Class StartScreen

java.lang.Object
greenfoot.World
StartScreen

---

public class **StartScreen**

extends greenfoot.World
StartScreen is, well the start screen for the game. It allows for the user to modify certain starting values of team red and team blue by clicking on different buttons. It also has the start game button.

Version:
1.2
Author:
Alex Li

Constructor Summary

StartScreen

public StartScreen()

Constructor for objects of class StartScreen, it creates all the labels, and buttons and displays them onto the world.

Method Summary

act

public void act()

Act - checks if a button is pressed and changes that buttons image to be selected. It also changes the starting money, viking count, or worker count of a team. If more than one button is pressed, the previous button will be reset. This method is called whenever the 'Act' or 'Run' button gets pressed in the environment.

Overrides:

act in class greenfoot.World

Class SpaceWorld

java.lang.Object
greenfoot.World
SpaceWorld

---

public class SpaceWorld
Extends greenfoot.World

SpaceWorld - the setting of the war. It contains all the static constants of the game along with the scoreboards and arraylists for the mines and factories of the world. It contains 2 planets where the 2 teams hold their base and perform all their activities.

Version: November 2019
Author: Albert Lai

Constructor:
SpaceWorld
public SpaceWorld (int redMoney, int blueMoney, int redViking, int blueViking, int redMiners, int blueMiners)
Creates all the initial actors of the game, sets the scoreboard stats and plays music

Parameters
redMoney - the amount of money team red starts with
blueMoney - the amount of money team blue starts with
redViking - the amount of vikings team red starts with
blueViking - the amount of vikings team blue starts with
redMiners - the amount of miners team red starts with
blueMiners - the amount of miners team blue starts with

Methods:
updateStat
public void updateStat (int board, boolean red, String statName, int change)
Updates the correct scoreboard by a certain specified value

Parameters
board - board to update (1: buildings, 2: units, 3: kills, 4: money)
red - specifies the team (true for red, false for blue)
value - the new value of the stat

getStat

public int getStat(int board, boolean red, String statName)
Retrieves a stat from the correct scoreboard

Parameters
board - board to retrieve from (1: buildings, 2: units, 3: kills, 4: money)
red - specifies the team (true for red, false for blue)

Returns
int - stat to be retrieved or -1 if invalid board

getFactoriesArrayList()
public ArrayList<Factories> getFactoriesArrayList()
Accessor for factories arraylist

Returns
Arraylist<factories> - the factories arraylist

addFactoriesArrayList
public void addFactoriesArrayList(Factories f)
Adds a factory to the factories arraylist

Parameters
f - the factory to add to the arraylist

removeFactoriesArrayList
public void removeFactoriesArrayList(Factories f)
Removes a factory from the factories arraylist

Parameters
f - the factory to be removed from the arraylist

getMinesArrayList()
public ArrayList<Mines> getMinesArrayList()
Accessor for mines arraylist

Returns
Arraylist<mines> - the mines arraylist

addMinesArrayList
public void addMinesArrayList(Mines m)
Adds a mine to the mines arraylist

Parameters
m - the mine to add to the arraylist

removeMinesArrayList
public void removeMinesArrayList(Mines m)
Removes a mine from the mines arraylist

Parameters
m - the mine to be removed from the arraylist

stopMusic
public void stopMusic()
Stops the music

Class GameOver

java.lang.Object
    greenfoot.World
        GameOver

---

public class GameOver
Extends greenfoot.World

GameOver is- the game over screen, which shows up when the game ends. It shows the stats of length of game and the kills per each side. It also has a button to allow the user to restart the game.

Version: November 2019
Author: Albert Lai

Constructor:
GameOver
public SpaceWorld (boolean red, int milliseconds, int redKills, int blueKills)
Sets the image and adds the labels and buttons

Parameters
red - specifies the winning team (true for red, false for blue)
milliseconds - how long the simulation lasted in milliseconds
redKills - amount of red kills
blueKills - amount of blue kills

Methods:
act
act()
Checks if the reStart button is pressed. This method is called whenever the 'Act' or 'Run' button gets pressed in the environment.

