Class TitleScreen

java.lang.Object
greenfoot.World
TitleScreen

```
public class TitleScreen

extends greenfoot.World
```

Title screen is a world where it allows the player to start a new game, load a previous save, or open the controls page.

Version:

1.1

Author:

Alex Li

Constructor Detail

TitleScreen

```
public TitleScreen()
```

Creates a new title screen where it allows the user to start a new game, load a save, and access the controls

Method Detail

act

```
public void act()
```

Act - do whatever the TitleScreen wants to do. This method is called whenever the 'Act' or 'Run' button gets pressed in the environment. Checks for mouse input and acts accordingly.

Overrides:

`act` in class `greenfoot.World`

Class Options

java.lang.Object
    greenfoot.World
        Options

```
public class Options

extends greenfoot.World
```

Options is the world where the user is told how to play the game. It displays all the controls and game features

Version:

0.3

Author:

Alex

Constructor Detail

Options

```
public Options()
```

Constructor for objects of class Options. Adds all the text and images to the world for the first tutorial page

Method Detail

act

```
public void act()
```

Act - do whatever Options wants to do. This method is called whenever the 'Act' or 'Run' button gets pressed in the environment. Checks for mouse input from the user and acts accordingly.

Overrides:

`act` in class `greenfoot.World`

Class PauseWorld

java.lang.Object
    greenfoot.World
        PauseWorld

```
public class PauseWorld
extends greenfoot.World
```

PauseWorld - world that is set when player clicks pause, opens the store, or ends the game.

Version:
January 2020
Author:
Albert Lai

Constructor Detail

public PauseWorld(java.lang.String menuType, GameWorld gameWorld)
Constructor for objects of class PauseWorld, called to create a PauseMenu

Parameters:
menuType - Type of the menu
gameWorld - gameWorld of the player

public PauseWorld(GameWorld gameWorld Player player, ItemInfo itemInfo)
Constructor for objects of class PauseWorld, called to create a Store

Parameters:
gameWorld - gameWorld of the player
menuType - Type of the menu
player - player in the world
itemInfo - itemInfo in the world

public PauseWorld(int score, boolean isWin)
Constructor for objects of class PauseWorld, called to create a gameOver screen

Parameters:
score - score of the player
isWin - true if the player won, false if not

Method Detail

public void act()
act - checks for button clicks
public void closeWorld()
closeWorld - sets the game back to gameWorld

Class GameWorld

java.lang.Object
    greenfoot.World
        GameWorld

```
public class GameWorld
extends greenfoot.World
```

GameWorld is the world where the game takes place. It holds the player, actors of the game, 2D array of actors, and itemInfo. It is loaded from a text file and has methods to create, modify, and delete them. It checks for keyboard presses to open menus and contains methods to modify its objects.

Gungeon is a remake of the classic Gungeon. The player travels through chambers/rooms, shooting enemies and avoiding traps to beat the game. It contains a store to allow the user to buy items like more guns, health refills, and ammo. It can be saved to a text file so the user can continue playing from a checkpoint. The tutorial shows how the game works and its controls. Enter the Gungeon!

Images credits to Star Xie, free4kwallpapers.com, and Enter the Gungeon.
Music credits to ZapSplat, soundcloud.com, Super Mario, and Enter the Gungeon.

Version:
January 2020
Author:
Albert Lai

Constructor Detail

public GameWorld()
Base Constructor - not to be called directly

public GameWorld(boolean newGame)
Constructor to initialize the world from "create a new game" or "load saved game"

Parameters:
boolean - true if creating new game, false loading from old game

public GameWorld(int curLevel, int fromLevel, Player player, ItemInfo itemInfo)
Constructor to switch between rooms

Parameters:
curLevel - current level of the world
fromLevel - level of the previous room
player - player transferred from the other world
itemInfo - ItemInfo transferred from the other world

Method Detail

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

public void updateScore(int scoreBoost, int moneyBoost)
updateScore - called when enemies die to update the score, money, and kills

Parameters:
scoreBoost - amount to increase the score by
moneyBoost - amount to increase the money by

public void gameOver(boolean win)
gameOver - checks if player is dead and ends the game, called by player class

Parameters:
win - true if player has won, false if not

public void closeWorld(int newLevel)
closeWorld - closes the current world

Parameters:

newLevel - level of the world the player is in

public void resumeWorld()
resumeWorld - resumes music playing

public void switchWorld(int newLevel)
switchWorld - switch to a different room

Parameters:
newLevel - new level to switch to

public static int convert(int index)
convert - takes the index value and converts to greenfoot coordinate value

Parameters:
index - index of 2D array to be converted

Returns:
int - greenfoot coordinate value

public boolean isWall(int firstInd, int secondInd)
isWall - takes position in 2D array and returns whether or not a wall exists

Parameters:
firstInd - first index of 2D array
secondInd - second index of 2D array

Returns:
boolean - whether or not there's a wall

public boolean isObstacle(int firstInd, int secondInd)
isObstacle - takes position in 2D array and returns whether or not an obstacle exists

Parameters:
firstInd - first index of 2D array
secondInd - second index of 2D array

Returns:

boolean whether or not there's a wall

public void createTextFiles()
createTextFiles - create new text files for the game (not to be called during gameplay)

Ammunition

java.lang.Object

greenfoot.Actor

Ammunition

`public abstract class` **`Ammunition`**

`extends greenfoot.Actor`

The ammunition class exchanges information with the player class and the weapons class. These subclasses hold information such as the current number of available ammunition, the direction visually, checks to see if it has hit an enemy or character, and lastly reloads the ammunition.

