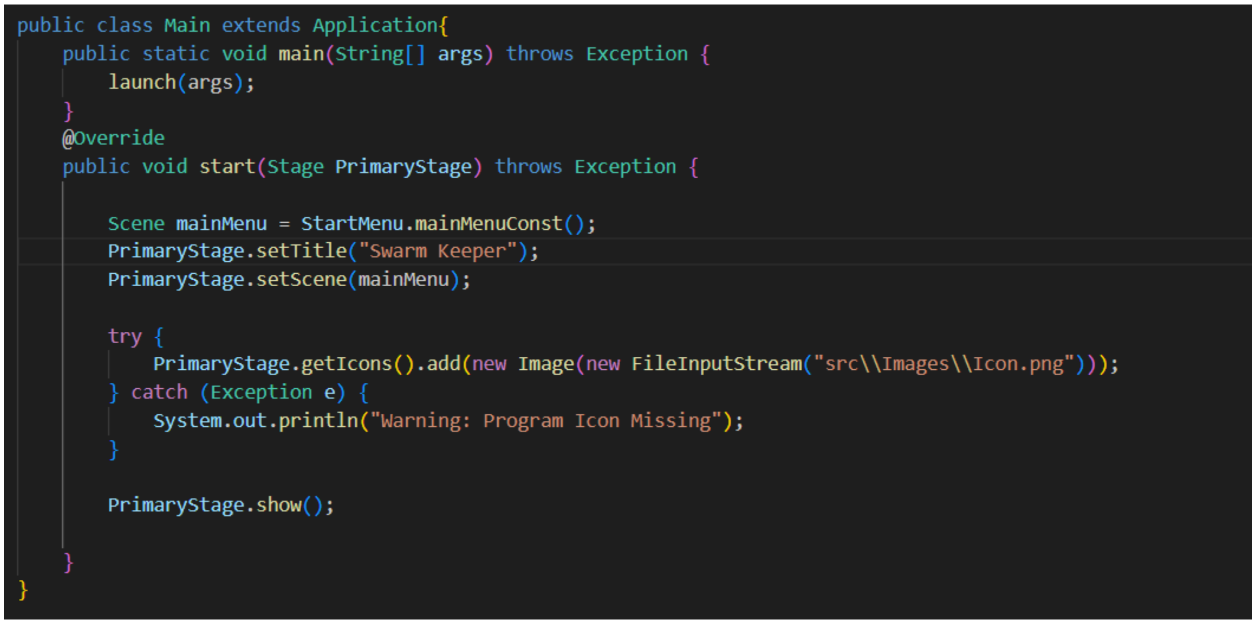
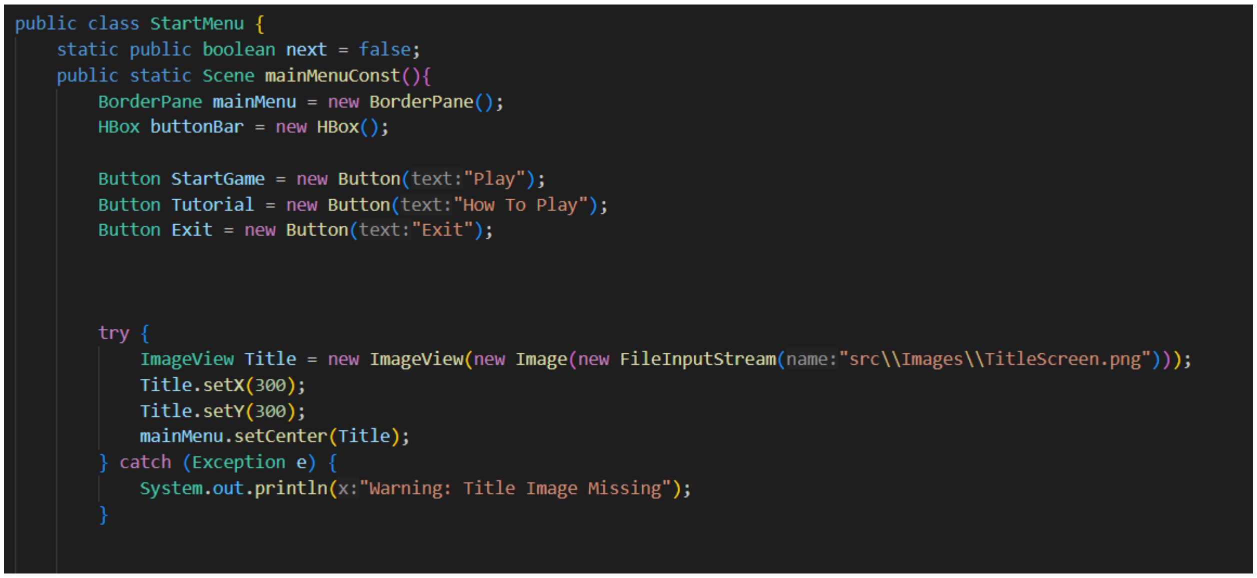
Main.java{



```java
public class Main extends Application{
    public static void main(String[] args) throws Exception {
        launch(args);
    }
    @Override
    public void start(Stage PrimaryStage) throws Exception {

        Scene mainMenu = StartMenu.mainMenuConst();
        PrimaryStage.setTitle("Swarm Keeper");
        PrimaryStage.setScene(mainMenu);

        try {
            PrimaryStage.getIcons().add(new Image(new FileInputStream("src\\Images\\Icon.png")));
        } catch (Exception e) {
            System.out.println("Warning: Program Icon Missing");
        }

        PrimaryStage.show();

    }
}
```

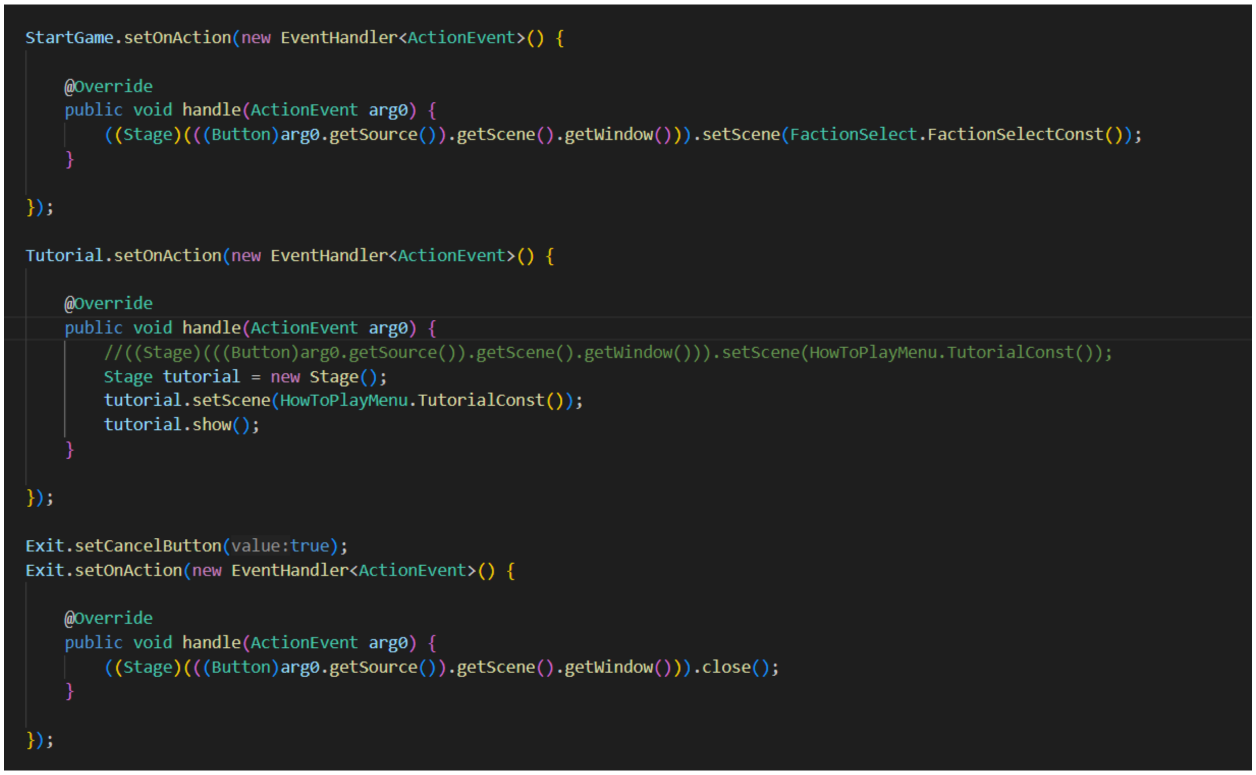
Does little on its own, sets the scene to the first menu screen, and sets the window title and icon(missing)
}

Menus[
StartMenu.java{



```java
public class StartMenu {
    static public boolean next = false;
    public static Scene mainMenuConst(){
        BorderPane mainMenu = new BorderPane();
        HBox buttonBar = new HBox();

        Button StartGame = new Button(text:"Play");
        Button Tutorial = new Button(text:"How To Play");
        Button Exit = new Button(text:"Exit");



        try {
            ImageView Title = new ImageView(new Image(new FileInputStream(name:"src\\Images\\TitleScreen.png")));
            Title.setX(300);
            Title.setY(300);
            mainMenu.setCenter(Title);
        } catch (Exception e) {
            System.out.println(x:"Warning: Title Image Missing");
        }

```

boolean next: This appears to be unused, but it's better to not risk removing it
public static Scene MainMenuConst{
Contains three buttons, as well as the panes to order them and a title screen image(missing)
Buttons:



```java
StartGame.setOnAction(new EventHandler<ActionEvent>() {

    @Override
    public void handle(ActionEvent arg0) {
        ((Stage)(((Button)arg0.getSource()).getScene().getWindow())).setScene(FactionSelect.FactionSelectConst());
    }

});

Tutorial.setOnAction(new EventHandler<ActionEvent>() {

    @Override
    public void handle(ActionEvent arg0) {
        //((Stage)(((Button)arg0.getSource()).getScene().getWindow())).setScene(HowToPlayMenu.TutorialConst());
        Stage tutorial = new Stage();
        tutorial.setScene(HowToPlayMenu.TutorialConst());
        tutorial.show();
    }

});

Exit.setCancelButton(value:true);
Exit.setOnAction(new EventHandler<ActionEvent>() {

    @Override
    public void handle(ActionEvent arg0) {
        ((Stage)(((Button)arg0.getSource()).getScene().getWindow())).close();
    }

});
```

