Milestone 6 Refactors + Code Smells

4Dimensional1D Games

Reducing "Long Parameter List" with a GameSettings Object:

Before refactor:

```
//Game constructor which creates the Player objects
/**
 * @param renderer (One renderer to contain all IRenderable objects)
 * @param numberofPlayers (number of players in the game)
 * @param tileSize (size of each tile in pixels
 * @param columns (number of columns on a single battleships board)
 * @param rows (number of rows on a single battleships board)
 */
private Game(Stage primaryStage, Render renderer, int numberofPlayers, int tileSize, int columns, int rows, int depth) throws IOException {

    //set all of our private game attributes
    this.renderer = renderer;
    this.tileSize = tileSize;
    this.columns = columns;
    this.rows = rows;
    this.depth = depth;
    this.width = columns*tileSize;
    this.height = columns*tileSize;
    this.numberofPlayers = numberofPlayers;
    this.player1Turn = false;
    this.gameState = GameState.player1_setup;

    //create the players for this game (and their boards in the process)
    for(int i = 0; i < this.numberofPlayers; i++){
        Player newPlayer = new Player( game: this, new Board(columns, rows, depth, this.renderer), new Board(columns, rows, depth, this.renderer));
        this.players.add(newPlayer);
    }
    startGame(primaryStage);
}
```

After refactor: *A GameSettings object is now passed in to game's constructor*

```
//Game constructor which creates the Player objects
/**
 * @param primaryStage (Stage which the application will display on user's computer)
 * @param renderer (One renderer to contain all IRenderable objects)
 * @param settings (Package of various game settings)
 */
private Game(Stage primaryStage, Render renderer, GameSettings settings) throws IOException {

    //set all of our private game attributes
    this.renderer = renderer;
    this.tileSize = settings.getTileSize();
    this.columns = settings.getColumns();
    this.rows = settings.getRows();
    this.depth = settings.getDepth();
    this.numberofPlayers = settings.getNumberOfPlayers();

    this.width = columns*tileSize;
    this.height = columns*tileSize;
    this.player1Turn = false;
    this.gameState = GameState.player1_setup;

    //create the players for this game (and their boards in the process)
    for(int i = 0; i < this.numberofPlayers; i++){
        Player newPlayer = new Player( game: this, new Board(columns, rows, depth, this.renderer), new Board(columns, rows, depth, this.renderer));
        this.players.add(newPlayer);
    }
    startGame(primaryStage);
}
```

Client Code after, creates a GameSettings object first, then passes that in as a param:

```
GameSettings settings = new GameSettings(numberOfPlayers, columns, rows, depth, tileSize);


//Create the game object
Game newGame = Game.getInstance(primaryStage, renderer, settings);
```

Reducing consecutive if statements for client to remove weapon:

Before refactor: *An if statement added for every weapon created… Bad for future change.*

```
//if sonar being used, remove a sonar object from player's list of weapons to decrement uses
if(weapon.getType().equals("Sonar Pulse")){
    removeWeapon( weaponToRemove: "Sonar Pulse");
}
```

Corresponding method call:

```
public void removeWeapon(String weaponToRemove){
    Weapon weaponToDelete = new SmallWeapon(new Attack(),Game.SINGLE_SHOT);
    for(Weapon weapon : this.weapons){
        if(weapon.getType().equals(weaponToRemove)){
            weaponToDelete = weapon;
        }
    }
    this.getWeapons().remove(weaponToDelete);
}
```

After weapon removal refactor:

Any weapon can now be removed at the client's leisure. No need for an additional if statement every time a weapon is added to the game

```
//if sonar being used, remove a sonar object from player's list of weapons to decrement uses
if(weapon.doRemove()){
    this.weapons.remove(weapon);
}
```

Weapons now all have an "AfterAttackBehavior" that determines what this .doRemove() call does for each type of weapon. Ie. The Single Shot weapon does not need to be removed after use, so it has a "NoAfterAttackBehavior" AfterAttackBehavior that does nothing, while counted weapons like Sonar Pulse, Cluster Bomb, Nuke, etc have a "PopCountAfterAttack" AfterAttackBehavior that decrements a count of that weapon, once the count his zero, then the doRemove() call returns true

```
public abstract class Weapon {
    protected final IAttackBehavior behavior;
    protected final IAfterAttackBehavior afterBehavior;
    protected final String weaponName;

    public Weapon(IAttackBehavior behavior, String weaponName) {
        this(behavior, weaponName, new NoAfterAttackBehavior());
    }

    public Weapon(IAttackBehavior behavior, String weaponName, IAfterAttackBehavior afterBehavior) {
        this.behavior = behavior;
        this.weaponName = weaponName;
        this.afterBehavior = afterBehavior;
    }

    public abstract List<AttackResult> useAt(Board board, Point2D position);

    public String getType() { return weaponName; }

    public boolean doRemove() {
        return afterBehavior.doRemove();
    }
}
```