Version:

0.2

Author:

Star Xie, Clarence Chau

Constructor Detail

Ammunition

```
public Ammunition (int xCoord, int yCoord, int damage, int
speed, boolean isEnemy)
```

Constructs ammunition depending on the x-coordinates, y-coordinates, the damage it deals when hitting an opposing character, the speed of the bullet, and who possesses the bullet.

Parameters:

xCoord - gives the x-coordinates
yCoord - gives the y-coordinates
damage- gives the damage dealt when hit
Speed- gives the speed of the bullet
isEnemy- gives if the bullet shot is from an enemy or not

Public Methods:

Protected abstract int checkAmmo();
checkAmmo - Returns the specific ammo number. To be implemented in subclasses

Returns:
int- returns the ammo amount

Protected abstract void reloadAmmo();
reloadAmmo - Sets the ammunition to its full amount. To be implemented in subclasses

Protected void checkAndHit();
checkAndHit - checks if the Bullet has hit a player, enemy, boss, or walls and deals damage to their HP if collision is detected.

Protected void addToWorld(World w);
addedToWorld - Makes sure if the bullet object is added to the world before altering it's rotation

Parameters:
World- takes a world parameter to check if the object has been added to the world

public void act()
Act - do whatever Button wants to do. This method is called whenever the 'Act' or 'Run' button gets pressed in the environment. Checks to see if the bullet is hitting something and moves if a bullet is spawned into the world.

PistolBullet

java.lang.Object
greenfoot.Actor
PistolBullet

Public Methods:

Public void reloadAmmo();
reloadAmmo - `Sets the ammunition to its full amount. To be implemented in subclasses`
Public int checkAmmo();
checkAmmo - `Returns the specific ammo number. To be implemented in subclasses`

RifleBullet

java.lang.Object

greenfoot.Actor

RifleBullet

Public Methods:

Public void reloadAmmo();

Sets the ammo back to 50

Public int checkAmmo();

Returns an integer of the remaining ammo the player has

ShotgunBullet

java.lang.Object

greenfoot.Actor

ShotgunBullet

Public Methods:

public void reloadAmmo();

Sets the ammo back to 5

public int checkAmmo();

Returns an integer of the remaining ammo the player has

Class Boss

- java.lang.Object
    - greenfoot.Actor
        - Boss

`public abstract class` **`Boss`**

- `extends greenfoot.Actor`
  Super class for bosses, bosses are a type of enemy that you fight after finishing a certain amount of levels
  Version: January 16, 2020
  Author: Henry Ma

Method Detail

getDamaged

`protected void getDamaged(int damage)`

Inflict damage dependent on the parameters given, also checks for death

Parameters:

`damage` **- Amount of damage to deal**

Class BeholsterBoss

- java.lang.Object
    - greenfoot.Actor
        - Boss
            - BeholsterBoss

`public class` **`BeholsterBoss`**

- `extends Boss`

  Beholster Boss of the game, has multiple different attacks, stronger attacks happen less, stronger attacks become harder to dodge (second boss)

  Version: January 17, 2020

  Author: Henry Ma

Constructor Detail

BeholsterBoss

```
public BeholsterBoss()
```

Constructor for beholster boss, set and initialize values for boss

Method Detail

act

```
public void act()
```

Act - Beholster stays still while charging up for attacks/attacking

Overrides:

```
act in class greenfoot.Actor
```

Method Detail

addedToWorld

```
protected void addedToWorld(greenfoot.World world)
```

Greenfoot method, called when object is added to world

Overrides:

```
addedToWorld in class greenfoot.Actor
```

Class BlobBoss

- java.lang.Object
  - greenfoot.Actor
    - BlobBoss

```
public class BlobBoss
extends greenfoot.Actor
```

Boss of the game, has multiple different attacks, stronger attacks happen less frequently, stronger

attacks become harder to dodge

Version: January 16, 2020

Author: Henry Ma

Constructor Detail

BlobBoss

```
public BlobBoss()
```

Constructor for blob boss, set and initialize values for boss

Method Detail

act

```
public void act()
```

Act - Blob stays still while charging up for attacks/attacking

Overrides:

```
act in class greenfoot.Actor
```

addedToWorld

```
protected void addedToWorld(greenfoot.World world)
```

Greenfoot method, called when object is added to world

Overrides:

```
addedToWorld in class greenfoot.Actor
```

Class Button

java.lang.Object

greenfoot.Actor

Button

```
public class Button
extends greenfoot.Actor
```

A button without a background that can have a custom font, font size, and text colour. It will highlight a different colour when the user mouses over the button, this colour can be customized too. Multiple buttons can be added to the world.

Version:

v1.2

Author:

Alex Li

Constructor Detail

Button

```
public Button(java.lang.String text, int fontSize)
```

Constructs a Button with a given String and a specified font size. It will have a default text colour of light gray and a default font of calibri

Parameters:

`text` - String value to display

`fontSize` - The font size, as an integer

Button

```
public Button(java.lang.String text, int fontSize, int txtR, int txtG, int txtB, int highlightR, int highlightG, int highlightB)
```

Constructs a Button with a given String, a specified font size, and custom text colour and highlight colour. It will use the default font, calibri

Parameters:

`text` - String value to display

`fontSize` - The font size, as an integer

`txtR` - The intensity of red in the text's colour. The R value in RGB

`txtG` - The intensity of green in the text's colour. The G value in RGB

`txtB` - The intensity of blue in the text's colour. The B value in RGB

`highlightR` - The intensity of red in the highlight colour of the button. The R value in RGB

`highlightG` - The intensity of green in the highlight colour of the button. The G value in RGB

`highlightB` - The intensity of blue in the highlight colour of the button. The B value in RGB

Button

```
public Button(java.lang.String text, java.lang.String fontName, int fontSize)
```

Constructs a Button with a given String and a specified font and font size. It will have a default text colour of light gray

Parameters:

`text` - String value to display

`fontName` - The name of the font you want the label to use. Must be a valid font in greenfoot and java

`fontSize` - The font size, as an integer

Button

```
public Button(java.lang.String text, java.lang.String fontName, int fontSize, int txtR, int txtG, int txtB, int highlightR, int highlightG, int highlightB)
```

Constructs a Button with a given String, a specified font and font size, and custom text colour and

highlight colour.