Class Units

- java.lang.Object
    - greenfoot.Actor
        - Units

---

public abstract class Units
extends greenfoot.Actor

Super class for all units in the game that move and have a team

Version: November 2019
Author: Alber Lai, Henry Ma

Constructor:
Units
public Units(int speed, boolean red)
Constructor for all subclasses

Parameters
speed - Speed of unit(amount moved per move)
red - Team of unit(true for red, false for blue)

Methods:
getRed()
public boolean getRed()
Returns the team of the unit

Class Workers

is a subclass of Units that creates a worker that can go to a factory and help create different vehicles or build a new factory when needed.

Version: November 2019
Author: Aristos Theocharoulas

CONSTRUCTOR
Public Workers (boolean red, boolean build)
- Creates a worker and manages the animations
- @param
    - *boolean red: true for red team, false for blue team*
    - *boolean build: true to build a factory, false to not*

Private void animateWorkersMovement()
- animation used when the worker is walking

Private void animateBuildingMovement()
- animation used when the worker is building a factory

Private void die()
- Removes the worker from the world

Private void goToFactory()
- Goes to the targeted factory

Private void build()
- worker goes to a random empty location and takes 20 seconds to build a factory

Private double getDistance(Actor a1, Actor a2)
- Gets the Distance between two actors.
- @param
    - Actor a1: get the location of actor a1
    - Actor a2: get the distance from the location of actor a1 to actor a2
- @return double distance between a1 and a2

Private void targetRandomFactory()
- targets a random factory from either the ones that exist in the world or the ones that need to be built

Private int[][] getAvailableCoords()
- Gets the available coordinates from the factories and adds them to an array
- @return int[][] the available coordinates as an array

Private boolean factoryExistsAt(int x, int y)
- checks to see if factory exists at a certain location
- @param
    - Int x: x coordinate of the world
    - Int y: y coordinate of the world
- @return boolean true if factory exists there, false if not

Parameters

Private void moveTowardTarget()
- moves toward targetFactoryCoords at the set speed

Class Miners

is a subclass of Units that creates a miner that goes to a random mine from those that exist in the world and makes $100 every 2 seconds for the team it is on.

Version: November 2019
Author: Aristos Theocharoulas

CONSTRUCTOR
Public Miners(boolean red)

- creates a miner and sets the images
- @param
  - Boolean red: true for red team, false for blue team

Methods:
Private void animateMinerMovement()

- animation used when the miner is walking

Private void die()

- removes the miner from the world

Private void mine()

- makes money for the team the miner is on every 2 seconds

Private double getDistance(Actor a1, Actor a2)

- Gets the Distance between two actors.
- @param
  - Actor a1: get the location of actor a1
  - Actor a2: get the distance from the location of actor a1 to actor a2
- @return double distance between a1 and a2

Private ArrayList<Mines> getMines()

- adds all the mines to an arraylist so that the miner can choose one to go randomly
- @return Arraylist<Mines> an arraylist of all the mines

Private void targetRandomMine()

- targets a random mine from the number of mines that exist in the world

Private void moveTowardMine()

- makes the miner move toward the targeted Mine

Class Vehicles

- java.lang.Object
    - greenfoot.Actor
        - Units
            - Vehicles

---

public abstract class Vehicles
extends Units
Abstract class for all vehicles that target and attack enemies
Version: November 2019
Author: Henry Ma, Debugging(Albert Lai and Star Xie)

Constructor:
Vehicles
public Vehicles(int speed,boolean red, int hp)
constructor that sets initial values for subclasses

Parameters
speed - Speed of the vehicle
red - Determines the team of the ship(true for red, false for blue)
hp - Max health points of the ship

Methods:
addedToWorld

public void addedToWorld(greenfoot.World w)

A Greenfoot method to run through code once when object is added to world

Parameters

w - World that healthbar is added to

getClosestEnemyBuilding

protected greenfoot.Actor getClosestEnemyBuilding()

Gets the closest enemy building relative to this actor

getClosestEnemy

protected greenfoot.Actor getClosestEnemy()