The IAfterAttackBehavior interface, very simple

```
public interface IAfterAttackBehavior {
    public boolean doRemove();
}
```

Overriding the doRemove behavior in NoAfterAttackBehavior (just returns false)

```
public class NoAfterAttackBehavior implements IAfterAttackBehavior {
    @Override
    public boolean doRemove() { return false; }
}
```

Overriding the doRemove behavior in NoAfterAttackBehavior (just returns false)

```java
public class PopCountAfterAttackBehavior implements IAfterAttackBehavior {
    private int count;

    public PopCountAfterAttackBehavior(int count) {
        this.count = count;
    }

    @Override
    public boolean doRemove() {
        count--;
        return count <= 0;
    }

    public void addCount(int n) {
        count += n;
    }
}
```

Example creation of a concrete weapon object:

```java
//player gets 2 sonar pulses for the rest of the game to use
player.addWeapon(new LargeWeapon(new Reveal(), Game.SONAR_PULSE, new PopCountAfterAttackBehavior(2)));
```

Type 2 Clone switch statement refactor:

Before refactor:

Differences between switch cases highlighted in yellow:

```
switch (direction){
    case up:

        newOrigin = new Point3D(x, y: y-1,z);
        coords = ship.generateCoordinates(newOrigin,findOrientation(ship));
        for(ShipTile tile : ship.getShipTiles()){
            previousCoord = new Point3D(tile.getColumn(),  y: tile.getRow()+1, tile.getDepth());
            if(isWithinBounds(previousCoord)){
                previous = tiles[tile.getColumn()][tile.getRow()+1][tile.getDepth()];
                if(previous instanceof SeaTile){
                    tiles[tile.getColumn()][tile.getRow()][tile.getDepth()] = new SeaTile(tile.getColumn(), tile.getRow(), tile.getDepth());
                }
            }
            else{
                tiles[tile.getColumn()][tile.getRow()][tile.getDepth()] = new SeaTile(tile.getColumn(), tile.getRow(), tile.getDepth());
            }
        }

        break;

    case down:

        newOrigin = new Point3D(x, y: y+1,z);
        coords = ship.generateCoordinates(newOrigin,findOrientation(ship));

        for(ShipTile tile : ship.getShipTiles()){
            previousCoord = new Point3D(tile.getColumn(),  y: tile.getRow()-1, tile.getDepth());
            if(isWithinBounds(previousCoord)){
                previous = tiles[tile.getColumn()][tile.getRow()-1][tile.getDepth()];
                if(previous instanceof SeaTile){
                    tiles[tile.getColumn()][tile.getRow()][tile.getDepth()] = new SeaTile(tile.getColumn(), tile.getRow(), tile.getDepth());
                }
            }
            else{
                tiles[tile.getColumn()][tile.getRow()][tile.getDepth()] = new SeaTile(tile.getColumn(), tile.getRow(), tile.getDepth());
    case left:
        newOrigin = new Point3D( x: x-1,y,z);
        coords = ship.generateCoordinates(newOrigin,findOrientation(ship));
        for(ShipTile tile : ship.getShipTiles()){
            previousCoord = new Point3D( x: tile.getColumn()+1, tile.getRow(), tile.getDepth());
            if(isWithinBounds(previousCoord)){
                previous = tiles[tile.getColumn()+1][tile.getRow()][tile.getDepth()];
                if(previous instanceof SeaTile){
                    tiles[tile.getColumn()][tile.getRow()][tile.getDepth()] = new SeaTile(tile.getColumn(), tile.getRow(), tile.getDepth());
                }
            }
            else{
                tiles[tile.getColumn()][tile.getRow()][tile.getDepth()] = new SeaTile(tile.getColumn(), tile.getRow(), tile.getDepth());
            }
        }
        break;

    case right:
        newOrigin = new Point3D( x: x+1,y,z);
        coords = ship.generateCoordinates(newOrigin,findOrientation(ship));
        for(ShipTile tile : ship.getShipTi
            Point3D newOrigin = new Point3D(x, y, z)
            previousCoord = new Point3D( x
            if(isWithinBounds(previousCoord)){
                previous = tiles[tile.getColumn()-1][tile.getRow()][tile.getDepth()];
                if(previous instanceof SeaTile){
                    tiles[tile.getColumn()][tile.getRow()][tile.getDepth()] = new SeaTile(tile.getColumn(), tile.getRow(), tile.getDepth());
                }
            }
            else{
                tiles[tile.getColumn()][tile.getRow()][tile.getDepth()] = new SeaTile(tile.getColumn(), tile.getRow(), tile.getDepth());
```