Parameters:

`text` - String value to display
`fontName` - The name of the font you want the label to use. Must be a valid font in greenfoot and java
`fontSize` - The font size, as an integer
`txtR` - The intensity of red in the text's colour. The R value in RGB
`txtG` - The intensity of green in the text's colour. The G value in RGB
`txtB` - The intensity of blue in the text's colour. The B value in RGB
`highlightR` - The intensity of red in the highlight colour of the button. The R value in RGB
`highlightG` - The intensity of green in the highlight colour of the button. The G value in RGB
`highlightB` - The intensity of blue in the highlight colour of the button. The B value in RGB

Method Detail

act

```
public void act()
```

Act - do whatever Button wants to do. This method is called whenever the 'Act' or 'Run' button gets pressed in the environment. Checks whether or not the mouse is hovering over the button, if it is highlight it, otherwise don't

Overrides:

`act` in class `greenfoot.Actor`

Cursor

- java.lang.Object
  - greenfoot.Actor
    - Cursor

```
public class Cursor
```

- `extends greenfoot.Actor`

  Cursor- a class meant to act as an image for the mouse to have the appearance as a targeting device.

  Version:

  `January 2020`

  Author:

  `Star Xie`

`Cursor()`

Constructor - creates a Cursor with a scaled image

`public void act()`
Act - do whatever the Cursor wants to do. This method is called whenever the 'Act' or 'Run' button gets pressed in the environment.

Class Enemy

- **java.lang.Object**
  - **greenfoot.Actor**
    - **Enemy**

All Implemented Interfaces:
`AnimationInterface`

`public abstract class Enemy`
`extends greenfoot.Actor`
`Implements AnimationInterface`
Abstract class for all enemies in the game that target and attack the player
Version: **`January 16, 2020`**
Author: **`Combined(Henry Ma), Debugging(Star, Albert), Pathfinding(Albert), Animations(Star)`**

Method Detail

animate

`protected void animate(int[] closestTileCoordinates)`
Animates the object, checks the coordinates given in the parameters to determine the direction of the animation

Parameters:
`closestTileCoordinates - The coordinates of the object to animate/face towards`

attack

`protected abstract void attack()`
All enemies must contain an attack method(method/pattern on how they attack)

die

`protected void die()`
Removes object from the world properly

findClosestAdjacentTileTowardsPlayer

```
public int[] findClosestAdjacentTileTowardsPlayer()
```

Gets the coordinates of a tile that is closest to player

Returns:

```
int[] Returns the coordinates in an array of the tile closest to player
```

getDamaged

```
public void getDamaged(int damage)
```

Call to inflict damage, checks for death as well

Parameters:

```
damage - Amount of damage this object takes
```

moveTowardsPlayer

```
protected void moveTowardsPlayer()
```

Move towards player and checks conditions for attacking and moving

Class BulletEnemy

- **java.lang.Object**
  - **greenfoot.Actor**
    - **Enemy**
      - **BulletEnemy**

All Implemented Interfaces:
`AnimationInterface`

`public class BulletEnemy`

`extends Enemy`

The beginner enemies, bullet enemies are average with normal fire rate and a normal shooting pattern

Version: **`January 16, 2020`**

Author: **`Henry Ma`**

Constructor Detail

BulletEnemy

```
public BulletEnemy()
```

Constructor for bullet enemies that sets the initial values

Method Detail

act

```
public void act()
```

Act - Move towards the player and check for line of sight before attempting to shoot

Overrides:

```
act in class greenfoot.Actor
```

addedToWorld

```
protected void addedToWorld(greenfoot.World world)
```

Greenfoot method, called when this object is added to the world

Overrides:

```
addedToWorld in class greenfoot.Actor
```

animateMovementDown

```
public void animateMovementDown()
```

Changes the image for animation of down movement

animateMovementLeft

```
public void animateMovementLeft()
```

Changes the image for animation of left movement

animateMovementRight

```
public void animateMovementRight()
```

Changes the image for animation of right movement

animateMovementDown

```
public void animateMovementUp()
```

Changes the image for animation of up movement

createImages

```
public static void createImages()
```

Create images for the class to use during animations

attack

```
public void attack()
```

Creates bullets of a given type depending on the enemy that calls it attacks differ from enemy to enemy

Specified by:

```
attack in class Enemy
```

Class ShotgunEnemy

- ava.lang.Object

- greenfoot.Actor
  - Enemy
    - ShotgunEnemy

All Implemented Interfaces:
`AnimationInterface`

```
public class ShotgunEnemy

extends Enemy
```

This enemy attacks by firing a spread of bullets. creates difficulty through the increase in the amount of bullets