Gets closest enemy vehicle relative to this actor

getDistance

protected double getDistance(greenfoot.Actor b)

Get distance from an object relative to the object that calls this method

Parameters

b - Actor that you want the distance between

checkHp

protected void checkHp()

Checks hp of object and acts accordingly

getHit
public void getHit(int damage)
Reduces hp of object that calls this

Parameters
damage - damage that actor takes

attack

protected void attack(boolean r, greenfoot.Actor a, int damage)

Creates a bullet to damage enemy team

Parameters

r - Team the bullet is on

a - Actor that bullet turn towards

damage - Damage that bullet deals

Class Vikings

- java.lang.Object
    - greenfoot.Actor
        - Units
            - Vehicles
                - Vikings

---

public class Vikings
extends Vehicles
Vikings attacks enemy vehicles when present in the world otherwise attack buildings Deals damage through bullets
Version: November 2019
Author: Henry Ma, Debugging(Albert Lai and Star Xie)

Constructor:

Vikings

public Vikings(boolean red)

Constructor for Vikings that sets initial values and images

Parameters

red - Team of the unit(true for red, false for blue)

Methods:

act

public void act()

Act - Checks hp if zero and removes itself if true, if not vehicle will attack vehicles first then buildings if no vehicles

Class Avengers

- java.lang.Object
    - greenfoot.Actor
        - Units
            - Vehicles
                - Avengers

---

public class **Avengers**
extends Vehicles

Avengers targets and attacks buildings Shoots bullets to deal damage

Version: November 2019
Author: Henry Ma, Debugging(Albert Lai and Star Xie)

Constructor:
Avengers
public Avengers(boolean red)
Constructor for avengers that sets initial values and images

Parameters
red - Team of the avenger(true for red, false for blue)

Methods:

act

public void act()

Act - Checks if it has a target building and if so target and damage said building, otherwise find target, while also removing itself if hp is 0

Class Nightingale

- java.lang.Object
    - greenfoot.Actor
        - Units
            - Vehicles
                - Nightingale

---

public class **Nightingale**
extends Vehicles

Nightingale defends base when enemy vehicles are present otherwise attack buildings, while using bullets to deal damage

Version: November 2019
Author: Henry Ma, Debugging(Albert Lai and Star Xie)

Constructor:
Nightingale
public Nightingale(boolean red)
Constructor for Nightingale that sets initial values and images

Parameters:
red - Team of the unit(true for red, false for blue)

Methods:
act
public void act()
Act - Checks if it enemies are in the world if not it attacks buildings, while also removing itself if hp is 0

Class Hyperion

- java.lang.Object
    - greenfoot.Actor
        - Units
            - Vehicles
                - Hyperion

---

public class **Hyperion**
extends Vehicles
Hyperion targets and attacks all enemy objects while also shooting missiles instead of bullets

Version: November 2019
Author: Henry Ma, Debugging(Albert Lai and Star Xie)

Constructor:
Hyperion
public Hyperion(boolean red)
Constructor for Hyperion that sets initial values and images

Parameters
red - Team of the unit(true for red, false for blue)

Methods:
act
public void act()
Act - Checks for enemies and targets said enemies, then shoots missiles if in range

Class Buildings

java.lang.Object
    greenfoot.Actor
        Buildings

---

public abstract class Buildings
Extends greenfoot.Actor

Abstract class of all standing unmoving big structures.

Version: November 2019
Author: Albert Lai and Alex Li

Constructor:
Buildings
public Buildings(boolean red, int hp)
Calls the superclass and creates the healthbar

Parameters
red - specifies the team (true for red, false for blue)
hp - the amount of hp of the building

Methods:
addedToWorld
public void addedToWorld(World w)
Called when the object is added to the world to add the healthbar to the correct location

Parameters
w - specifies the current world the building is in

getRed
public boolean getRed()
Returns the team of the building

Returns
boolean - specifies the team (true for red, false for blue)

getHit
public void getHit(int damage)

Deals damage to the building

Parameters
damage - amount of damage to transfer to building

checkBlowUp
protected abstract void checkBlowUp()
Checks if building is destroyed. To be implemented in subclasses

Class CommandCentre

java.lang.Object
greenfoot.Actor.
Buildings
CommandCentre

---

public class CommandCentre
Extends greenfoot.Actor

CommandCentre is the commander of each side. They make the workers and miners, level up factories and mines, and order the factories to make vehicles. It lights up whenever it makes a person. When the CommandCentre explodes, the war is lost.