-Exit: Closes the application

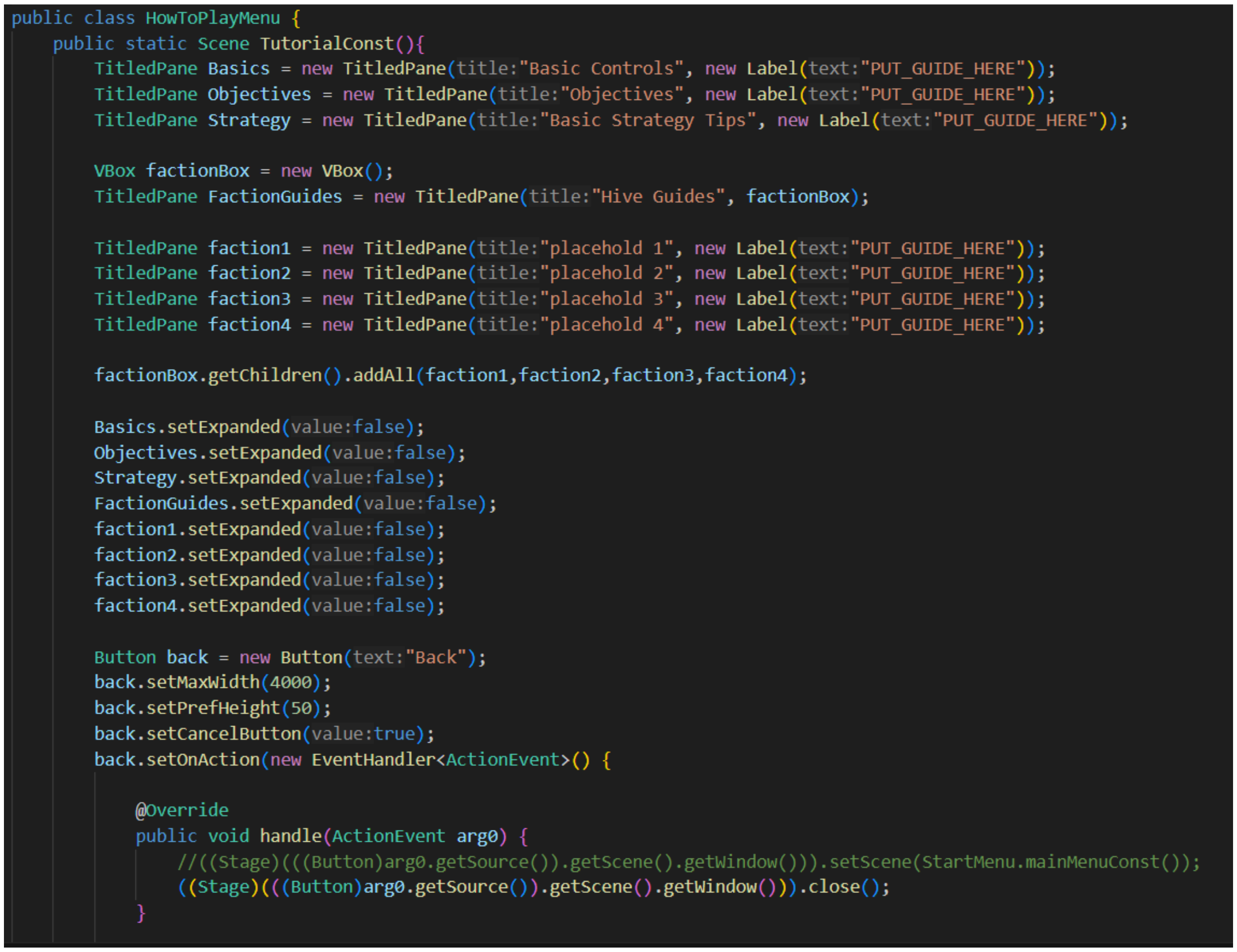
-Tutorial: Opens a new stage based on the HowToPlayMenu.TutorialConst() function

-StartGame: Changes the main menu scene to the one specified in FactionSelect.FactionSelectConst()

}

}

HowToPlayMenu.java{



```java
public class HowToPlayMenu {
    public static Scene TutorialConst(){
        TitledPane Basics = new TitledPane(title:"Basic Controls", new Label(text:"PUT_GUIDE_HERE"));
        TitledPane Objectives = new TitledPane(title:"Objectives", new Label(text:"PUT_GUIDE_HERE"));
        TitledPane Strategy = new TitledPane(title:"Basic Strategy Tips", new Label(text:"PUT_GUIDE_HERE"));

        VBox factionBox = new VBox();
        TitledPane FactionGuides = new TitledPane(title:"Hive Guides", factionBox);

        TitledPane faction1 = new TitledPane(title:"placehold 1", new Label(text:"PUT_GUIDE_HERE"));
        TitledPane faction2 = new TitledPane(title:"placehold 2", new Label(text:"PUT_GUIDE_HERE"));
        TitledPane faction3 = new TitledPane(title:"placehold 3", new Label(text:"PUT_GUIDE_HERE"));
        TitledPane faction4 = new TitledPane(title:"placehold 4", new Label(text:"PUT_GUIDE_HERE"));

        factionBox.getChildren().addAll(faction1,faction2,faction3,faction4);

        Basics.setExpanded(value:false);
        Objectives.setExpanded(value:false);
        Strategy.setExpanded(value:false);
        FactionGuides.setExpanded(value:false);
        faction1.setExpanded(value:false);
        faction2.setExpanded(value:false);
        faction3.setExpanded(value:false);
        faction4.setExpanded(value:false);

        Button back = new Button(text:"Back");
        back.setMaxWidth(4000);
        back.setPrefHeight(50);
        back.setCancelButton(value:true);
        back.setOnAction(new EventHandler<ActionEvent>() {

            @Override
            public void handle(ActionEvent arg0) {
                //((Stage)(((Button)arg0.getSource()).getScene().getWindow())).setScene(StartMenu.mainMenuConst());
                ((Stage)(((Button)arg0.getSource()).getScene().getWindow())).close();
            }
```

public static Scene TutorialConst(){

Contains various panes with labels to describe basic gameplay (may be unimplemented), as well as a single button

Button back: closes the window, allowing one to return to the main menu scene

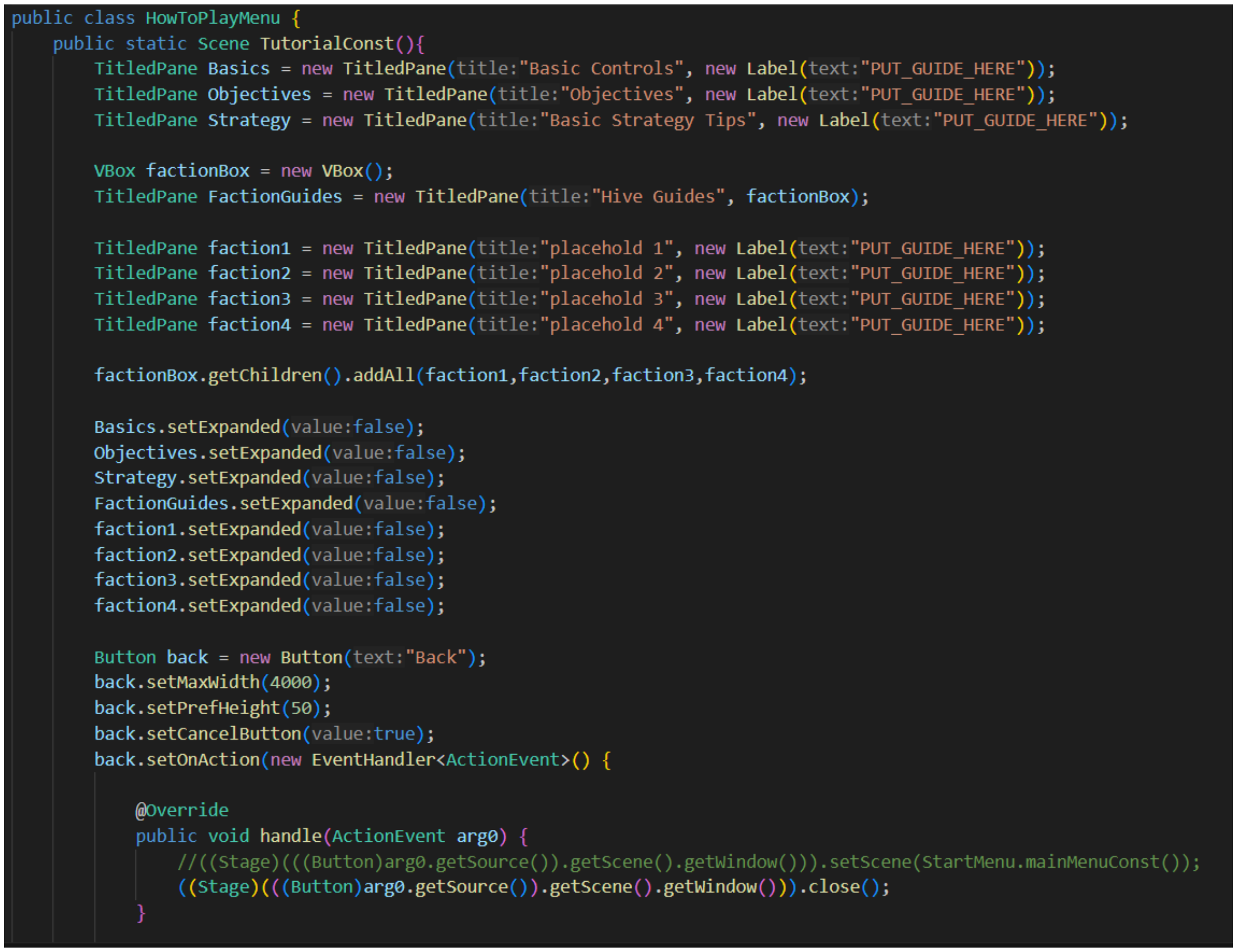
}

}

FactionSelect.java{

contains the booleans P1Selected and P2Selected, used to track whether players 1 and 2 respectively have selected a group to play

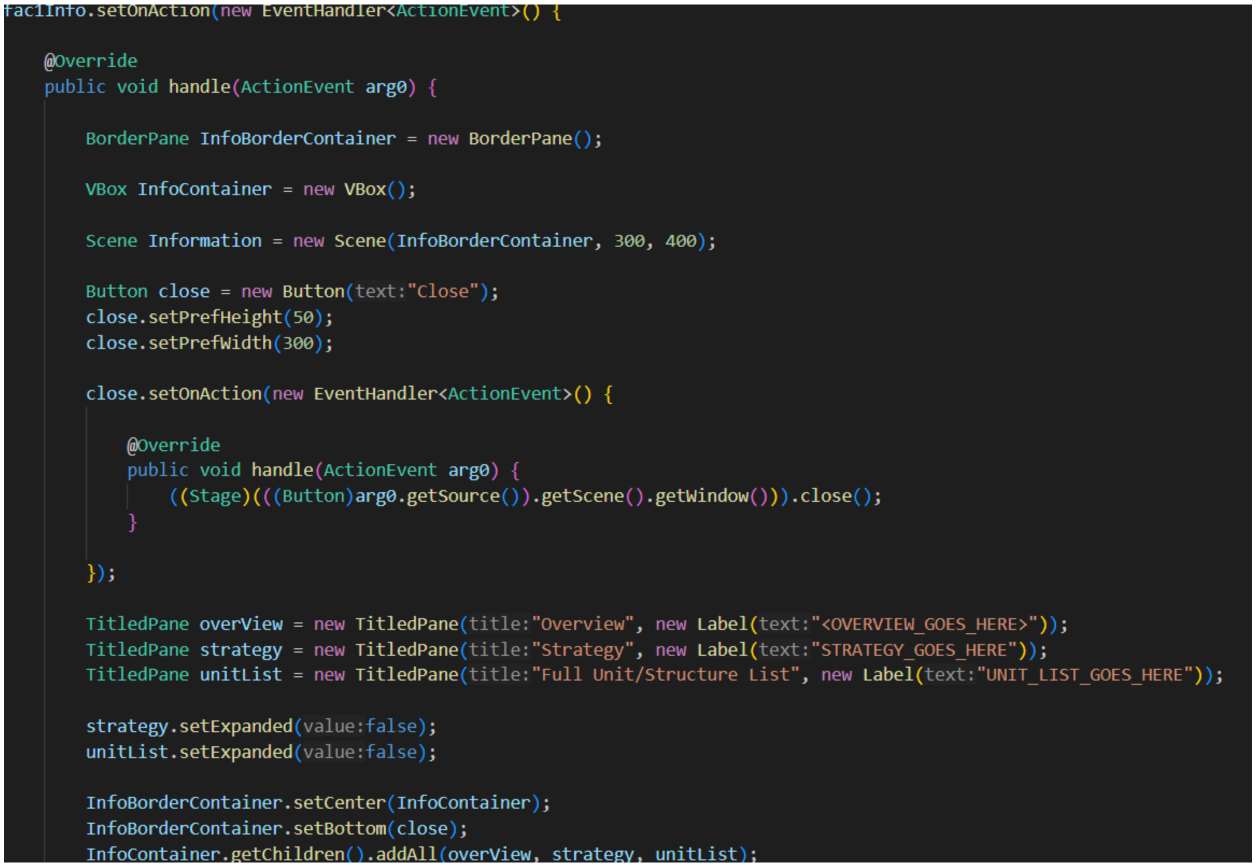
FactionSelect.FactionSelectConst.java(){



```java
public class HowToPlayMenu {
    public static Scene TutorialConst(){
        TitledPane Basics = new TitledPane(title:"Basic Controls", new Label(text:"PUT_GUIDE_HERE"));
        TitledPane Objectives = new TitledPane(title:"Objectives", new Label(text:"PUT_GUIDE_HERE"));
        TitledPane Strategy = new TitledPane(title:"Basic Strategy Tips", new Label(text:"PUT_GUIDE_HERE"));

        VBox factionBox = new VBox();
        TitledPane FactionGuides = new TitledPane(title:"Hive Guides", factionBox);

        TitledPane faction1 = new TitledPane(title:"placehold 1", new Label(text:"PUT_GUIDE_HERE"));
        TitledPane faction2 = new TitledPane(title:"placehold 2", new Label(text:"PUT_GUIDE_HERE"));
        TitledPane faction3 = new TitledPane(title:"placehold 3", new Label(text:"PUT_GUIDE_HERE"));
        TitledPane faction4 = new TitledPane(title:"placehold 4", new Label(text:"PUT_GUIDE_HERE"));

        factionBox.getChildren().addAll(faction1,faction2,faction3,faction4);

        Basics.setExpanded(value:false);
        Objectives.setExpanded(value:false);
        Strategy.setExpanded(value:false);
        FactionGuides.setExpanded(value:false);
        faction1.setExpanded(value:false);
        faction2.setExpanded(value:false);
        faction3.setExpanded(value:false);
        faction4.setExpanded(value:false);

        Button back = new Button(text:"Back");
        back.setMaxWidth(4000);
        back.setPrefHeight(50);
        back.setCancelButton(value:true);
        back.setOnAction(new EventHandler<ActionEvent>() {

            @Override
            public void handle(ActionEvent arg0) {
                //((Stage)(((Button)arg0.getSource()).getScene().getWindow())).setScene(StartMenu.mainMenuConst());
                ((Stage)(((Button)arg0.getSource()).getScene().getWindow())).close();
            }
```