Version: `January 16, 2020`
Author: `Henry Ma`

Constructor Detail

ShotgunEnemy

```
public ShotgunEnemy()
```

Constructor for shotgun enemies that sets the initial values

Method Detail

act

```
public void act()
```

Act - Move towards the player and check for line of sight before attempting to shoot

Overrides:

`act in class greenfoot.Actor`

addedToWorld

```
protected void addedToWorld(greenfoot.World world)
```

Greenfoot method, called when this object is added to the world

Overrides:

`addedToWorld in class greenfoot.Actor`

animateMovementDown

```
public void animateMovementDown()
```

Changes the image for animation of down movement

animateMovementLeft

```
public void animateMovementLeft()
```

Changes the image for animation of left movement

animateMovementRight

`public void animateMovementRight()`

Changes the image for animation of right movement

animateMovementDown

`public void animateMovementUp()`

Changes the image for animation of up movement

createImages

`public static void createImages()`

Create images for the class to use during animations

attack

`public void attack()`

Creates bullets of a given type depending on the enemy that calls it attacks differ from enemy to enemy

Class SniperEnemy

- **java.lang.Object**
  - **greenfoot.Actor**
    - **Enemy**
      - **SniperEnemy**

All Implemented Interfaces:
`AnimationInterface`

`public class SniperEnemy`
`extends Enemy`
This enemy attacks similar to bullet enemies but their attacks are slower with their bullets being faster creates difficulty through the speed of the bullet
Version: **`January 16, 2020`**
Author: **`Henry Ma`**

Constructor Detail

ShotgunEnemy

`public SniperEnemy()`

Constructor for sniper enemies that sets the initial values

Method Detail

act

`public void act()`

Act - Move towards the player and check for line of sight before attempting to shoot

Overrides:

act in class greenfoot.Actor

addedToWorld

protected void addedToWorld(greenfoot.World world)

Greenfoot method, called when this object is added to the world

Overrides:

addedToWorld in class greenfoot.Actor

animateMovementDown

public void animateMovementDown()

Changes the image for animation of down movement

animateMovementLeft

public void animateMovementLeft()

Changes the image for animation of left movement

animateMovementRight

public void animateMovementRight()

Changes the image for animation of right movement

animateMovementDown

public void animateMovementUp()

Changes the image for animation of up movement

createImages

public static void createImages()

Create images for the class to use during animations

attack

public void attack()

Creates bullets of a given type depending on the enemy that calls it attacks differ from enemy to enemy

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 ItemInfo

java.lang.Object
greenfoot.Actor
ItemInfo

```
public class ItemInfo

extends greenfoot.Actor
```

ItemInfo tracks and displays the current gun, the amount of bullets in the gun's magazine, the total ammo (excluding what is already in the magazine, and the amount of kills, the amount of money and the current score the player has. Although you can create multiple instances of ItemInfo, only one should exist in a world at any given time.

It consists of a 247 by 79 gray rectangle with a line that divides the gun information and the player stats

Version:

0.9

Author:

Alex Li

Constructor Detail

ItemInfo

```
public ItemInfo(int currentGun, int totalAmmo, int currentAmmo, int money)
```

Creates an ItemInfo board and adds all the starting information of the player

Parameters:

`currentGun` - The current gun the player is holding. 0 = pistol, 1 = rifle, 2 = shotgun

`totalAmmo` - The total ammo of a specific type the player has. If the current gun is the pistol, totalAmmo = -1

`currentAmmo` - The amount of bullets in gun's magazine

`money` - The player's starting money

Method Detail

addedToWorld

```
public void addedToWorld(greenfoot.World world)
```

Adds all the necessary labels to the world

Overrides:

`addedToWorld` in class `greenfoot.Actor`

Parameters:

`world` - The Greenfoot World where ItemInfo is added into.

updateGun

```
public void updateGun(int currentGun, int totalAmmo, int currentAmmo)
```

Updates the current gun the user is holding

Parameters:

`currentGun` - The current gun the user is using. 0 = pistol, 1 = rifle, 2 = shotgun

`totalAmmo` - The total ammo of a specific gun the player has, excluding the magazine. If the current gun is the pistol, totalAmmo = -1

`currentAmmo` - The amount of bullets in gun's magazine

incrementKills

```
public void incrementKills()
```

Increases total kills by one

updateMoney

```
public void updateMoney(int increment)
```

Updates the amount of money the user has

Parameters:

`increment` - the amount of money the user gains or spends

updateScore

```
public void updateScore(int increment)
```

Updates the user's score

Parameters:

`increment` - the amount of score the user gains

updateAmmo

```
public void updateAmmo(int totalAmmo, int currentAmmo)
```

Updates the user's total ammo and ammo in the magazine whenever the user reloads their gun

Parameters:

`totalAmmo` - The total ammo of a specific gun the player has excluding the ammo in the gun's magazine. If the current gun is the pistol, totalAmmo = -1

`currentAmmo` - The amount of bullets in gun's magazine

updateAmmo

```
public void updateAmmo()
```

Updates the user's ammo in the magazine whenever the user shoots

getScore

```
public int getScore()
```

Returns the player's score

Returns:

int The score the player has

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 with or without a background. You can create a label with a custom font, a custom text size, custom letter colours, and background colours.

The Label is an actor, so you will need to create it, and then add it to the world in Greenfoot.

Version:

1.5

Author:

Alex Li

Constructor Detail

Label

```
public Label(java.lang.String text, int fontSize,boolean isTransparent)
```

Creates a default label with customizable font size and text, and an option to have a transparent background. It will use the default font of calibri.