After refactor: *All similar code out of the switch cases:*

```
int xChange = 0;
int yChange = 0;

switch (direction){
    case up:
        xChange = 0;
        yChange = -1;
        break;

    case down:
        xChange = 0;
        yChange = 1;
        break;

    case left:
        xChange = -1;
        yChange = 0;
        break;

    case right:
        xChange = 1;
        yChange = 0;
        break;
}

newOrigin = new Point3D( x: x+xChange, y: y+yChange,z);
coords = ship.generateCoordinates(newOrigin,findOrientation(ship));
for(ShipTile tile : ship.getShipTiles()){
    previousCoord = new Point3D( x: tile.getColumn()-xChange,  y: tile.getRow()-yChange, tile.getDepth());
    if(isWithinBounds(previousCoord)){
        previous = tiles[tile.getColumn()-xChange][tile.getRow()-yChange][tile.getDepth()];
        if(previous instanceof SeaTile){
            tiles[tile.getColumn()][tile.getRow()][tile.getDepth()] = new SeaTile(tile.getColumn(), tile.getRow(), tile.getDepth());
        }
    }
    else{
        tiles[tile.getColumn()][tile.getRow()][tile.getDepth()] = new SeaTile(tile.getColumn(), tile.getRow(), tile.getDepth());
    }
}
```

Weird "if" statements both having a return statement

Fixing a poorly thought out "hacky" solution

Before refactor:

```
public void handlePassTurnButton(ActionEvent e){

    if(game.getGameState() == GameState.player1_setup){
        game.setGameState(GameState.player2_setup);
        showCombatButtons();
        game.passSetupTurn();
        return;
    }

    if(game.getGameState() == GameState.player2_setup){
        game.setGameState(GameState.first_turn);
        showCombatButtons();
        game.passTurn();
        game.updateScene();
        return;
    }

    //Turn is over when button is pressed
    this.game.passTurn();
    this.game.updateScene();
}
```

After refactor, swapped if's and added an else. Both return statements now gone:

```
public void handlePassTurnButton(ActionEvent e){
    //one time if statement for passing to player 1's first combat turn
    if(game.getGameState() == GameState.player2_setup){
        game.setGameState(GameState.first_turn);
        showCombatButtons();
        game.passTurn();
        game.updateScene();
    }

    //one time if statement for passing to player 2's setup turn
    if(game.getGameState() == GameState.player1_setup){
        game.setGameState(GameState.player2_setup);
        showCombatButtons();
        game.passSetupTurn();
    }

    //normal game pass turn conditions
    else{
        //Turn is over when button is pressed
        this.game.passTurn();
        this.game.updateScene();
    }
}
```

MoveFleet method size refactor (method was 100+ lines of code)

Before refactor:

```
215 @    public boolean moveFleet(Orientation direction){
216          // check for border collision
217          switch (direction){
218              case up:
219                  for (Ship ship : fleet.getShips()){
220                      for (ShipTile tile : ship.getShipTiles()){
221                          if (tile.getRow() - 1 < 1){
222                              return false;
223                          }
224                      }
225                  }
226                  break;
227              case down:
228                  for (Ship ship : fleet.getShips()){
229                      for (ShipTile tile : ship.getShipTiles()){
230                          if (tile.getRow() + 1 > 10){
231                              return false;
232                          }
233                      }
234                  }
235                  break;
236              case right:
237                  for (Ship ship : fleet.getShips()){
238                      for (ShipTile tile : ship.getShipTiles()){
239                          if (tile.getColumn() + 1 > 10){
240                              return false;
241                          }
242                      }
243                  }
244                  break;
245              case left:
246                  for (Ship ship : fleet.getShips()){
247                      for (ShipTile tile : ship.getShipTiles()){
248                          if (tile.getColumn() - 1 < 1){
249                              return false;
250                          }
251                      }
252                  }
253                  break;
```

```
254             default:
255                 break;
256         }
257
258         // actually move the fleet now that we confirmed it can be moved
259         List<Weapon> weaponsToAdd = new ArrayList<>();
260         for (Ship ship : fleet.getShips()){
261             //dead ships should not move
262             if(ship.destroyed()){
263                 continue;
264             }
265             weaponsToAdd.addAll(board.moveShip(ship, direction));
266         }
267         //update GUI to show the ships moved
268         board.updateLocalObservers();
269         //now add all weapons that user may have picked up and display AlertBoxes for each one picked up
270         for (Weapon weapon : weaponsToAdd) {
271             boolean isInAlreadyExistingWeapons = false;
272             for (Weapon weapon2 : weapons) {
273                 if (weapon2.getType().equals(weapon.getType())) {
274                     isInAlreadyExistingWeapons = true;
275                     weapon2.addCount(weapon.getCount());
276                     if(!game.isTestMode()){
277                         AlertBox.display( title: "Power Up Acquired", message: "You just picked up another " + weaponsToAdd.get(0).getType()
278                                 weaponsToAdd.get(0).getCount()+" left");
279                     }
280                 }
281             }
282             if (!isInAlreadyExistingWeapons) {
283                 weapons.add(weapon);
284                 if(!game.isTestMode()){
285                     AlertBox.display( title: "Power Up Acquired", message: "You just picked up a " + weaponsToAdd.get(0).getType() + "! Use
286                 }
287             }
288         }
289         return true;
290     }
```