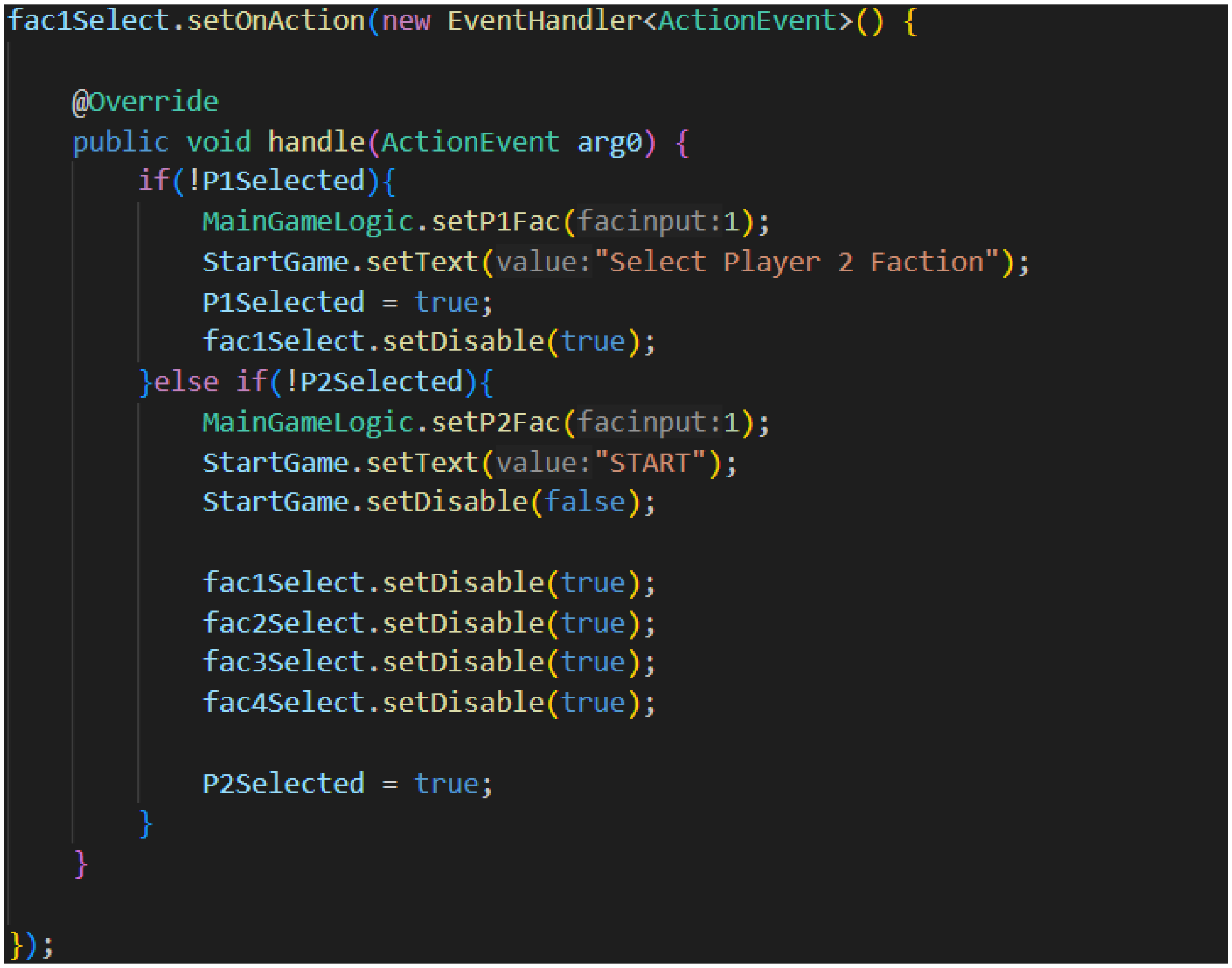
contains 9 buttons and 4 images(unimplemented)

Buttons can be divided into 3 groups, 4 are information buttons, 4 are selection buttons, and 1 is the start button



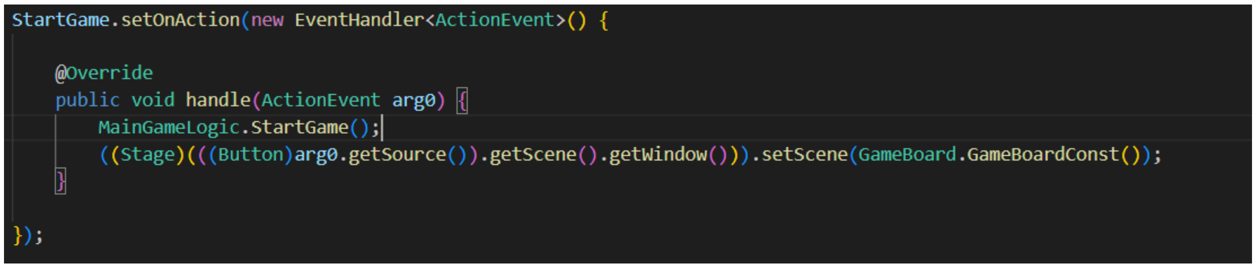
```
fac1Info.setOnAction(new EventHandler<ActionEvent>() {

    @Override
    public void handle(ActionEvent arg0) {

        BorderPane InfoBorderContainer = new BorderPane();

        VBox InfoContainer = new VBox();

        Scene Information = new Scene(InfoBorderContainer, 300, 400);

        Button close = new Button(text:"Close");
        close.setPrefHeight(50);
        close.setPrefWidth(300);

        close.setOnAction(new EventHandler<ActionEvent>() {

            @Override
            public void handle(ActionEvent arg0) {
                ((Stage)(((Button)arg0.getSource()).getScene().getWindow())).close();
            }

        });

        TitledPane overView = new TitledPane(title:"Overview", new Label(text:"<OVERVIEW_GOES_HERE>"));
        TitledPane strategy = new TitledPane(title:"Strategy", new Label(text:"STRATEGY_GOES_HERE"));
        TitledPane unitList = new TitledPane(title:"Full Unit/Structure List", new Label(text:"UNIT_LIST_GOES_HERE"));

        strategy.setExpanded(value:false);
        unitList.setExpanded(value:false);

        InfoBorderContainer.setCenter(InfoContainer);
        InfoBorderContainer.setBottom(close);
        InfoContainer.getChildren().addAll(overView, strategy, unitList);
```

InformationButtons: Fac1Info, Fac2Info, Fac3Info, Fac4Info: each creates a window with titled panes of information about their respective factions (unimplemented/partially implemented) as well as a close button



```
fac1Select.setOnAction(new EventHandler<ActionEvent>() {

    @Override
    public void handle(ActionEvent arg0) {
        if(!P1Selected){
            MainGameLogic.setP1Fac(facinput:1);
            StartGame.setText(value:"Select Player 2 Faction");
            P1Selected = true;
            fac1Select.setDisable(true);
        }else if(!P2Selected){
            MainGameLogic.setP2Fac(facinput:1);
            StartGame.setText(value:"START");
            StartGame.setDisable(false);

            fac1Select.setDisable(true);
            fac2Select.setDisable(true);
            fac3Select.setDisable(true);
            fac4Select.setDisable(true);

            P2Selected = true;
        }
    }

});
```

SelectionButtons: Fac1Select, Fac2Select(Disabled), Fac3Select(Disabled), Fac4Select: Each button checks the state of P1Selected, if false, it passes its respective number to MainGameLogic.setP1Fac as well as disabling itself. If true, it instead passes the number to MainGameLogic.setP2Fac, disables every selection button, and enables the start button



```
StartGame.setOnAction(new EventHandler<ActionEvent>() {

    @Override
    public void handle(ActionEvent arg0) {
        MainGameLogic.StartGame();
        ((Stage)(((Button)arg0.getSource()).getScene().getWindow())).setScene(GameBoard.GameBoardConst());
    }

});
```

StartButton: StartGame: Changes the current scene to the one specified by GameBoard.GameBoardConst()

```
      }
   }
```

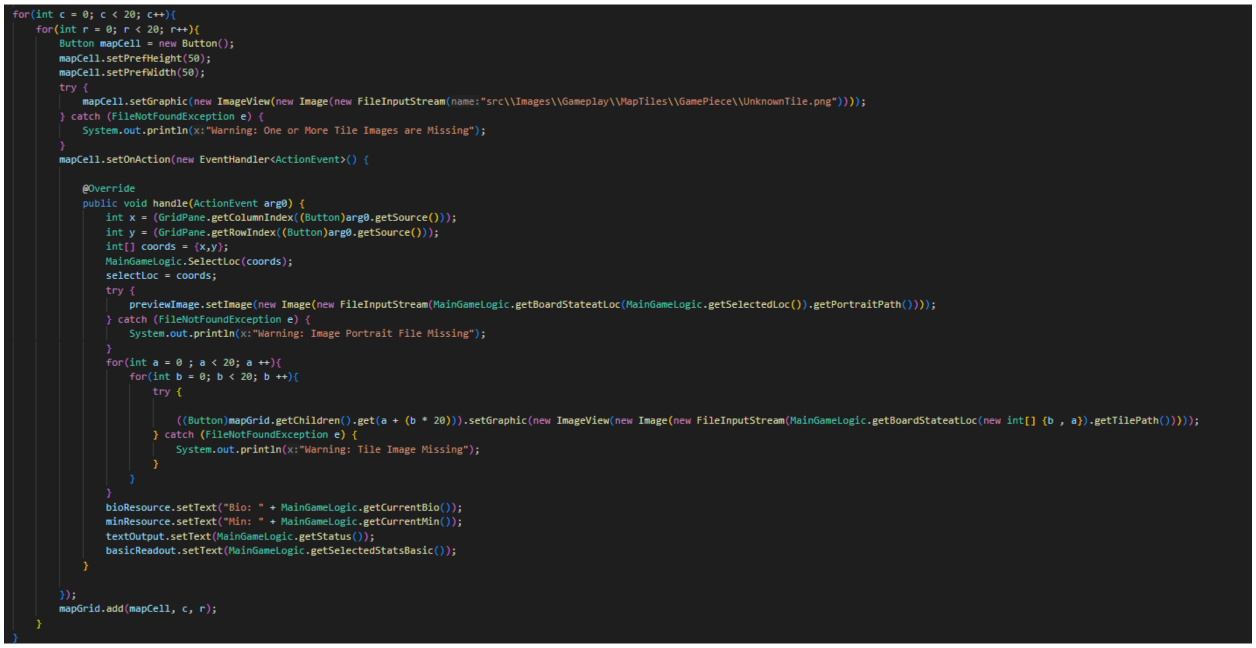
GameBoard.Java{

static int[] selectLoc: used for passing values to MainGameLogic.java

public static Scene GameBoardConst{

Contains various nested panes for formatting, several labels and images that change based on the state of gameplay, as well as two categories of buttons, mapGrid and menuButtons