Parameters:

`text` - The text that you want to label to have
`fontSize` - The desired fontSize
`isTransparent` - Must be set to true for the background to be transparent, otherwise it will be white

Label

`public Label(java.lang.String text, int fontSize, int txtR, int txtG, int txtB, boolean isTransparent)`

Creates a default label with customizable font size, text, text colour, and an option to have a transparent background. It will use the default font of calibri.

Parameters:

`text` - The text that you want to label to have
`fontSize` - The desired fontSize
`txtR` - The intensity of red in the text's colour. The R value in RGB
`txtG` - The intensity of green in the text's colour. The G value in RGB
`txtB` - The intensity of blue in the text's colour. The B value in RGB
`isTransparent` - Must be set to true for the background to be transparent, otherwise it will be white

Label

`public Label(java.lang.String text, int fontSize,  int txtR, int txtG, int txtB, int backR, int backG, int backB)`

Creates a default label with customizable font size, text, text colour, and background colour. It will use the default font of calibri.

Parameters:

`text` - The text that you want to label to have
`fontSize` - The desired fontSize
`txtR` - The intensity of red in the text's colour. The R value in RGB
`txtG` - The intensity of green in the text's colour. The G value in RGB
`txtB` - The intensity of blue in the text's colour. The B value in RGB
`backR` - The intensity of red in the background colour. The R value in RGB
`backG` - The intensity of green in the background colour. The G value in RGB
`backB` - The intensity of blue in the background colour. The B value in RGB

Label

`public Label(java.lang.String text, java.lang.String fontName, int fontSize, boolean isTransparent)`

Creates a default label with customizable font size, font and text, and an option to have a transparent background.

Parameters:

`text` - The text that you want to label to have
`fontName` - The name of the font you want the label to use. Must be a valid font in greenfoot and java
`fontSize` - The desired fontSize
`isTransparent` - Must be set to true for the background to be transparent, otherwise it will be white

Label

`public Label(java.lang.String text, java.lang.String fontName, int fontSize, int txtR, int txtG, int txtB, boolean isTransparent)`

Creates a default label with customizable font, font size, text, text colour, and an option to have a

transparent background.

Parameters:

`text` - The text that you want to label to have
`fontName` - The name of the font you want the label to use. Must be a valid font in greenfoot and java
`fontSize` - The desired fontSize
`txtR` - The intensity of red in the text's colour. The R value in RGB
`txtG` - The intensity of green in the text's colour. The G value in RGB
`txtB` - The intensity of blue in the text's colour. The B value in RGB
`isTransparent` - Must be set to true for the background to be transparent, otherwise it will be white

Label

```
public Label(java.lang.String text, java.lang.String fontName, int
fontSize, int txtR, int txtG, int txtB,  int backR, int backG, int backB)
```

Creates a default label with customizable font, font size, text, text colour, and background colour.

Parameters:

`text` - The text that you want to label to have
`fontName` - The name of the font you want the label to use. Must be a valid font in greenfoot and java
`fontSize` - The desired fontSize
`txtR` - The intensity of red in the text's colour. The R value in RGB
`txtG` - The intensity of green in the text's colour. The G value in RGB
`txtB` - The intensity of blue in the text's colour. The B value in RGB
`backR` - The intensity of red in the background colour. The R value in RGB
`backG` - The intensity of green in the background colour. The G value in RGB
`backB` - The intensity of blue in the background colour. The B value in RGB

Menu

java.lang.Object
    greenfoot.Actor
        Menu

```
public abstract class Menu
extends greenfoot.Actor
```

Menu is the abstract superclass of PauseMenu and StoreMenu. It contains methods to handle button clicks.

Version:
January 2020
Author:
Albert Lai

Method Detail

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

PauseMenu

java.lang.Object
    greenfoot.Actor
        Menu
            PauseMenu

```
public class PauseMenu
extends Menu
```

PauseMenu is a menu class that allows the user to pause the game, giving them the option to either resume or exit and save.

Version:
January 2020
Author:
Albert Lai

Constructor Detail

public PauseMenu()
Constructor for objects of class PauseMenu.

Method Detail

public void addedToWorld(greenfoot.World w)
addedtoWorld - adds labels and buttons to the world

Parameters:
w - current world of the menu

StoreMenu

java.lang.Object
    greenfoot.Actor
        Menu
            StoreMenu

```
public class StoreMenu
extends Menu
```

StoreMenu is the store that exists in the game. The user can open the Store anytime when playing the game and choose to buy items/powerups or equip themselves with an item.

Version:
January 2020
Author:
Albert Lai

Constructor Detail

public StoreMenu(Player player, ItemInfo itemInfo)
Constructor for Store Menu

Parameters:
player - reference to player object in the GameWorld
itemInfo - reference to ItemInfo object in the GameWorld

Constructor Detail

public static void createImages()
createImages - static method to create all the images for the items

public void addedToWorld(greenfoot.World w)
addedToWorld - add buttons, labels, and StoreItems to the world

Parameters:
w - current world of the StoreMenu

public void getLastPressed(StoreItem item, java.lang.String itemName, int itemCost)
getLastPressed - get the properties of the last item pressed

Parameters:
item - reference to StoreItem object
itemName - item name of store object
itemCost - item cost of store object

Obstacles

java.lang.Object
    greenfoot.Actor
        Obstacles

```
public abstract class Obstacles
extends greenfoot.Actor
```

Obstacles is the abstract superclass of all obstacles in the world. It contains methods to damage enemies.