Version: November 2019
Author: Albert Lai

Constructor:
CommandCentre
public CommandCentre(boolean red)
Creates a Command Centre, sets the image, and starts the gametime

Parameters
red - specifies the team (true for red, false for blue)

Methods:
act
public void act()
Do whatever the CommandCentre wants to do. This method is called whenever the 'Act' or 'Run' button gets pressed in the environment.

animation
private void animation()
Makes the CommandCentre light up whenever producing a unit

checkBlowUp
protected void checkBlowUp()
Checks if the CommandCentre is destroyed and if it is, removes it from the world, creates an explosion, and ends the game by setting the GameOver world

upgrade
private void upgrade()
Randomly upgrades a mine or a factory.

addToProductionQueueUnit
private boolean addToProductionQueueUnit()
Adds a vehicle to the queue of the factory with the lowest time
Returns
boolean - true if vehicle was added to the queue and false if not

makePersonUnit
private void makePersonUnit()
Creates either a worker or a miner randomly.

Class Factories

java.lang.Object
greenfoot.Actor
Buildings
Factories

---

public class **Factories**

extends Buildings
Factory is an actor that makes ships and can be upgraded to make even better ships.
All the ships take x amount of time to build, and can be decreased by 100 milliseconds multiplied by how many workers there are in the factory. A maximum of 10 workers are allowed per factory at one instance of time. Each factory starts with one worker. The factory will play a sound when a ship is made.
New factories can be built, but a maximum of 4 per side is allowed. They can also be rebuilt if one of them is destroyed, this rebuilt factory will start at lvl 1. All of its functions are controlled by the command centre.
The factory increases in size and will play a sound when it is upgraded. The factory will not lvl up if it is making something. Its sizes are as followed (in pixels):
Lvl 1: 90 x 88
Lvl 2: 93 x 73
Lvl 3: 96 x 90
Lvl 4: 100 x 96
Version:
1.1.1
Author:
Alex Li

Constructor Summary

Factories

public Factories(boolean red)

Creates a factory belonging to either team red or team blue and sets its image. The factory will have a maximum health of 3000.

Parameters:

red - Specifies which side the factory belongs to

Method Detail

act
public void act()
Act - checks whether or not the factory has been added to the factories arraylist in SpaceWorld,

checks whether or not its production queue is empty or not very 30 acts, and checks whether or not it should play its animation
Overrides:
act in class greenfoot.Actor
addToProductionQueue
public void addToProductionQueue(int n)
Adds a unit to the factories production queue. It also changes its total production time.
Parameters:
n - an integer to specify what ship to make (1 = viking, 2 = avenger, 3 = nightingale, 4 = hyperion)

getTime
public int getTime()
Returns the time it will take to make all the units in the factory's production queue.
Returns:
Int - The total time it will take to make all the units in its queue

getFactoryLevel
public int getFactoryLevel()
Returns the factory level
Returns:
Int - The factory level

getWorkerCnt
public int getWorkerCnt()
Returns the amount of workers in the factory
Returns:
Int - The amount of workers in the factory

addWorker
public void addWorker()
Adds a worker to the factory if it has less than 10 workers

isFull
public boolean isFull()
Returns a boolean stating whether it is full or not
Returns:
Boolean - A boolean stating whether the factory is full or not

addToLvlUpCnt
public void addToLvlUpCnt()
Increments the amount of times the factory needs to level up since it can level up when making a unit

checkBlowUp
public void checkBlowUp()
Checks if the factory is destroyed and if it is, removes it from the world, and creates an explosion.
Specified by:
checkBlowUp in class Buildings

getBuilding
public boolean getBuilding()
Returns whether or not a factory is being built
Returns:
Boolean - Whether the factory is being built or not

setIsBeingBuilt
public void setIsBeingBuilt(boolean isBeingBuilt)
Sets isBeingBuilt
Parameters:
The - desired value of isBeingBuilt

Class Mines

java.lang.Object
greenfoot.Actor
Mines

---

public class Mines
Extends greenfoot.Actor

Mines are the money generating machines. Miners work in them to make money for the war effort. A maximum of 10 workers can work in each mine and the world starts off with 4 mines. Every mine starts off at level one, and can be levelled up to level 4. Higher level mines make more money per miner. Once mines are destroyed, they cannot be rebuilt.

