            Ship attackedShip = attackResult.ship;
 
             if (attackedShip == null) {
                 //when missed shot
```

```
         enemyBoardGui.updateObservers();
 
         for(AttackResult attackResult : results){
+            Ship attackedShip = attackResult.getShip();
 
             if (attackedShip == null) {
                 //when missed shot
```

7 src/main/java/edu/colorado/fourdimensionalonedgames/game/attack/AttackResult.java

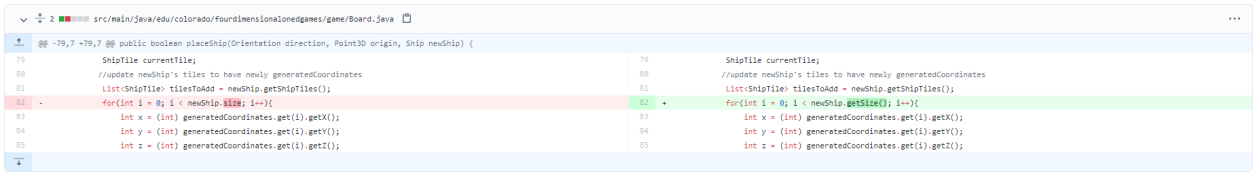
```
@@ -3,14 +3,17 @@
 import edu.colorado.fourdimensionalonedgames.game.ship.*;
 
 public class AttackResult {
-    public final AttackResultType type;
-    public final Ship ship;
 
     public AttackResult(AttackResultType type, Ship ship) {
         this.type = type;
         this.ship = ship;
     }
 
     @Override
     public boolean equals(Object o) {
         if (!(o instanceof AttackResult)) return false;
```

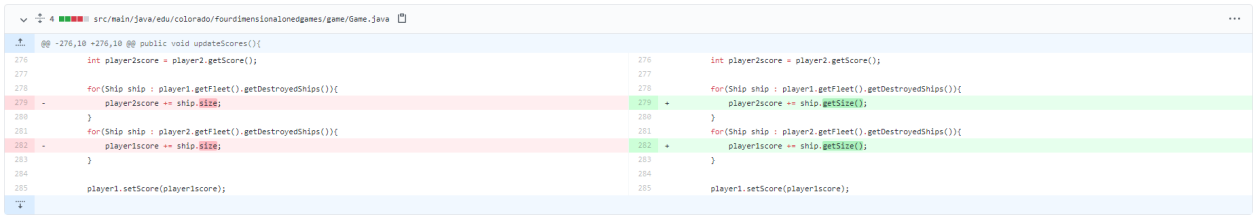
```
 import edu.colorado.fourdimensionalonedgames.game.ship.*;
 
 public class AttackResult {
+    private final AttackResultType type;
+    private final Ship ship;
 
     public AttackResult(AttackResultType type, Ship ship) {
         this.type = type;
         this.ship = ship;
     }
 
+    public Ship getShip(){ return ship; }
+    public AttackResultType getType(){ return type; }
+
     @Override
     public boolean equals(Object o) {
         if (!(o instanceof AttackResult)) return false;
```

2 src/main/java/edu/colorado/fourdimensionalonedgames/PlayerController.java

```
@@ -148,7 +148,7 @@ public void fireWeapon(Board opponentBoard, PlayerFireInput userInput){
 
             //Inform player of result of attacking each enemy tile (could be 1 tile or many)
             for(AttackResult attackResult : results){
-                Ship attackedShip = attackResult.ship;
 
                 if (attackedShip == null) {
                     AlertBox.display("Miss", "Shot missed");
```

```
 
             //Inform player of result of attacking each enemy tile (could be 1 tile or many)
             for(AttackResult attackResult : results){
+                Ship attackedShip = attackResult.getShip();
 
                 if (attackedShip == null) {
                     AlertBox.display("Miss", "Shot missed");
```





BEFORE OBSERVER PATTERN:

```
79      public boolean placeShip(GridPane currentBoard, Orientation direction, Point2D origin, Ship newShip) {
80          List<Point2D> newCoordinates = new ArrayList<>();
81
82          double xCoordinate = origin.getX();
83          double yCoordinate = origin.getY();
84
85          // get coordinate set of tiles ship would occupy if placed in given orientation
86          switch (direction) {
87              case up:
88                  for(double y = yCoordinate; y > (yCoordinate - newShip.size); y--) {
89                      newCoordinates.add(new Point2D(xCoordinate, y));
90                  }
91                  break;
92
93              case down:
94                  for(double y = yCoordinate; y < (yCoordinate + newShip.size); y++){
95                      newCoordinates.add(new Point2D(xCoordinate, y));
96                  }
97                  break;
98
99              case left:
100                 for(double x = xCoordinate; x > (xCoordinate - newShip.size); x--){
101                     newCoordinates.add(new Point2D(x, yCoordinate));
102                 }
103                 break;
104             case right:
105                 for(double x = xCoordinate; x < (xCoordinate + newShip.size); x++){
106                     newCoordinates.add(new Point2D(x, yCoordinate));
107                 }
108                 break;
109         }
110
```