Version:
January 2020
Author:
Albert Lai

Constructor Detail

public Obstacles(int damage)
Constructor for obstacles

Parameters:
damage - damage inflicted by obstacle

public Obstacles(int damage, int firstInd, int secondInd)
Constructor for obstacles

Parameters:
damage - damage inflicted by obstacle
firstInd - first index of array
secondInd - second index of array

Arrows

java.lang.Object
    greenfoot.Actor
        Obstacles
            Arrows

```
public class Arrows
extends Obstacles
```

Arrows is an obstacle that inflicts damage on any player or enemy that touches it. It includes an animation of the arrow hitting its target.

Version:
January 2020
Author:
Albert Lai

Constructor Detail

public Arrows()
Constructor for Arrows

Method Detail

public void addedToWorld(greenfoot.World w)
addedToWorld - sets coordinates

Parameters:
w - Current world of the actor

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

Door

java.lang.Object
    greenfoot.Actor

Obstacles
Door

```
public class Door
extends Obstacles
```

Door is an obstacle that provides passages to other rooms. When all the enemies in the room are defeated, the door will open.

Version:
January 2020
Author:
Albert Lai

Constructor Detail

Door(int curLevel, boolean isComplete)
Constructor for Door

Parameters:
curLevel - the level the door leads to

Method Detail

public void addedToWorld(greenfoot.World w)
addedToWorld - sets the image

Parameters:
w - the current world

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

public void completeLevel()
completeLevel - called when the player completes the level

public boolean getComplete()
getLevel - returns the level of the door

Returns:
boolean - true if the level is complete and false if not

Fire

java.lang.Object
    greenfoot.Actor
        Obstacles
            Fire

```
public class Fire
extends Obstacles
```

Fire is an obstacle that inflicts damage on the player/enemies that touch it

Version:
January 2020
Author:
Albert Lai

Constructor Detail

public Fire(int firstInd, int secondInd)
Constructor for Fire

Parameters:
firstInd - the first index of the array
secondInd - the second index of the array

Method Detail

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

Spikes

java.lang.Object
    greenfoot.Actor
        Obstacles
            Spikes

```
public class Spikes
extends Obstacles
```

Spikes is an obstacle that inflicts damage on the player/enemies that touch it

Version:
January 2020
Author:
Albert Lai

Constructor Detail

public Spikes(int firstInd, int secondInd)
Constructor for Spikes

Parameters:
firstInd - the first index of the array
secondInd - the second index of the array

Method Detail

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

Walls

java.lang.Object
    greenfoot.Actor
        Obstacles
            Walls

```
public class Walls
```

`extends Obstacles`

Walls are obstacles that the player cannot pass through.

Version:
January 2020
Author:
Albert Lai

Constructor Detail

public Walls(int firstInd, int secondInd)
Constructor for Walls

Parameters:
firstInd - the first index of the array
secondInd - the second index of the array

Method Detail

public void addedToWorld(greenfoot.World w)
addedToWorld - sets the image

Parameters:
w - the current world

public Walls(String name)
Constructor for Walls in the center of the screen

Parameters:
String - name of the file

Player

java.lang.Object
greenfoot.Actor
Player
All Implemented Interfaces:
`AnimationInterface`

```
public class Player
extends greenfoot.Actor
implements AnimationInterface
```

Player is the main character in the game. It is saved and loaded from a text file and holds the gun. It is controlled by the user with key presses, allowing it to move around and shoot. It has stats like level, health, speed, and money. It implements the animation interface to move smoothly in the 4 different directions.

Version:

January 2020

Author:

Albert Lai, Star Xie, and Clarence Chau

Constructor Detail

public Player(java.io.File txtFile,ItemInfo itemInfo)
Constructor for Player Class

Parameters:
txtFile - txtFile storing the player
itemInfo - itemInfo of the world

Method Detail

public void addedToWorld(greenfoot.World w)
addedToWorld - adds guns and resource bars to the world

Parameters:
w - current world of the player

public static void createImages()
createImages - create images used by player class if not already done so

public void act()

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

public java.lang.String getCurrentGun()
getCurrentgun - returns the name of the current gun

Returns:
String the name of the current gun

public boolean hasGun(java.lang.String gunName)
hasGun - returns whether or not the player has the specified gun

Parameters:
gunName - Name of the gun

Returns:
boolean true if the player has the gun and false if not

public void changeGun()
changeGun - changes the gun to the next one in the arraylist

public void newGun(java.lang.String gun)
newGun - adds a new gun to the player

Parameters:
gun - name of the gun

public boolean loseOneHeart()
loseOneHeart - takes one half-heart away from the player

Returns:
boolean - true if the player was damaged

public void addOneHeart()
addOneHeart - adds a half-heart to the player

public void animateMovementUp()

animateMovementUp - animation when player moves up.

Specified by:
animateMovementUp in interface AnimationInterface

public void animateMovementDown()
animateMovementDown - animation when player moves down.

Specified by:
animateMovementDown in interface AnimationInterface

public void animateMovementRight()
animateMovementRight - animation when player moves right.

Specified by:
animateMovementRight in interface AnimationInterface

public void animateMovementLeft()
animateMovementLeft - animation when player moves left.