Version: November 2019
Author: Albert Lai

Constructor:
Mines
public Mines(boolean red)
Creates a Mine and sets the image

Parameters
red - specifies the team (true for red, false for blue)

Methods:
act
public void act()
Do whatever the Mines wants to do. This method is called whenever the 'Act' or 'Run' button gets pressed in the environment.

checkBlowUp
protected void checkBlowUp()
Checks if the Mine is destroyed and if it is, removes it from the world, creates an explosion, and updates the world stat

isFull
public boolean isFull()

Checks if the mines is at full capacity

Returns
Boolean - true if mines is full and false if not

incrementMiner
public void incrementMiner()
Increases the numbers of miners by 1

makeMoney()
public void makeMoney()
Called by the miner to record the amount of money made

getMineLevel()
public int getMineLevel()
Returns the level of the mine

Returns
int - the level of the mine

levelUp
public void levelUp()
Levels up the factory and sets the image accordingly

Class Ammunition

java.lang.Object
greenfoot.Actor
Ammunition

Public abstract class Ammunition
Extends greenfoot.Actor

Abstract class of all dynamic and moving objects used to deal damage.

Version: November 2019
Author: Star Xie

Constructor:
Buildings
public Ammunition(boolean red, Actor actor, int damage, int speed)
Instantiates variables of the class

Parameters
red- specifies the team (true for red, false for blue)
actor- specifies the targeted Actor
damage- the amount of damage dealt to the object hit
speed- the speed of the current ammunition

Methods:
checkAndHit()
protected abstract checkAndHit()
Called to check if the ammunition has hit any object and deals damage if so. To be implemented in subclasses.

Class Bullet

java.lang.Object
greenfoot.Actor
Ammunition
Bullet

---

public class Bullet
Extends greenfoot.Actor

The Bullet class is a subclass of Ammunition and is where bullets are made to follow the path of the shot from spaceships in the game. A bullet will do a constant damage and will follow the trajectory it was shot at in the first place. Bullets travel in straight lines.

Version: November 2019
Author: Star Xie

Constructor:
Bullet
public Bullet(boolean red, Actor actor, int damage)
Creates a Bullet and instantiates class variables

Parameters
red- specifies the team (true for red, false for blue)
actor- specifies the targeted Actor
damage- the amount of damage dealt to the object hit

Methods:
act
public void act()
Do whatever the Bullet wants to do. This method is called whenever the ‘Act’ or ‘Run’ button gets pressed in the environment.

addedToWorld(Greenfoot.world w)
public void addedToWorld(Greenfoot.world w)
Turns towards a specified vehicle or building when the Bullet object is added to the world

checkAndHit()
public void checkAndHit()
Checks to see if the bullet has hit either a Building or a Vehicle. Decreases its HP if hit.

Class Missile

java.lang.Object
    greenfoot.Actor
        Ammunition
            Missile

Public class Missile
Extends Ammunition

Missile- A subclass of Ammunition with the purpose is to target buildings and vehicles. Upon collision, the Missile deals damage within a radius of 50 pixels and decreases the HP of all Vehicles and Buildings around it. This class is only meant to be used with the Hyperion Vehicle.

Version: November 2019
Author: Star Xie

Constructor:
Missile
public Missile(boolean red, Actor actor, int damage)
Creates a Missile and instantiates class variables

Parameters
red- specifies the team (true for red, false for blue)
actor- specifies the targeted Actor
damage- the amount of damage dealt to the object hit

Methods:
act
public void act()
Do whatever the Missile wants to do. This method is called whenever the ‘Act’ or ‘Run’ button gets pressed in the environment.

checkAndHit()
public void checkAndHit()
Checks to see if the missile has hit either a Building or a Vehicle. Decreases its HP if hit.