Buttons:

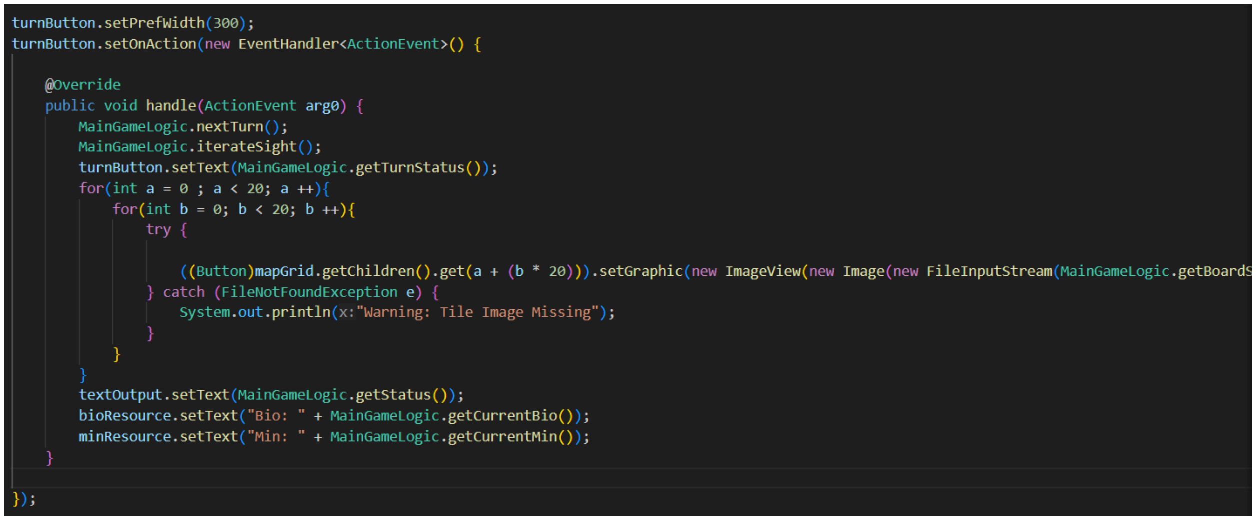


```java
for(int c = 0; c < 20; c++){
    for(int r = 0; r < 20; r++){
        Button mapCell = new Button();
        mapCell.setPrefHeight(50);
        mapCell.setPrefWidth(50);
        try {
            mapCell.setGraphic(new ImageView(new Image(new FileInputStream(name:"src\\Images\\Gameplay\\MapTiles\\GamePiece\\UnknownTile.png"))));
        } catch (FileNotFoundException e) {
            System.out.println(x:"Warning: One or More Tile Images are Missing");
        }
        mapCell.setOnAction(new EventHandler<ActionEvent>() {

            @Override
            public void handle(ActionEvent arg0) {
                int x = (GridPane.getColumnIndex((Button)arg0.getSource()));
                int y = (GridPane.getRowIndex((Button)arg0.getSource()));
                int[] coords = {x,y};
                MainGameLogic.SelectLoc(coords);
                selectLoc = coords;
                try {
                    previewImage.setImage(new Image(new FileInputStream(MainGameLogic.getBoardStateatLoc(MainGameLogic.getSelectedLoc()).getPortraitPath())));
                } catch (FileNotFoundException e) {
                    System.out.println(x:"Warning: Image Portrait File Missing");
                }
                for(int a = 0 ; a < 20; a ++){
                    for(int b = 0; b < 20; b ++){
                        try {

                            ((Button)mapGrid.getChildren().get(a + (b * 20))).setGraphic(new ImageView(new Image(new FileInputStream(MainGameLogic.getBoardStateatLoc(new int[] {b , a}).getTilePath()))));
                        } catch (FileNotFoundException e) {
                            System.out.println(x:"Warning: Tile Image Missing");
                        }
                    }
                }
                bioResource.setText("Bio: " + MainGameLogic.getCurrentBio());
                minResource.setText("Min: " + MainGameLogic.getCurrentMin());
                textOutput.setText(MainGameLogic.getStatus());
                basicReadout.setText(MainGameLogic.getSelectedStatsBasic());
            }

        });
        mapGrid.add(mapCell, c, r);
    }
}
```

MapGrid: a 20x20 grid of square buttons contained in a gridpane, on click each of them sends their position in the gridpane to MainGameLogic.selectLoc(), then images and texts are updated with the MainGameLogic functions .getBoardStateAtLoc(), .getCurrentBio(), .getCurrentMin(), .getSelectedStatsBasic() and .getStatus()

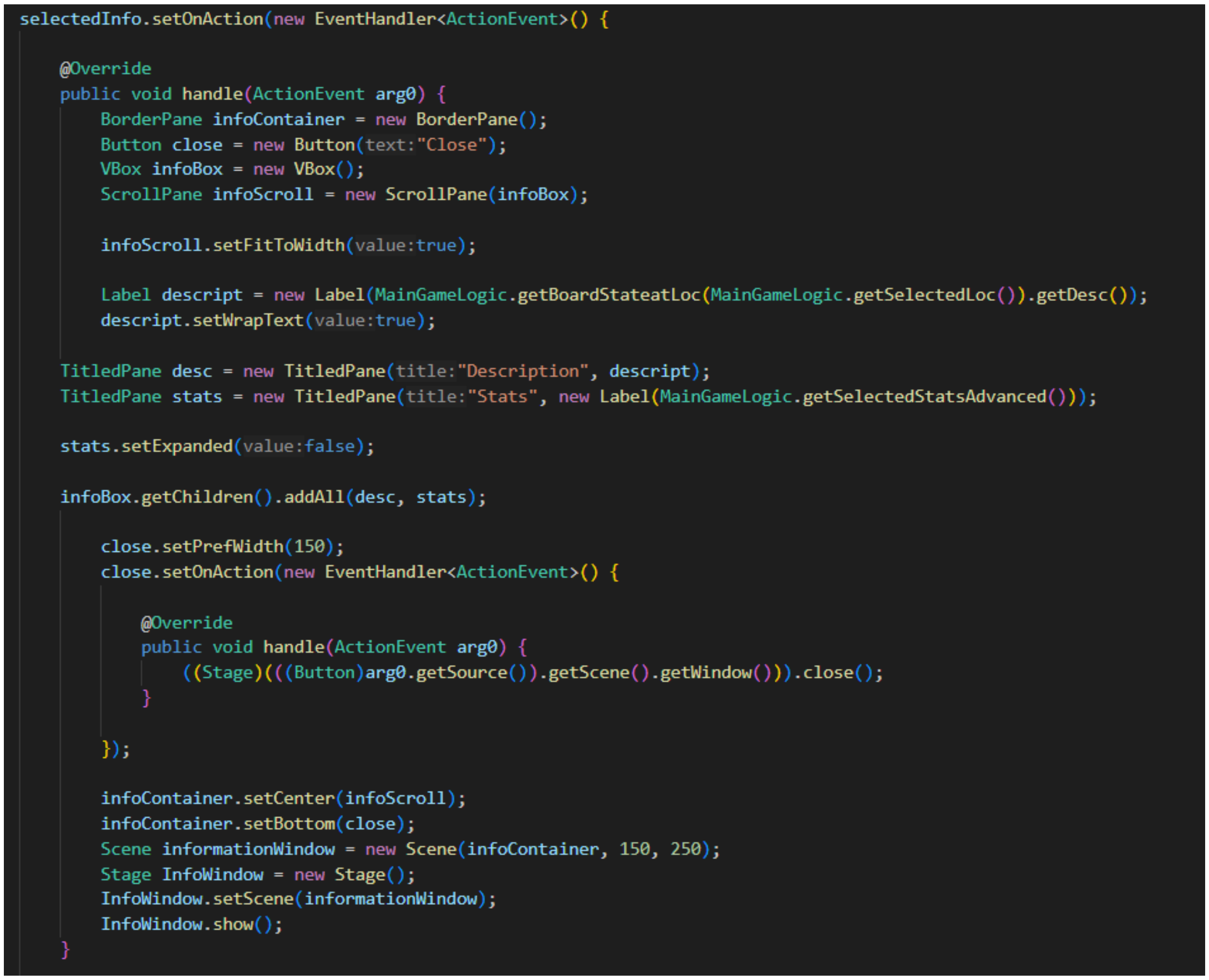
Additionally uses the functions .getPortraitPath() and .getTilePath() from the Entity class

MenuButtons:



```java
turnButton.setPrefWidth(300);
turnButton.setOnAction(new EventHandler<ActionEvent>() {

    @Override
    public void handle(ActionEvent arg0) {
        MainGameLogic.nextTurn();
        MainGameLogic.iterateSight();
        turnButton.setText(MainGameLogic.getTurnStatus());
        for(int a = 0 ; a < 20; a ++){
            for(int b = 0; b < 20; b ++){
                try {

                    ((Button)mapGrid.getChildren().get(a + (b * 20))).setGraphic(new ImageView(new Image(new FileInputStream(MainGameLogic.getBoardS
                } catch (FileNotFoundException e) {
                    System.out.println(x:"Warning: Tile Image Missing");
                }
            }
        }
        textOutput.setText(MainGameLogic.getStatus());
        bioResource.setText("Bio: " + MainGameLogic.getCurrentBio());
        minResource.setText("Min: " + MainGameLogic.getCurrentMin());
    }

});
```