After refactor:

moveFleet() is now less than 50 lines of code with new it's new borderCollision() call highlighted

```
215     public boolean moveFleet(Orientation direction){
216         // check for border collision on all ships in the fleet
217         if(borderCollision(direction)){
218             return false;
219         }
220         else{
221             // actually move the fleet now that we confirmed it can be moved
222             List<Weapon> weaponsToAdd = new ArrayList<>();
223             for (Ship ship : fleet.getShips()){
224                 //dead ships should not move
225                 if(ship.destroyed()){
226                     continue;
227                 }
228                 weaponsToAdd.addAll(board.moveShip(ship, direction));
229             }
230             //update GUI to show the ships moved
231             board.updateLocalObservers();
232             //now add all weapons that user may have picked up and display AlertBoxes for each one picked up
233             for (Weapon weapon : weaponsToAdd) {
234                 boolean isInAlreadyExistingWeapons = false;
235                 for (Weapon weapon2 : weapons) {
236                     if (weapon2.getType().equals(weapon.getType())) {
237                         isInAlreadyExistingWeapons = true;
238                         weapon2.addCount(weapon.getCount());
239                         if(!game.isTestMode()){
240                             AlertBox.display( title: "Power Up Acquired",  message: "You just picked up another " + weaponsToAdd.get(0)
241                                     weaponsToAdd.get(0).getCount()+" left");
242                         }
243                     }
244                 }
245                 if (!isInAlreadyExistingWeapons) {
246                     weapons.add(weapon);
247                     if(!game.isTestMode()){
248                         AlertBox.display( title: "Power Up Acquired",  message: "You just picked up a " + weaponsToAdd.get(0).getType()
249                     }
250                 }
251             }
252             return true;
253         }
254     }
```

New borderCollision() method created to assist moveFleet

```
256 @   private boolean borderCollision(Orientation direction){
257         switch (direction){
258             case up:
259                 for (Ship ship : fleet.getShips()){
260                     for (ShipTile tile : ship.getShipTiles()){
261                         if (tile.getRow() - 1 < 1){
262                             return false;
263                         }
264                     }
265                 }
266                 break;
267             case down:
268                 for (Ship ship : fleet.getShips()){
269                     for (ShipTile tile : ship.getShipTiles()){
270                         if (tile.getRow() + 1 > 10){
271                             return false;
272                         }
273                     }
274                 }
275                 break;
276             case right:
277                 for (Ship ship : fleet.getShips()){
278                     for (ShipTile tile : ship.getShipTiles()){
279                         if (tile.getColumn() + 1 > 10){
280                             return false;
281                         }
282                     }
```

“Feature Envy” refactor: Methods that make extensive use of another class (should be in that other class)

placeMines() and placePowerups() ***had no Player object dependencies*** *yet they were in the Player class. Furthermore, they had extensive calls to things in the Board class. Obvious solution: Move these methods to the Board. Identical change for the placePowerUps method*

Before refactor (sitting in Player class):

```
//method to actually place mines down on the board
public void placeMines(){
    Random random = new Random();
    int minesToPlace = 5;

    while(minesToPlace > 0){
        int i = random.nextInt( bound: 10) + 1;
        int j = random.nextInt( bound: 10) + 1;
        Tile oldTile = board.tiles[i][j][0];
        if (oldTile instanceof SeaTile){
            Tile mineTile = new MineTile(i,j, depth: 0);//START HERE
            board.tiles[i][j][0] = mineTile;
            minesToPlace--;
        }
    }
    board.updateLocalObservers();
}
```

After refactor (moved to Board class):

```
//method to actually place mines down on the board
public void placeMines(){
    Random random = new Random();
    int minesToPlace = 5;

    while(minesToPlace > 0){
        int i = random.nextInt( bound: 10) + 1;
        int j = random.nextInt( bound: 10) + 1;
        Tile oldTile = tiles[i][j][0];
        if (oldTile instanceof SeaTile){
            Tile mineTile = new MineTile(i,j, depth: 0);
            tiles[i][j][0] = mineTile;
            minesToPlace--;
        }
    }
    updateLocalObservers();
}
```