Controller

Class Controller

java.lang.Object
    Controller.Controller

```
public class Controller
extends java.lang.Object
```

Version:

1.5

Author:

Youssef Saghbini, Ryan Boucher

Field Summary

Fields

Modifier and Type	Field and Description
private Square[][]	board Dual-array gameboard to be played on.
private Coordinate	clickedButtonLocation
private java.util.List<java.lang.String>	loggingList
private int	moneyPouch User's money pouch during the game.
private View	view
private int	zombieLimit The amount of zombies to be spawned within the board.

Constructor Summary

Constructors

Constructor and Description
Controller(View view) Will generate a brand new board with initial values.

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
void	`actionListener()` Main code for the "Controller" aspect of the MVC model that is required for this milestone.
boolean	`add(Coordinate coordinate, Piece piece)` Adding pieces around the generated gameBoard.
void	`addingZombie()` Adding zombies randomly at the end of the board.
void	`gameOver()` Will end the game, if any zombies have reached at the end of the gameboard.
void	`gameWon()` Once all the zombies have been spawned, it will go through all the squares in the board; To see if any zombies are "alive".
void	`getLogging()`
private Square	`getSquare(Coordinate c)` Receiving the square at specific coordinate, as the square contains both the coordinate and piece.
void	`hitUpdate()` When piece is within range of attack, it will affect the other piece's health.
boolean	`move(Coordinate src, Coordinate dest)` This method is to move a piece from one coordinate to another.
void	`movingZombie()` Used for the zombies to move one square forward after every round.
boolean	`purchasePiece(Piece piece)` To see if the user is able to purchase a new piece.
void	`removeUpdate()` It will remove pieces when health is equal to zero and below.
void	`reset()` Will re-initialize the gameboard, where no piece has spawned.
void	`runTime()` This method is used to call the other methods required to finish a turn, after the player has placed his/her plants.

void	sunflowerMoney() Whenever there is a sunflower spawned in the game, it will add money into the user's money pouch.
java.lang.String	toString()

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

board

```
private Square[][] board
```

Dual-array gameboard to be played on.

moneyPouch

```
private int moneyPouch
```

User's money pouch during the game.

zombieLimit

```
private int zombieLimit
```

The amount of zombies to be spawned within the board.

view

```
private View view
```

clickedButtonLocation

```
private Coordinate clickedButtonLocation
```

loggingList

```
private java.util.List<java.lang.String> loggingList
```

Constructor Detail

Controller

```
public Controller(View view)
```

Will generate a brand new board with initial values. Board will consist of a dual array of squares, and each square would contain a specific coordinate and piece when added and or removed. Logging is to keep track of every event happening. Money pouch is the amount of money the player will have. ZombieLimit is the amount of zombies allowed to be spawned into the board.

Method Detail

actionListener

```
public void actionListener()
```

Main code for the "Controller" aspect of the MVC model that is required for this milestone. Adds action listeners to all buttons on the game board, and handles user-input on the pop-up menus that allow for the placing of plants in the game, through the use of action events on the popups. After plant is placed, runtime() is called to finish the turn, perform zombie logic, and award sun-points. Currently supports the placing of peashooters and sunflowers.

runTime

```
public void runTime()
```

This method is used to call the other methods required to finish a turn, after the player has placed his/her plants.

add

```
public boolean add(Coordinate coordinate,
                   Piece piece)
```

Adding pieces around the generated gameBoard. Will use the addPiece() and removePiece() methods, when necessary.

Parameters:

`coordinate - receiving the coordinate at which the piece will be placed`

`piece - receiving the type of piece to be added at specific coordinate`

Returns:

`Whether if it is possible to add the piece within conditions`

move

```
public boolean move(Coordinate src,
                    Coordinate dest)
```

This method is to move a piece from one coordinate to another. It will receive the current and new coordinate; in those coordinates, the method will retrieve the piece and move them.

Parameters:

`src - the current coordinate in the piece is currently placed`

`dest - the potential new coordinate where the piece will be move to`

hitUpdate

```
public void hitUpdate()
```

When piece is within range of attack, it will affect the other piece's health.

addingZombie

```
public void addingZombie()
```

Adding zombies randomly at the end of the board.

movingZombie

```
public void movingZombie()
```

Used for the zombies to move one square forward after every round.

removeUpdate

```
public void removeUpdate()
```

It will remove pieces when health is equal to zero and below.

sunflowerMoney

```
public void sunflowerMoney()
```

Whenever there is a sunflower spawned in the game, it will add money into the user's money pouch.

purchasePiece

```
public boolean purchasePiece(Piece piece)
```

To see if the user is able to purchase a new piece.

Parameters:

`piece - The piece wanting to purchase.`

Returns:

`The ability to purchase a piece.`

reset

```
public void reset()
```

Will re-initialize the gameboard, where no piece has spawned.

gameWon

```
public void gameWon()
```

Once all the zombies have been spawned, it will go through all the squares in the board; To see if any zombies are "alive". If there are zombies still alive then the game keeps going. If all are killed, then the game ends.

gameOver

```
public void gameOver()
```

Will end the game, if any zombies have reached at the end of the gameboard.

getSquare

```
private Square getSquare(Coordinate c)
```

Receiving the square at specific coordinate, as the square contains both the coordinate and piece.

Parameters:

`c - Model.Coordinate of the square needed`

Returns:

`The square at specific coordinate`

getLogging

```
public void getLogging()
```

toString

```
public java.lang.String toString()
```

Overrides:

`toString in class java.lang.Object`

Returns:

String implementation of the gameboard. Also, containing logs and money pouch.