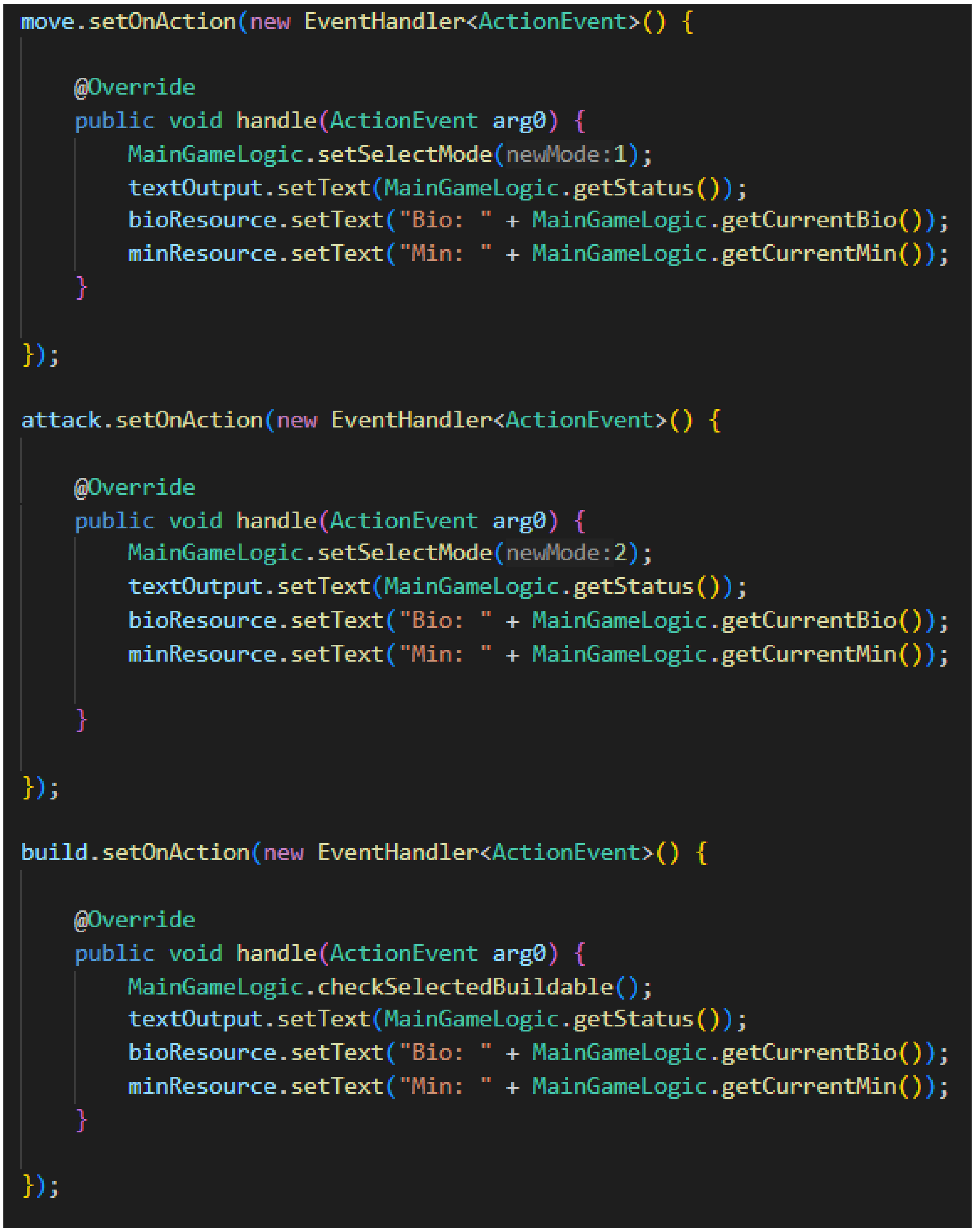
TurnButton: Calls MainGameLogic.nextTurn(), as well as update information similar to the mapGrid buttons



```java
selectedInfo.setOnAction(new EventHandler<ActionEvent>() {

    @Override
    public void handle(ActionEvent arg0) {
        BorderPane infoContainer = new BorderPane();
        Button close = new Button(text:"Close");
        VBox infoBox = new VBox();
        ScrollPane infoScroll = new ScrollPane(infoBox);

        infoScroll.setFitToWidth(value:true);

        Label descript = new Label(MainGameLogic.getBoardStateatLoc(MainGameLogic.getSelectedLoc()).getDesc());
        descript.setWrapText(value:true);

    TitledPane desc = new TitledPane(title:"Description", descript);
    TitledPane stats = new TitledPane(title:"Stats", new Label(MainGameLogic.getSelectedStatsAdvanced()));

    stats.setExpanded(value:false);

    infoBox.getChildren().addAll(desc, stats);

        close.setPrefWidth(150);
        close.setOnAction(new EventHandler<ActionEvent>() {

            @Override
            public void handle(ActionEvent arg0) {
                ((Stage)(((Button)arg0.getSource()).getScene().getWindow())).close();
            }

        });

        infoContainer.setCenter(infoScroll);
        infoContainer.setBottom(close);
        Scene informationWindow = new Scene(infoContainer, 150, 250);
        Stage InfoWindow = new Stage();
        InfoWindow.setScene(informationWindow);
        InfoWindow.show();
    }
```

SelectedInfo: Opens a new window of titled panes with labels, along with a close button. Labels contain output of

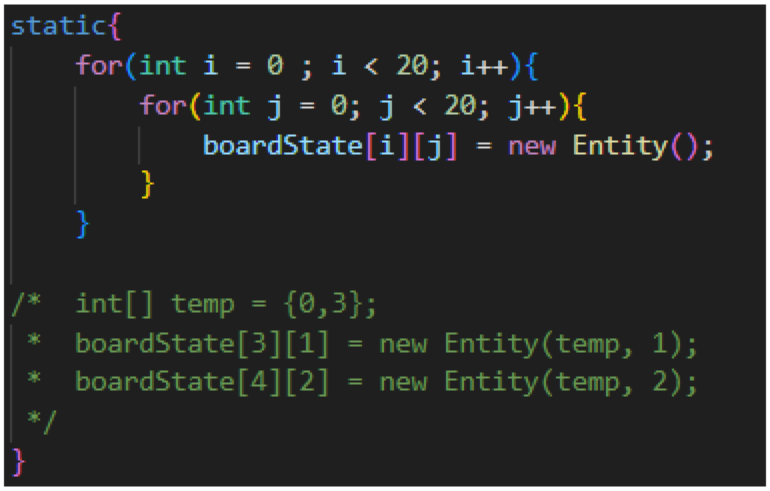
MainGameLogic.getBoardStateatLoc(MainGameLogic.getSelectedLoc()).getDesc() and MainGameLogic.getSelectedStatsAdvanced()



```
move.setOnAction(new EventHandler<ActionEvent>() {

    @Override
    public void handle(ActionEvent arg0) {
        MainGameLogic.setSelectMode(newMode:1);
        textOutput.setText(MainGameLogic.getStatus());
        bioResource.setText("Bio: " + MainGameLogic.getCurrentBio());
        minResource.setText("Min: " + MainGameLogic.getCurrentMin());
    }

});

attack.setOnAction(new EventHandler<ActionEvent>() {

    @Override
    public void handle(ActionEvent arg0) {
        MainGameLogic.setSelectMode(newMode:2);
        textOutput.setText(MainGameLogic.getStatus());
        bioResource.setText("Bio: " + MainGameLogic.getCurrentBio());
        minResource.setText("Min: " + MainGameLogic.getCurrentMin());

    }

});

build.setOnAction(new EventHandler<ActionEvent>() {

    @Override
    public void handle(ActionEvent arg0) {
        MainGameLogic.checkSelectedBuildable();
        textOutput.setText(MainGameLogic.getStatus());
        bioResource.setText("Bio: " + MainGameLogic.getCurrentBio());
        minResource.setText("Min: " + MainGameLogic.getCurrentMin());
    }

});
```

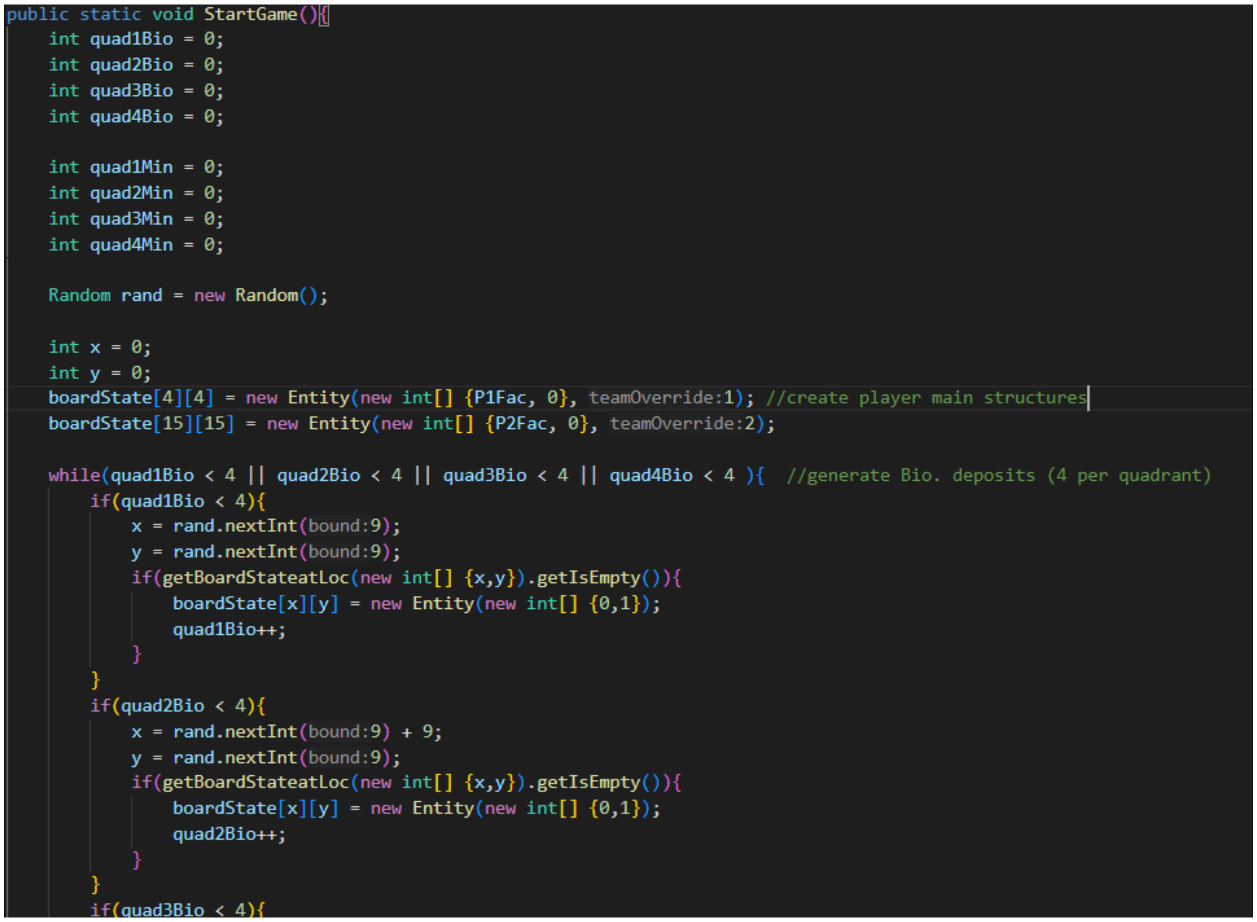
Move: sets MainGameLogic.selectedMode to 1 through MainGameLogic.setSelectedMode(int); additionally updates text outputs

```
            Attack: sets MainGameLogic.selectedMode to 1 through MainGameLogic.setSelectedMode(int); additionally updates text outputs

            Build: sets MainGameLogic.selectedMode to 1 through MainGameLogic.setSelectedMode(int); additionally updates text outputs

        }

    }
]
GameMechanics[
    MainGameLogic.java{
        static int P1Fac : stores numerical value of player 1's selected faction
        static int P2Fac : stores numerical value of player 2's selected faction
        static int turn : stores the current turn
        static int nextTurn : stores the next non-zero turn value
        static int p1Bio : stores player 1's current biomass resource
        static int p1BioInc : unused, intended to contain p1 bio income per turn
        static int p1Min : stores player 1's current mineral resource
        static int p1MinInc : unused, intended to contain p1 min income per turn
        static int p2Bio : stores player 2's current biomass resource
        static int p2BioInc : unused, intended to contain p2 bio income per turn
        static int p2Min : stores player 2's current mineral resource
        static int p2MinInc : unused, intended to contain p2 min income per turn
        static int X1 : Appears unused
        static int Y1 : Appears unused
        static int X2 : Appears unused
        static int Y2 : Appears unused
        static int selectMode : stores the current mode of selection
        static int[] selectedLoc : stores the current selected location
        static int[] savedLoc : stores the previously selected location for the purposes of various select modes
        static int[] waitingUnitID : stores the ID of an entity before it is created
        static int constructionSpecialMode : stores whether the current entity being placed has special conditions to its location

        Entity[][] boardState = new Entity[20][20] : 2d array to represent the game board through code

        static block{
```


```
static{
    for(int i = 0 ; i < 20; i++){
        for(int j = 0; j < 20; j++){
            boardState[i][j] = new Entity();
        }
    }

/*  int[] temp = {0,3};
 *  boardState[3][1] = new Entity(temp, 1);
 *  boardState[4][2] = new Entity(temp, 2);
 */
}
```
iterates through the boardstate array, setting default values to each tile