Specified by:
animateMovementLeft in interface AnimationInterface

public int canReload(Weapon w)
canReload - checks if there is enough ammo for reload and if there is, update itemInfo and reloadBar

Parameters:
w - current weapon of the player

Returns:
int number of bullets in mag

public void saveData()
saveData - save data from player into text file

public void parseData()

parseData - load data from Player.txt

public int getMoney()
getMoney - get the amount of money

Returns:
int the amount of money

public void setMoney(int amount)
setMoney - set the money

Parameters:
amount - the amount of money to add/remove

public void setScore(int amount)
setScore - set the score

Parameters:
amount - score to add

public void setKills()
incrementKills - increment kills by one

public int getItemNumber(java.lang.String name)
getItemNumber - gets the amount of item that the player has

Parameters:
name - Item name

Returns:
int amount of item

public void changeItemNumber(java.lang.String name, int amount)
changeItemNumber - adds or removes an item

Parameters:
name - Item name

amount - Amount to add or remove

public void speedBoost()
speedBoost - increase the speed

public int getCurLevel()
getCurLevel - returns the current level

Returns:
int current level of the player

public void changeCurLevel(int amount)
changeCurLevel - changes level of the player

Parameters:
amount - new level

public int getMaxLevel()
getMaxLevel - returns max level of the player

Returns:
int max level of the player

public void incrementMaxLevel()
incrementMaxLevel - increases max level

public int getHearts()
getHearts - returns number of hearts of the player

Returns:
int number of hearts of the player

public void reduceAmmo()
reduceAmmo - updates reloadBar

Resource

- java.lang.Object
    - greenfoot.Actor
        - Resource

```
public class Resource
extends greenfoot.Actor
```

Resource class is managed by the resource bar manager(most methods in this class are only called by resource bar manager) Resource class helps to create a bar (this class are the images that show up as the bar)

Version: January 16, 2020

Author: Henry Ma

Constructor Detail

Resource

```
public Resource(greenfoot.GreenfootImage first)
```

Constructor 1: Used for any kind of resource(Usually used for ammo bar)

Parameters:

```
first - Image of the resource
```

Resource

```
public Resource(greenfoot.GreenfootImage first, greenfoot.GreenfootImage
second)
```

Constructor 2: Used for the health bar as hearts have to images

Parameters:

```
first - The first image of the heart(full heart)
second - The second image of the heart(half heart)
```

Method Detail

getStatus

```
public int getStatus()
```

Get the status of the heart

Returns:

```
Returns 1 if at half heart or 2 if at full(returns -1 if not a heart
resource)
```

switchImage

public boolean switchImage()

Can either heal or remove a heart(half) depending on its current status

ResourceBarManager

- java.lang.Object
    - greenfoot.Actor
        - ResourceBarManager

---

public class **ResourceBarManager**

extends greenfoot.Actor

Manages and displays the resource that is created through the parameters, creates a bar that can be managed through methods in this class Manages the resource class through a stack, call methods in this class to manage the resources it holds/manages

Version: January 16, 2020

Author: Henry Ma

Constructor Detail

ResourceBarManager

public ResourceBarManager(int max, int interval, int x, int y, greenfoot.GreenfootImage firstImage, greenfoot.GreenfootImage secondImage, greenfoot.World world)

Constructor 1: Used for health bar(creates a bar of images specified in the parameters)

Parameters:

max - Max amount of this resource possible
interval - Spacing of the images/resources
x - X coordinate of the bar
y - Y coordinate of the bar
firstImage - First image of the resource
secondImage - Second image of the resource
world - World the bar is to be added in (do not add the resource bar manager to the world)

ResourceBarManager

public ResourceBarManager(int max, int cur, int interval, int x, int y, greenfoot.GreenfootImage resourceImage, greenfoot.World world)

Constructor 1: Used for any kind of resource(creates a bar of images specified in the parameters)

Parameters:
`max - Max amount of this resource possible`
`cur - Current value for the resource(starting value)`
`interval - Spacing of the images/resources`
`x - X coordinate of the bar`
`y - Y coordinate of the bar`
`resourceImage - Image of the resource`
`world - World the bar is to be added in (do not add the resource bar manager to the world)`

Method Detail

addBarToWorld
`public void addBarToWorld(greenfoot.World world)`
Adds the resources in the stack to the world, creating a bar

reduceAmmo
`public boolean reduceAmmo(greenfoot.World world)`
Remove a resource/ammo from the stack, in the world remove one as well

reduceHealth
`public boolean reduceHealth(int hits, greenfoot.World world)`
Take damage depending amount specified in the parameters(only works if this is a health bar manager)

refillAmmo
`public boolean refillAmmo(greenfoot.World world)`
Refills the stack and re-adds the resources back to the world

refillHealth
`public boolean refillHealth(int Amount, greenfoot.World world)`
Heal/increase the health of the health bar

remove
`public void remove(greenfoot.World world)`
Removes a resource from the stack and the world

StoreItem

java.lang.Object
    greenfoot.Actor
        StoreItem

`public class StoreItem`
`extends greenfoot.Actor`

StoreItem is the class of items found in the store. They can be clicked to be selected.

Version:
January 2020
Author:
Albert Lai

Constructor Detail

public StoreItem(greenfoot.GreenfootImage itemImage, java.lang.String item, StoreMenu storeMenu, int cost)
Constructor for StoreItem
Parameters:
itemImage - image of the item
item - name of the item
StoreMenu - reference to StoreMenu object
cost - cost of the object

Method Detail

public void addedToWorld(greenfoot.World w)
addedToWorld - adds the label to the world

Parameters:
w - world of the StoreItem

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

public void changeImage()
changeImage - changes the image of StoreItem

Class TextSizeFinder

java.lang.Object

greenfoot.Actor

TextSizeFinder

```
public class TextSizeFinder
extends greenfoot.Actor
```

TextSizeFinder is a helper class that can return the width standard height of a String with a given font and font size. It uses Java's Font, not Greenfoot's. It is not meant to be added to the world