```
111         // check each coordinate to make sure not off board or occupied by other ship
112         for (Point2D coordinate : newCoordinates) {
113             if (coordinate.getX() < 1) return false;
114             if (coordinate.getX() > columns) return false;
115             if (coordinate.getY() < 1) return false;
116             if (coordinate.getY() > rows) return false;
117 
118             Tile oldTile = tiles[(int) coordinate.getX()][(int) coordinate.getY()];
119             if (oldTile instanceof ShipTile) return false;
120         }
121 
122         // if verified that placement is valid, add ship tiles to board
123         ShipTile newTile;
124         for (Point2D coordinate : newCoordinates) {
125             int x = (int) coordinate.getX();
126             int y = (int) coordinate.getY();
127             newTile = new ShipTile(newShip, x, y);
128 
129             Tile oldTile = tiles[x][y];
130             renderer.unregister(oldTile);
131 
132             renderer.register(newTile);
133 
134             currentBoard.getChildren().remove(oldTile);
135             currentBoard.add(newTile, x, y);
136 
137             tiles[x][y] = newTile;
138             newShip.addTile(newTile);
139         }
140 
141         fleet.addShip(newShip);
142 
143         return true;
144     }
```

AFTER OBSERVER PATTERN:

```
1  package edu.colorado.fourdimensionalonedgames.render.gui;
2 
3  /**boards
4  public interface Subject {
5      void registerObserver(Observer observer);
6      void removeObserver(Observer observer);
7      void updateObservers();
8  }
9 
```

```java
package edu.colorado.fourdimensionalonedgames.game;

import edu.colorado.fourdimensionalonedgames.game.ship.Orientation;
import edu.colorado.fourdimensionalonedgames.render.Render;
import edu.colorado.fourdimensionalonedgames.game.ship.Ship;
import edu.colorado.fourdimensionalonedgames.render.gui.Observer;
import edu.colorado.fourdimensionalonedgames.render.gui.Subject;
import edu.colorado.fourdimensionalonedgames.render.tile.*;
import javafx.geometry.Point3D;
import javafx.scene.layout.GridPane;

import java.util.ArrayList;
import java.util.List;

public class Board implements Subject {

    private final int rows;
    private final int columns;
    private final int depth;

    public Tile[][][] tiles;
    public Render renderer;

    private List<Observer> observers;

    public Board(int columns, int rows, int depth, Render renderer) {

        this.rows = rows;
        this.columns = columns;
        this.depth = depth;
```

```java
package edu.colorado.fourdimensionalonedgames.render.gui;

import edu.colorado.fourdimensionalonedgames.render.tile.Tile;

//grid panes
public interface Observer {
    void update(Tile[][][] tiles);
}
```

```java
package edu.colorado.fourdimensionalonedgames.render.gui;

import ...

public class Display implements Observer{
    protected GridPane gpane;
    protected Tile[][][] boardState;
    protected Render renderer;

    //Send initial board to display
    public Display(GridPane gpane, Tile[][][] board, Render renderer){
        this.boardState = board;
        this.renderer = renderer;
        this.gpane = gpane;
    }

    @Override
    public void update(Tile[][][] newBoardState) {
        //unregister and reregister tiles to renderer
        //remove and add children to gpane

        for(Tile[][] tileColumn : newBoardState){
            for(Tile[] tileRow : tileColumn){
                swapTile(tileRow[0]);
            }
        }

        boardState = newBoardState;
    }
```

```java
import edu.colorado.fourdimensionalonedgames.render.tile.SeaTile;
import edu.colorado.fourdimensionalonedgames.render.tile.ShipTile;
import edu.colorado.fourdimensionalonedgames.render.tile.Tile;
import javafx.scene.layout.GridPane;

public class EnemyDisplay extends Display implements Observer{

    //Send initial board to display
    public EnemyDisplay(GridPane gpane, Tile[][][] board, Render renderer) { super(gpane, board, renderer)

    @Override
    public void update(Tile[][][] newBoardState) {
        //unregister and reregister tiles to renderer
        //remove and add children to gpane

        for(Tile[][] tileColumn : newBoardState){
            for(Tile[] tileRow : tileColumn){
                Tile newTile = tileRow[0];
                if(newTile instanceof ShipTile){
                    if(newTile.revealed || newTile.shot){
                        swapTile(newTile);
                    }
                    else{
                        //swapTile(new SeaTile(newTile.getColumn(), newTile.getRow()));
                    }
                }
                else if(newTile instanceof SeaTile || newTile instanceof LetterTile){
                    swapTile(newTile);
                }
```