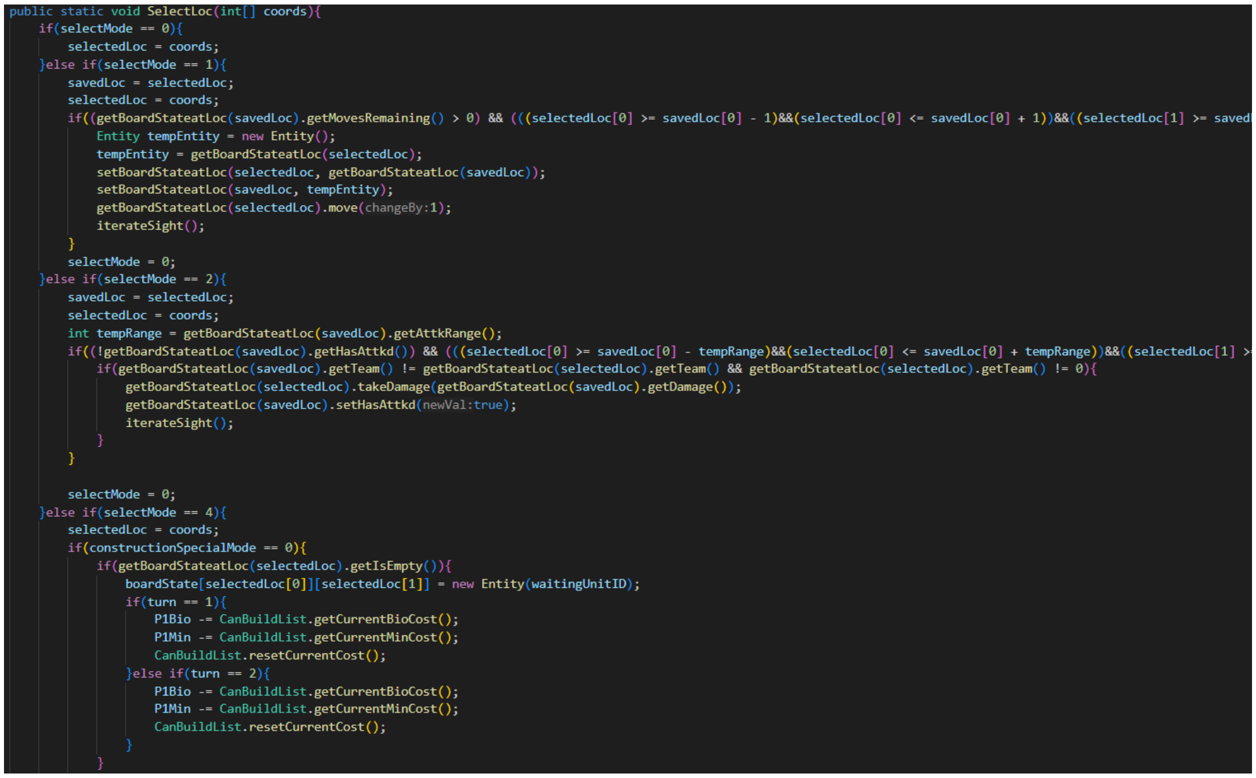
}

public static void startGame(){


```
public static void StartGame(){
    int quad1Bio = 0;
    int quad2Bio = 0;
    int quad3Bio = 0;
    int quad4Bio = 0;

    int quad1Min = 0;
    int quad2Min = 0;
    int quad3Min = 0;
    int quad4Min = 0;

    Random rand = new Random();

    int x = 0;
    int y = 0;
    boardState[4][4] = new Entity(new int[] {P1Fac, 0}, teamOverride:1); //create player main structures
    boardState[15][15] = new Entity(new int[] {P2Fac, 0}, teamOverride:2);

    while(quad1Bio < 4 || quad2Bio < 4 || quad3Bio < 4 || quad4Bio < 4 ){  //generate Bio. deposits (4 per quadrant)
        if(quad1Bio < 4){
            x = rand.nextInt(bound:9);
            y = rand.nextInt(bound:9);
            if(getBoardStateatLoc(new int[] {x,y}).getIsEmpty()){
                boardState[x][y] = new Entity(new int[] {0,1});
                quad1Bio++;
            }
        }
        if(quad2Bio < 4){
            x = rand.nextInt(bound:9) + 9;
            y = rand.nextInt(bound:9);
            if(getBoardStateatLoc(new int[] {x,y}).getIsEmpty()){
                boardState[x][y] = new Entity(new int[] {0,1});
                quad2Bio++;
            }
        }
        if(quad3Bio < 4){
```
adds each player's starting location to the boardState Entity array, then scatters neutral min and bio deposits throughout it

}

```
public static int getCurrentTurn(){
    returns the current turn
}
public static void setP1Fac(int){
    Sets the selected faction of player 1
}
public static void setP2Fac(int){
    Same, but for player 2
}
public static int getP1Fac(){
    returns player 1's numerical faction value
}
public static int getP2Fac(){
    same as above, but for player 2 instead
}

public static void selectLoc(int[]){
```



```java
public static void SelectLoc(int[] coords){
    if(selectMode == 0){
        selectedLoc = coords;
    }else if(selectMode == 1){
        savedLoc = selectedLoc;
        selectedLoc = coords;
        if((getBoardStateatLoc(savedLoc).getMovesRemaining() > 0) && (((selectedLoc[0] >= savedLoc[0] - 1)&&(selectedLoc[0] <= savedLoc[0] + 1))&&((selectedLoc[1] >= saved
            Entity tempEntity = new Entity();
            tempEntity = getBoardStateatLoc(selectedLoc);
            setBoardStateatLoc(selectedLoc, getBoardStateatLoc(savedLoc));
            setBoardStateatLoc(savedLoc, tempEntity);
            getBoardStateatLoc(selectedLoc).move(changeBy:1);
            iterateSight();
        }
        selectMode = 0;
    }else if(selectMode == 2){
        savedLoc = selectedLoc;
        selectedLoc = coords;
        int tempRange = getBoardStateatLoc(savedLoc).getAttkRange();
        if((!getBoardStateatLoc(savedLoc).getHasAttkd()) && (((selectedLoc[0] >= savedLoc[0] - tempRange)&&(selectedLoc[0] <= savedLoc[0] + tempRange))&&((selectedLoc[1] >
            if(getBoardStateatLoc(savedLoc).getTeam() != getBoardStateatLoc(selectedLoc).getTeam() && getBoardStateatLoc(selectedLoc).getTeam() != 0){
                getBoardStateatLoc(selectedLoc).takeDamage(getBoardStateatLoc(savedLoc).getDamage());
                getBoardStateatLoc(savedLoc).setHasAttkd(newVal:true);
                iterateSight();
            }
        }

        selectMode = 0;
    }else if(selectMode == 4){
        selectedLoc = coords;
        if(constructionSpecialMode == 0){
            if(getBoardStateatLoc(selectedLoc).getIsEmpty()){
                boardState[selectedLoc[0]][selectedLoc[1]] = new Entity(waitingUnitID);
                if(turn == 1){
                    P1Bio -= CanBuildList.getCurrentBioCost();
                    P1Min -= CanBuildList.getCurrentMinCost();
                    CanBuildList.resetCurrentCost();
                }else if(turn == 2){
                    P1Bio -= CanBuildList.getCurrentBioCost();
                    P1Min -= CanBuildList.getCurrentMinCost();
                    CanBuildList.resetCurrentCost();
                }
            }
```

changes the selected location to the inputted integer array, then depending on the select mode, swaps values in the boardState[][] array, runs Entity.takeDamage(int) on specified index, or creates a new entity at the location specified

```
}
public static int[] getSelectedLoc(){
    returns the selectedLoc array
}
```

public static Entity getBoardStateAtLoc(int[] coords){

returns the value of the boardState array at the specified location, as specified with Boardstate[coords[0]][coords[1]]

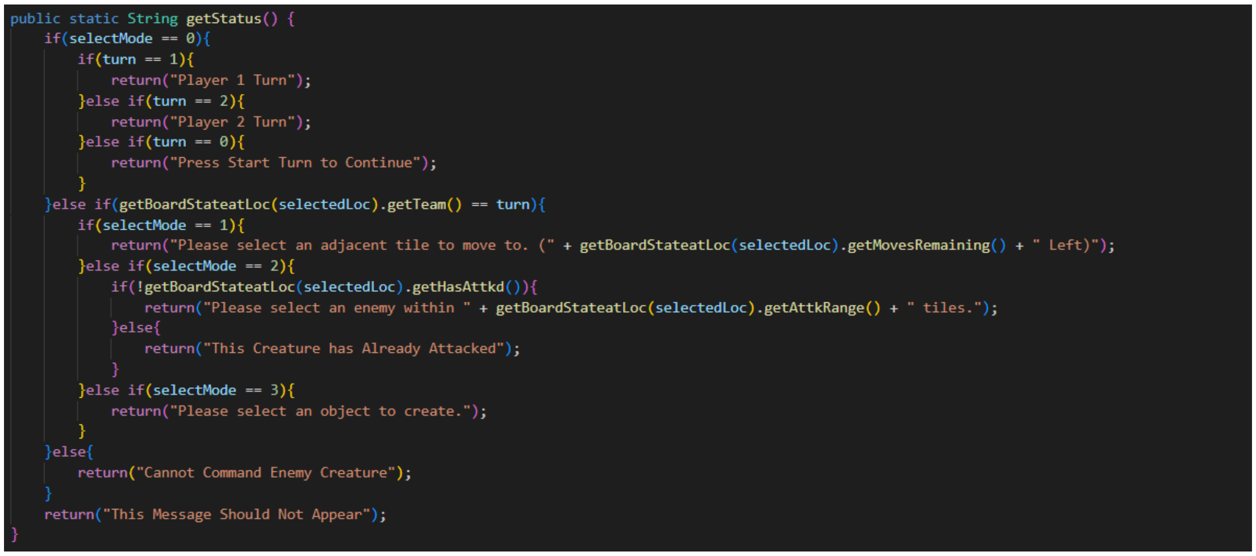
}

public static void setBoardStateAtLoc(int[],Entity){

selects a location in the same process as above, then sets that location to the Entity parameter

}

public static String getStatus(){



```java
public static String getStatus() {
    if(selectMode == 0){
        if(turn == 1){
            return("Player 1 Turn");
        }else if(turn == 2){
            return("Player 2 Turn");
        }else if(turn == 0){
            return("Press Start Turn to Continue");
        }
    }else if(getBoardStateatLoc(selectedLoc).getTeam() == turn){
        if(selectMode == 1){
            return("Please select an adjacent tile to move to. (" + getBoardStateatLoc(selectedLoc).getMovesRemaining() + " Left)");
        }else if(selectMode == 2){
            if(!getBoardStateatLoc(selectedLoc).getHasAttkd()){
                return("Please select an enemy within " + getBoardStateatLoc(selectedLoc).getAttkRange() + " tiles.");
            }else{
                return("This Creature has Already Attacked");
            }
        }else if(selectMode == 3){
            return("Please select an object to create.");
        }
    }else{
        return("Cannot Command Enemy Creature");
    }
    return("This Message Should Not Appear");
}
```

returns a variety of strings depending on the current turn, player actions, and the selectedMode value