Version:

0.2

Author:

Alex Li

Constructor Detail

TextSizeFinder

```
public TextSizeFinder()
```

Creates an instance of TextSizeFinder

Method Detail

getTextWidth

```
public int getTextWidth(java.lang.String text, int fontSize)
```

Returns the width of a String in calibri with a custom font size

Parameters:

`text` - The text you want to find the width of

`fontSize` - The font size

Returns:

int The width of the String

getTextHeight

```
public int getTextHeight(java.lang.String text, int fontSize)
```

Returns the standard height of the String in calibri with a custom font size

Parameters:

`text` - The text you want to find the width of

`fontSize` - The font size

Returns:

int The standard height of the String

getTextWidth

`public int getTextWidth(java.lang.String text, java.lang.String fontName, int fontSize)`

Returns the width of a String in a custom font with a custom font size

Parameters:

`text` - The text you want to find the width of

`fontName` - The name of the custom font. Must be a valid font in greenfoot and java

`fontSize` - The font size

Returns:

int The width of the String

getTextHeight

`public int getTextHeight(java.lang.String text, java.lang.String fontName, int fontSize)`

Returns the standard height of the String in calibri with a custom font size

Parameters:

`text` - The text you want to find the width of

`fontName` - The name of the custom font. Must be a valid font in greenfoot and java

`fontSize` - The font size

Returns:

int The standard height of the String

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)

Weapons

public Weapon(ItemInfo itemInfo, Player player, int bulletDamage, int bulletSpeed, long fireRate, long bulletReadyTime , long reloadTime, int magSize, int ammoInMag)
Constructor - Initiates the instance variables
Parameters:
itemInfo - the current ItemInfo object
player - the current Player object
bulletDamage - the damage that the bullet will do to enemies
bulletSpeed - the speed that the bullet will have
fireRate - the time between each bullet when user is holding down the left click in milliseconds
bulletReadyTime - the time bet each bullet when user is clicking the left click in milliseconds
reloadTime - the time it takes for the weapon to reload
magSize - the number of magazines the user has
ammoInMag - the current ammo in the magazine
reloadTimer - true to enable the timer when reloading, false to not

abstract protected void createBullet(int xcoord, int ycoord);
Public void act()
Checks for mouse clicks and button presses to shoot or reload
Parameters:
xcoord - x coordinate of the player
ycoord - y coordinate of the player

private void stopFiring()
Stops Shooting Bullets from the weapon

private boolean isReloading()
Checks if the gun is currently reloading, returns true if yes, false if not

private void startReload()
Starts the reloading process

private void shoot()

Calls the createBullet abstract method and creates a suitable bullet for each gun and decreases the ammo

public int getAmmo()
Gets the current ammo in the magazine
Returns:
Int - current ammo in the magazine

public long getFireRate()
Gets the fire rate of the current weapon
Returns:
Long - fire rate of weapon

public int getMagSize()
Gets the magazine size of the weapon
Returns:
Int - magazine size of weapon

public long getReloadTime()
Gets the reload time of the weapon
Returns:
Long - Reload time of the bullet

public long getBulletReadyTime()
Gets the time that the bullet is ready to shoot
Returns:
Long - time that the bullet is ready to shoot

public int getBulletDamage()
Gets the damage of the bullet
Returns:
Int - damage of the bullet

public int getBulletSpeed()
Gets the speed of the bullet
Returns:
Int - speed of the bullet

Pistol

Public Methods:
Public Pistol(ItemInfo itemInfo, Player player, int bulletDamage, int bulletSpeed, long fireRate, long bulletReadyTime , long reloadTime, int magSize, int ammoInMag)
Calls the superclass constructor and scales and sets the gun's image
Protected abstract void createBullet(int xcoord, int ycoord)
Creates a pistol bullet that targets xcoord and ycoord

Rifle

Public Methods:
Public Rifle(ItemInfo itemInfo, Player player, int bulletDamage, int bulletSpeed, long fireRate, long bulletReadyTime , long reloadTime, int magSize, int ammoInMag)
Calls the superclass constructor and scales and sets the gun's image
Protected abstract void createBullet(int xcoord, int ycoord)
Creates a rifle bullet that targets xcoord and ycoord

Shotgun

Public Methods:
Public Shotgun(ItemInfo itemInfo, Player player, int bulletDamage, int bulletSpeed, long fireRate, long bulletReadyTime , long reloadTime, int magSize, int ammoInMag)
Calls the superclass constructor and scales and sets the gun's image
Protected abstract void createBullet(int xcoord, int ycoord)
Creates a shotgun bullet that shoots 5 pellets each shot and targets xcoord and ycoord

AnimationInterface

```
java.lang.Object
      AnimationInterface
```

```
public interface AnimationInterface
```

The animation interface is designed to be implemented by enemies or characters requiring movement in all 4 directions of freedom. The interface ensures all enemies and characters follow through with this animation as required.

Version:

```
January 2020
```

Author:

```
Star Xie
```

Pathfinding

java.lang.Object
Pathfinding

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

Pathfinding is used by the Enemies class to find an optimal path to the player. It utilizes BFS (breadth first search) and backtracking.

Version:
January 2020
Author:
Albert Lai

Method Detail

public static int[] nextCoord(int startX, int startY, GameWorld world, Player player)
nextCoord - finds the next tile in the 2D array the enemy should move to

Parameters:
startX - X coordinate of enemy

secondY - Y coordinate of enemy
world - world of the enemy
player - player in the world

Returns:
int[] new coordinates to move to