Class Explosion

java.lang.Object
    greenfoot.Actor
        Explosion

`public class` **`Explosion`**

`extends greenfoot.Actor`

Explosion is an actor that plays an explosion when added to to world

Version:

1.1

Author:

Alex Li

Constructor Detail

Explosion

`public Explosion()`

Creates an explosion with all needed frames

Method Detail

act

`public void act()`

Act - plays and explosion. This method is called whenever the 'Act' or 'Run' button gets pressed in the environment.

Overrides:

`act` in class `greenfoot.Actor`

Class HealthBar

- java.lang.Object
    - greenfoot.Actor
        - HealthBar

---

public class HealthBar
extends greenfoot.Actor
HealthBar creates a customizable bar. Can be used as a HealthBar (for keeping track of health of an object). Can be used as a energy/timer bar (for decreasing of a resource at a set rate).
Takes in Integers and Greenfoot Objects(ex.color)
Version: November 2019
Author: Henry Ma

Constructor:
HealthBar
public HealthBar(int x, int y, int max, greenfoot.Color m)
Creates a health bar

Parameters
x - X size/width of health bar
y - Y size/height of health bar
max - Max value of health bar(max HP)
m - Color of middle bar(reflects current HP)

Methods:
update
public void update(int newValue, greenfoot.Color m)
Updates value of current resource

Parameters:
newValue - New value of resource(update to this value)
m - Color of middle bar(reflects resource)

drawBar
private void drawBar(int current, int maxValue, Color m)
Draws bar/image for HealthBar

Parameters
current - Current value of resource
maxValue - Max value of resource
m - Color of middle bar(reflects current resource)

follow

public void follow(int x, int y)

Sets bar to location given by x and y

Parameters

x - X coordinate to be placed

y - Y coordinate to be placed

Class Timer

java.lang.Object
    greenfoot.Actor
        Timer

---

public class Timer
Extends greenfoot.Actor

Timer is an in-game countdown object. The countdown is displayed through a circle and numbers counting down inside the circle. When it reaches zero, it removes itself from the world.

Version: November 2019
Author: Albert Lai and Star Xie

Constructor:
Timer
public Timer()
Main Constructor initializes initial values and is only called by the other constructors, not intended to be called directly.

Timer
public Timer(int maxMilliSeconds)
Constructor takes one int, for objects with a max time to count down from. Both the current time and max time will be set to the same value.

Parameters
maxMilliSeconds - maximum time to start countdown at

Timer
public Timer(boolean red, int maxMilliSeconds, int milliSeconds)
Constructor takes two ints for objects with a max time to countdown from and a current time value, which will both be set accordingly. It also takes a boolean, representing the team and uses it to set the color.

Parameters
maxMilliSeconds - max amount of milliSeconds the countdown can hold
milliSeconds - time the countdown will start at
red - specifies the team (true for Red and false for Blue)

Methods:

act

public void act()

Do whatever the Timer wants to do. This method is called whenever the 'Act' or 'Run' button gets pressed in the environment.

getTime

public int getTime()

Returns the amount of time left on the timer

Returns

int - amount of time left (milliSeconds)

Class Scoreboard

java.lang.Object
    greenfoot.Actor
        Scoreboard

---

public class **Scoreboard**
extends greenfoot.Actor
Scoreboard is an actor that will track only one stat for each team (eg money, kills, etc). Multiple instances of scoreboard can be created.
Its default background colour is light-gray and its default text colour is black, this cannot be changed. However, it does have a customizable width, height and font size.
Version:
1.1.1
Author:
Alex Li

Constructor Detail

Scoreboard

public Scoreboard(int width, int height, int fontSize)
Creates a scoreboard with a custom width, height, and font size and default colours.
Parameters:
width - The desired width of the scoreboard
height - The desired height of the scoreboard
fontSize - The desired font size