}

public static int setSelectedMode(int){

sets the selectedMode to the specified integer

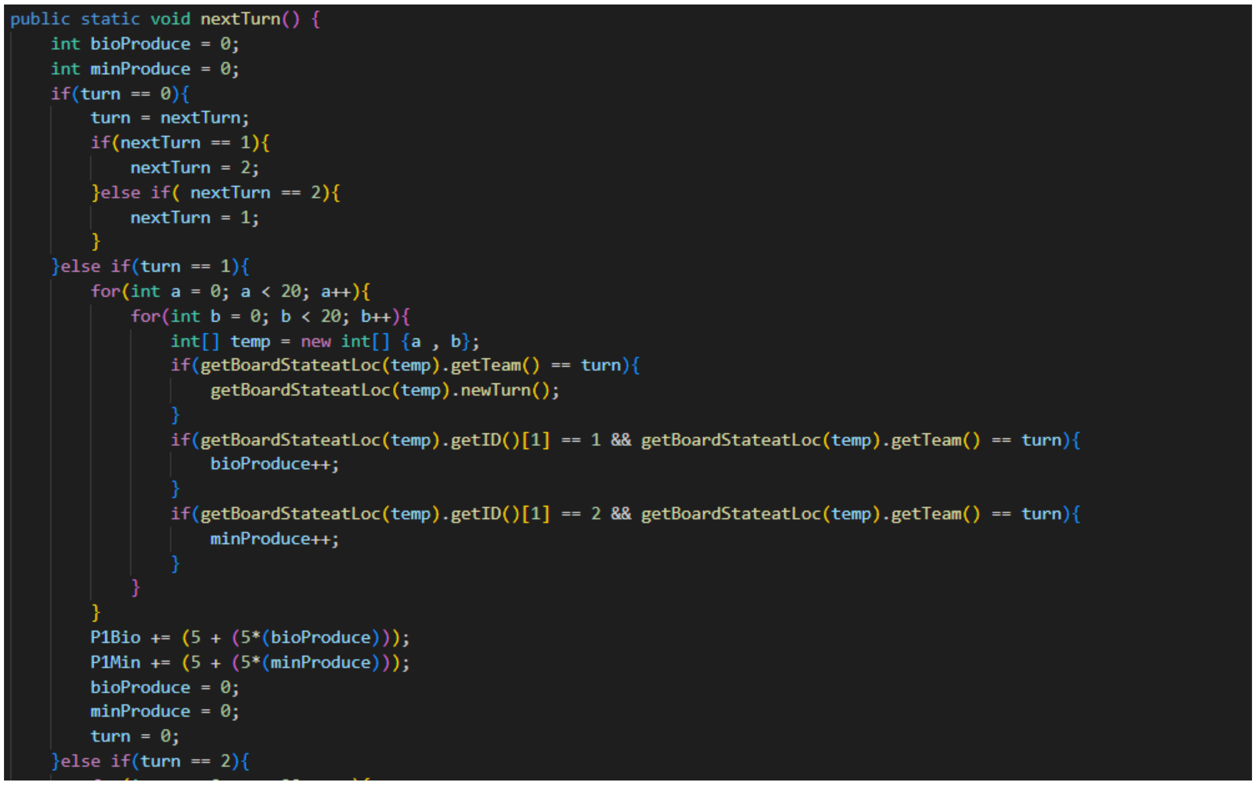
}

public static void iterateSight(){

iterates through the boardState[][] array, updating the visible boolean of each of the contained entities based on the current turn

}

public static void nextTurn(){



```
public static void nextTurn() {
    int bioProduce = 0;
    int minProduce = 0;
    if(turn == 0){
        turn = nextTurn;
        if(nextTurn == 1){
            nextTurn = 2;
        }else if( nextTurn == 2){
            nextTurn = 1;
        }
    }else if(turn == 1){
        for(int a = 0; a < 20; a++){
            for(int b = 0; b < 20; b++){
                int[] temp = new int[] {a , b};
                if(getBoardStateatLoc(temp).getTeam() == turn){
                    getBoardStateatLoc(temp).newTurn();
                }
                if(getBoardStateatLoc(temp).getID()[1] == 1 && getBoardStateatLoc(temp).getTeam() == turn){
                    bioProduce++;
                }
                if(getBoardStateatLoc(temp).getID()[1] == 2 && getBoardStateatLoc(temp).getTeam() == turn){
                    minProduce++;
                }
            }
        }
        P1Bio += (5 + (5*(bioProduce)));
        P1Min += (5 + (5*(minProduce)));
        bioProduce = 0;
        minProduce = 0;
        turn = 0;
    }else if(turn == 2){
```

if current turn is 0, start the next player's turn

if current turn is not 0, change the nextTurn value to the opposite of the current turn, iterate through the player's entities to reset their movesMade and hasAttkd values, and adds to the current player's Bio and Min resources, then sets the turn to 0

}

public static int getCurrentBio(){

if turn = 0, return 0

otherwise, return the current player's bio resource

}

public static int getCurrentMin(){

Same as above, but for Min instead

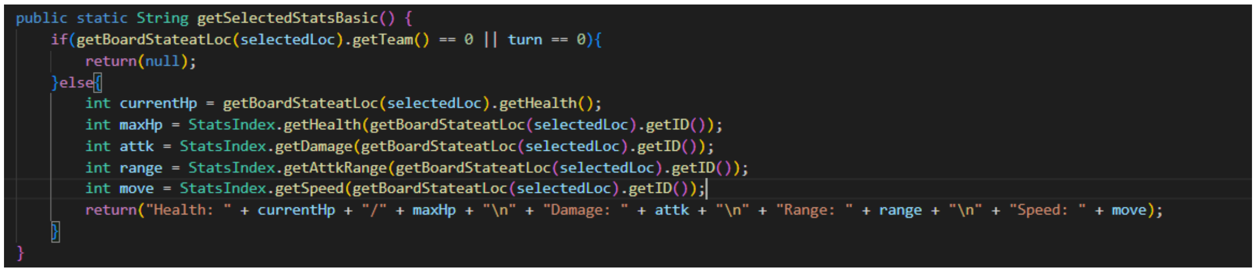
}

public static string getTurnStatus(){

returns either "Start turn" or "End turn" if the turn value is or is not 0 respectively

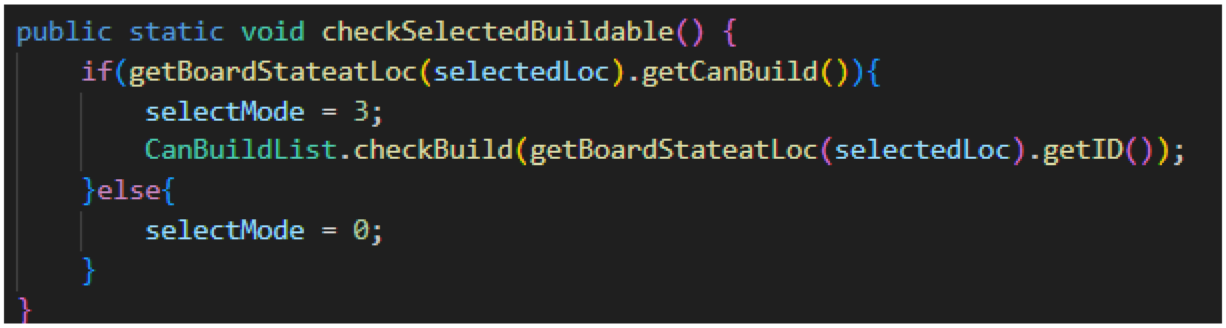
}

public static string getSelectedStatsBasic(){



```
public static String getSelectedStatsBasic() {
    if(getBoardStateatLoc(selectedLoc).getTeam() == 0 || turn == 0){
        return(null);
    }else{
        int currentHp = getBoardStateatLoc(selectedLoc).getHealth();
        int maxHp = StatsIndex.getHealth(getBoardStateatLoc(selectedLoc).getID());
        int attk = StatsIndex.getDamage(getBoardStateatLoc(selectedLoc).getID());
        int range = StatsIndex.getAttkRange(getBoardStateatLoc(selectedLoc).getID());
        int move = StatsIndex.getSpeed(getBoardStateatLoc(selectedLoc).getID());
        return("Health: " + currentHp + "/" + maxHp + "\n" + "Damage: " + attk + "\n" + "Range: " + range + "\n" + "Speed: " + move);
    }
}
```

    if the selected tile does not belong to team 0, return a few of the units statistics in the form of a formatted string, otherwise return null
  }
  public static string getSelectedStatsAdvanced(){
    essentially the same as above, but returns more information, does not depend on team number
  }
  public static void checkSelectedBuildable(){

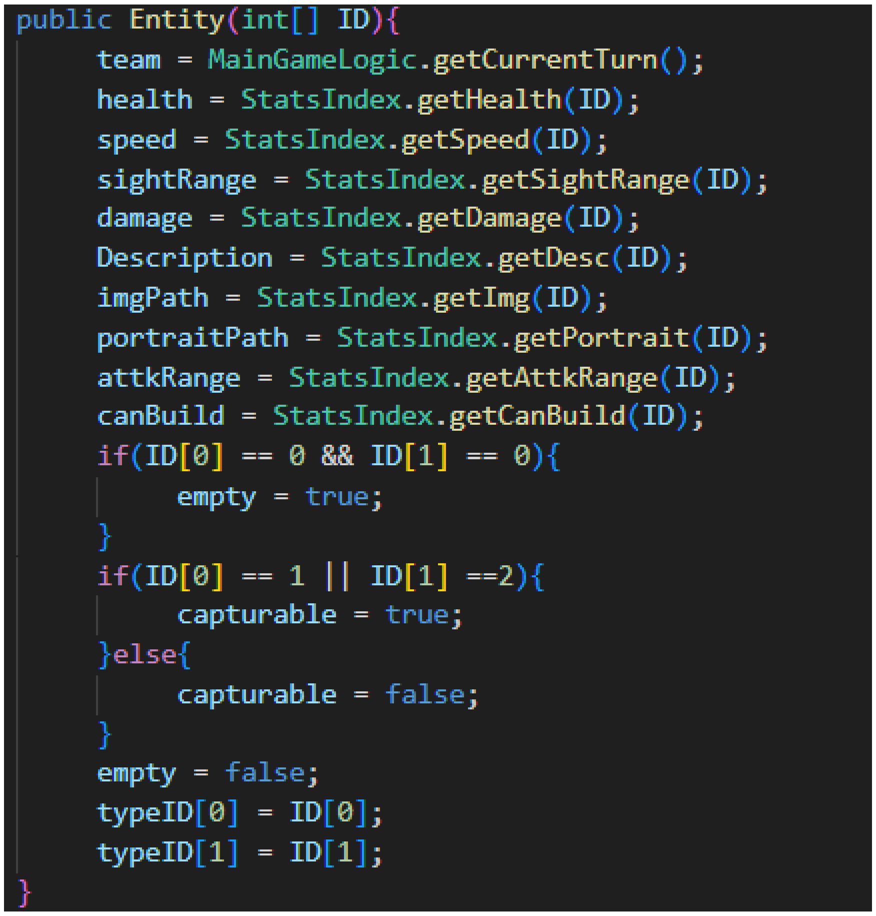


```java
public static void checkSelectedBuildable() {
    if(getBoardStateatLoc(selectedLoc).getCanBuild()){
        selectMode = 3;
        CanBuildList.checkBuild(getBoardStateatLoc(selectedLoc).getID());
    }else{
        selectMode = 0;
    }
}
```

    checks the Entity at the selected location, either continues with the build function, or cancels it depending on its canBuild boolean
  }
  public static void buildReturnValue(boolean){
    Sets select mode to 4 is boolean is true, 0 otherwise
  }
  public static void UnitCreate(int[]){
    sets waitingUnitID to the specified integer array
  }
  public static void setConstructionSpecialMode(int){
    sets the constructionSpecialMode value to the specified integer
  }
}
Entity.java{
  private int team : Stores the numerical value of the entitys' controller
  private int health : Stores the current hp of the entity
  private int speed : Stores stores the move speed of the entity
  private int movesMade : Stores how many moves the entity has made this turn
  private int sightRange : Stores the sight range of the entity
  private int damage : Stores the damage value of the entity
  private int attkRange : Stores the attack range of the entity
  private String imgPath : Stores the path to the game tile representation of the entity
  private String portraitPath : Stores the path to the portrait representation of the entity
  private String Description : Stores the description of the entity
  private boolean visible : Stores whether the entity can be seen by the current active player
  private boolean empty : Stores whether the entity is considered empty space for the purposes of movement and creation
  private boolean hasAttkd : Stores whether the entity has attacked this turn

private boolean canBuild : Stores whether the entity can be the source of another
private boolean capturable : Stores whether the entity is considered capturable (essentially if it is considered a resource node)
private int[] typeID : Stores the two integer ID of the entity, such as for use with any of the StatsIndex.java functions

contains 3 constructors:
public Entity(int[]){

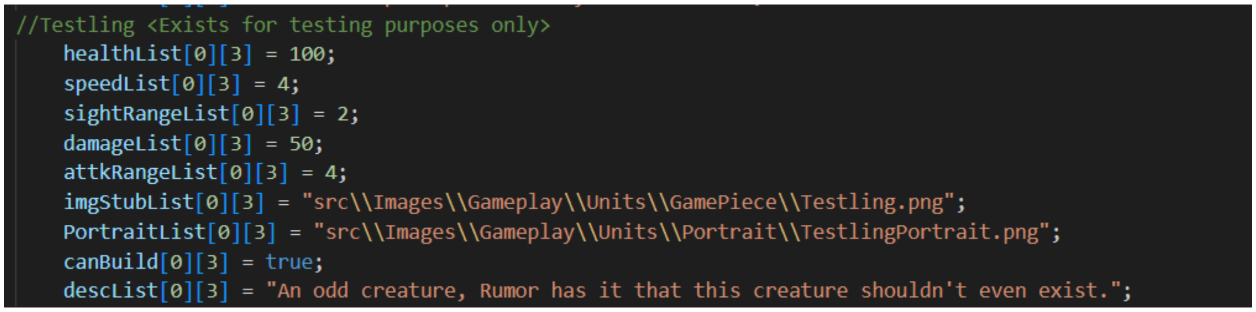


```java
public Entity(int[] ID){
    team = MainGameLogic.getCurrentTurn();
    health = StatsIndex.getHealth(ID);
    speed = StatsIndex.getSpeed(ID);
    sightRange = StatsIndex.getSightRange(ID);
    damage = StatsIndex.getDamage(ID);
    Description = StatsIndex.getDesc(ID);
    imgPath = StatsIndex.getImg(ID);
    portraitPath = StatsIndex.getPortrait(ID);
    attkRange = StatsIndex.getAttkRange(ID);
    canBuild = StatsIndex.getCanBuild(ID);
    if(ID[0] == 0 && ID[1] == 0){
        empty = true;
    }
    if(ID[0] == 1 || ID[1] ==2){
        capturable = true;
    }else{
        capturable = false;
    }
    empty = false;
    typeID[0] = ID[0];
    typeID[1] = ID[1];
}
```

sets each of the fields above according to the inputted int[], using StatsIndex.get<NAME>(int[]) functions
sets team to the current active team using MainGameLogic.getCurrentTurn()
}
public Entity(int[],int){
same as above, but the second int is used to override the team number
}
public Entity(){
Creates an entity with default values for each of the fields, used for initializing a two dimensional array of the Entity type
}

Various Getters and Setters for the above fields

public void takeDamage(int){
subtracts the inputted integer from the Entity's health value, then if <= 0 , reverts the entity to either an empty tile, or a neutral resource deposit, depending on whether capturable is false or true respectively
}
}
StatsIndex.java{

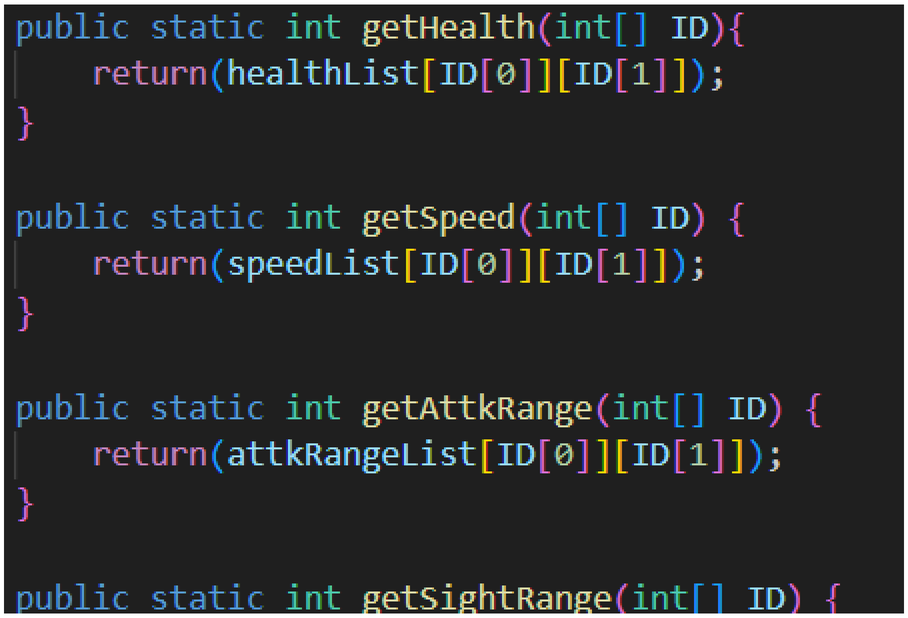


```
//Testling <Exists for testing purposes only>
    healthList[0][3] = 100;
    speedList[0][3] = 4;
    sightRangeList[0][3] = 2;
    damageList[0][3] = 50;
    attkRangeList[0][3] = 4;
    imgStubList[0][3] = "src\\Images\\Gameplay\\Units\\GamePiece\\Testling.png";
    PortraitList[0][3] = "src\\Images\\Gameplay\\Units\\Portrait\\TestlingPortrait.png";
    canBuild[0][3] = true;
    descList[0][3] = "An odd creature, Rumor has it that this creature shouldn't even exist.";
```

Contains a variety of two dimensional arrays: int[][] healthList, int[][] speedList, int[][] sightRangeList, int[][] damageList, int[][] attkRange, String[][] imgStubList, String[][] portraitList, String[][] descList, and Boolean[][] canBuild.

each has a getter method with the format:

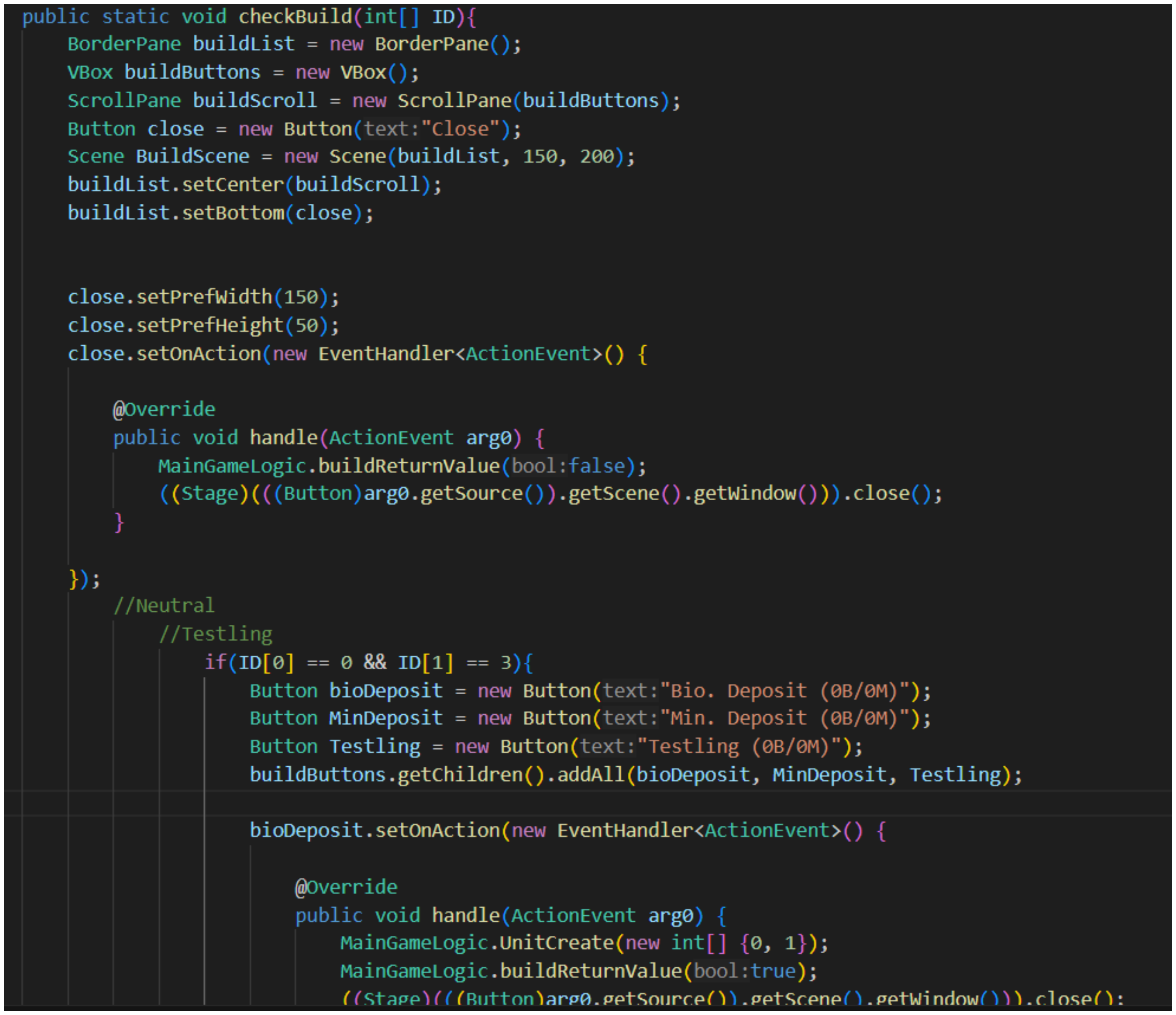
```
get<NAME>(int[] ID){
    return(<NAMELIST>[ID[0]][ID[1]]);
}
```



```
public static int getHealth(int[] ID){
    return(healthList[ID[0]][ID[1]]);
}

public static int getSpeed(int[] ID) {
    return(speedList[ID[0]][ID[1]]);
}

public static int getAttkRange(int[] ID) {
    return(attkRangeList[ID[0]][ID[1]]);
}

public static int getSightRange(int[] ID) {
```

}
CanBuildList.java{
public static int currentBioCost
public static int currentMinCost

public static void checkBuild(int[] ID){



```java
public static void checkBuild(int[] ID){
    BorderPane buildList = new BorderPane();
    VBox buildButtons = new VBox();
    ScrollPane buildScroll = new ScrollPane(buildButtons);
    Button close = new Button(text:"Close");
    Scene BuildScene = new Scene(buildList, 150, 200);
    buildList.setCenter(buildScroll);
    buildList.setBottom(close);


    close.setPrefWidth(150);
    close.setPrefHeight(50);
    close.setOnAction(new EventHandler<ActionEvent>() {

        @Override
        public void handle(ActionEvent arg0) {
            MainGameLogic.buildReturnValue(bool:false);
            ((Stage)(((Button)arg0.getSource()).getScene().getWindow())).close();
        }

    });
        //Neutral
            //Testling
                if(ID[0] == 0 && ID[1] == 3){
                    Button bioDeposit = new Button(text:"Bio. Deposit (0B/0M)");
                    Button MinDeposit = new Button(text:"Min. Deposit (0B/0M)");
                    Button Testling = new Button(text:"Testling (0B/0M)");
                    buildButtons.getChildren().addAll(bioDeposit, MinDeposit, Testling);

                    bioDeposit.setOnAction(new EventHandler<ActionEvent>() {

                        @Override
                        public void handle(ActionEvent arg0) {
                            MainGameLogic.UnitCreate(new int[] {0, 1});
                            MainGameLogic.buildReturnValue(bool:true);
                            ((Stage)(((Button)arg0.getSource()).getScene().getWindow())).close();
```

Creates a button list of options in a new window based on the inputted ID array if the current player's Bio or Min values are lower than a specified cost, then the button is Disabled

when clicked, each button sets currentBioCost and currentMinCost to the specified cost sends an integer array to MainGameLogic.UnitCreate(int[]) based on the selected option, passes a value to MainGameLogic.constructionSpecialMode(int) if necessary, calls MainGameLogic.setBuildReturnValue(true), and closes the popup list

Additionally, an exit button that calls MainGameLogic.setBuildReturnValue(false) and closes the window

the window also calls that function if it is closed through outside means, such as the Windows close button

}

public static int getCurrentBioCost{

returns the currentBioCost integer value

}

```
        public static int getCurrentMinCost{
            returns the currentMinCost integer value
        }
        public static void resetCurrentCost(){
            sets currentBioCost and currentMinCost to 0
        }
    }
]
```