Method Detail

showScore
public void showScore(boolean ShowScore)
Toggles the transparency of the scoreboard from transparent to opaque. If true, then the scoreboard will be opaque, otherwise it will be transparent
Parameters:
ShowScore - A boolean to indicate whether or not the scoreboard should be transparent or not

addStat
public void addStat(boolean isRed, java.lang.String statName, int Amount)
Adds a stat to the board.
Parameters:
isRed - If true, then the stat added will belong to team red, else it will add a stat belonging to team blue
statName - The name of the stat you want to add
Amount - The amount you want the starting value to be

updateStats
public void updateStats(boolean isRed)
Increments the stat of a team by one
Parameters:
isRed - If true, then the stat adjusted will belong to team red, else it will adjust a stat belonging to team blue
Increment - If true, then it adds one to the stat, else it subtracts one

updateStats
public void updateStats(boolean isRed, int changeAmount)
Increments the stat of a team by x
Parameters:
isRed - If true, then the stat adjusted will belong to team red, else it will adjust a stat belonging to team blue
changeAmount - The amount you want to add to the current statValue

getStat
public int getStat(boolean isRed)
Returns the stat value that a team has
Parameters:
isRed - If true, then the stat returned will belong to team red, else it will belong to team blue
Returns:
int The stat value that a team has or -1 if the team name is invalid

Class Label

java.lang.Object

greenfoot.Actor

Label

---

public class **Label**

extends greenfoot.Actor
A Label class that allows you to display a textual value on screen.
The Label is an actor, so you will need to create it, and then add it to the world in Greenfoot. If you keep a reference to the Label then you can change the text it displays.
Version:
1.2.1
Author:
Amjad Altadmri, Alex Li

Constructor Detail

Label
public Label(java.lang.String value, int fontSize)
Creates a default label with customizable font size and text. The text would have a white colour with a black outline
Parameters:
value - The text that you want to label to have
fontSize - The desired fontSize

Label
public Label(java.lang.String value, int fontSize, boolean isRed)
Create a new label, but the text colour will be red if isRed is true or blue if it is false, initialises it with the needed text and the font size.
Parameters:
value - The text that you want to label to have
fontSize - The desired fontSize
isRed - A boolean state whether the text colour is red or blue

Method Detail

setValue
public void setValue(java.lang.String value)
Sets the value as a text
Parameters:
value - The text to be show

setValue

public void setValue(int value)

Sets the value as an integer

Parameters:

value - the value to be show

setLineColor

public void setLineColor(greenfoot.Color lineColor)

Sets the line color of the text

Parameters:

lineColor - the line color of the text

setFillColor

public void setFillColor(greenfoot.Color fillColor)

Sets the fill color of the text

Parameters:

fillColor - the fill color of the text

Class TextButton

java.lang.Object
greenfoot.Actor
TextButton

---

public class **TextButton**

extends greenfoot.Actor
A Generic Button to display text that is clickable. This should be added to, and controlled by, a world.
Version:
v1.0.1
Author:
Jordan Cohen, Alex Li

Constructor Detail

TextButton
public TextButton(java.lang.String text, int textSize, boolean isRed, int statValue)
Constructs a TextButton given a String and a text size
Parameters:
String - String to be displayed
textSize - The size of the text to be displayed
isRed - Changes the colour of the text depending on isRed
statValue - The value of the stat that the button holds

TextButton

public TextButton(java.lang.String text,int textSize, boolean regularBtn)
Creates a default button that only holds text. It has black text, a black border, and a white background. This button will not display as a regular button if regularBtn is false

Parameters:

String - String to be displayed

textSize - The size of the text to be displayed

regularBtn - A boolean to state whether this is a regular button or not

Method Detail

updateMe
public void updateMe(java.lang.String text)
Updates the current TextButton text
Parameters:
text - The text to be displayed

Update
public void Update()
Updates button to be pressed (changes its image)

buttonReset
public void buttonReset()
Resets the button to no be pressed (changes its image)

getStatValue
public int getStatValue()
Returns the statValue that the button holds
Returns:
int The statValue that the button holds