

Table Of Content

collisiondetection	2
Collision	2
display	4
Display	4
entities	6
Alligator	7
AlligatorBank	8
Bus	9
Car	10
Entity	12
Frog	16
Log	18
Taxi	19
Truck	20
Turtle	22
game	24
Game	24
Launcher	27
graphics	29
Assets	29
ImageLoader	31
SpriteSheet	32
input	34
KeyManager	34
MouseManager	36
objectsarrays	40
Player	40
RiverItems	41
Vehicles	44
score	48
FileReader	48
Score	49
states	51
GameStates	51
Playing	53
Index	55

Package collisiondetection

Class Summary

[Collision](#)

collisiondetection

Class Collision

```
java.lang.Object
|
+--collisiondetection.Collision
```

< [Constructors](#) > < [Methods](#) >

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

Author:

Eder

Constructors

Collision

```
public Collision()
```

Methods

frogAndAlligators

```
public int frogAndAlligators(Frog frog,
                             java.util.ArrayList alligator)
```

Parameters:

frog -
alligator -

Returns:

frogAndLogs

```
public int frogAndLogs(Frog frog,  
                      java.util.ArrayList log)
```

Parameters:

frog -
log -

Returns:

frogAndTurtles

```
public int frogAndTurtles(Frog frog,  
                        java.util.ArrayList turtle)
```

Parameters:

frog -
turtle -

Returns:

frogAndVehicles

```
public boolean frogAndVehicles(Frog frog,  
                             Vehicles vehicles)
```

Parameters:

frog -
vehicles -

Returns:

Package display

Class Summary

[Display](#)

This is the class responsible for creating and closing the window of the game.

display

Class Display

```
java.lang.Object
|
+--display.Display
```

< [Constructors](#) > < [Methods](#) >

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

This is the class responsible for creating and closing the window of the game.

Author:

Eder Paz ; Neil Blake ; Logan Wedel

Constructors

Display

```
public Display(java.lang.String title,
               int width,
               int height)
```

The constructor defines the title, width and height of the game that was passed to it.

Parameters:

title - Title of the game
width - Width of the game window
height - Height of the game window

Methods

close

```
public void close()
```

This method closes the frame of the game by calling the dispose() method.

createDisplay

```
public void createDisplay()
```

This method creates the game frame and canvas, configures them and opens the window of the game.

getCanvas

```
public java.awt.Canvas getCanvas()
```

Getter for the game canvas.

Returns:

Canvas object

getFrame

```
public javax.swing.JFrame getFrame()
```

Getter for the game frame.

Returns:

Frame object

Package entities

Class Summary

[Alligator](#)

This is the class that defines every alligator that shows up in the river.
It extends the abstract class entity.

[AlligatorBank](#)

This is the class that defines every alligator that shows up in the river bank.
It extends the abstract class Entity.

[Bus](#)

This is the class that defines every Bus that shows up on the road.
It extends the abstract class Entity.

[Car](#)

This is the class that defines every car that shows up on the road.
It extends the abstract class Entity.

[Entity](#)

This is the main class for the entities in the game, every entity that the game has will extend this one.
This class contains every variables, methods and objects that are common to every different entity object in the game.

[Frog](#)

This is the class that defines the frogs that are used by the player.
It extends the abstract class entity.

[Log](#)

This is the class that defines every log that shows up in the river.
It extends the abstract class entity.

[Taxi](#)

This is the class that defines every taxi that shows up on the road.
It extends the abstract class Entity.

[Truck](#)

This is the class that defines every Truck that shows up on the road;

[Turtle](#)

This is the class that defines every turtle group that show up in the river.
It extends the abstract class Entity.

entities

Class Alligator

```
java.lang.Object
|
+--Entity
|
+--entities.Alligator
```

< [Constructors](#) > < [Methods](#) >

```
public class Alligator
extends Entity
```

This is the class that defines every alligator that shows up in the river.
It extends the abstract class entity.

Author:

Eder Paz ; Neil Blake ; Logan Wedel

Constructors

Alligator

```
public Alligator(Game game,
                 int pos)
```

Defines the alligator's width, height, initial position, speed and direction in which the alligator has to move to.

Parameters:

game - Instance of the game so the alligator can rely on games variables.
pos - Defines the initial position of the alligator in the river

Methods

render

```
public void render(java.awt.Graphics g)
```

Draws the alligator on the screen according to its x, y position and the figure it is define by the tick() method.

Overrides:

[render](#) in class [Entity](#)

tick

```
public void tick()
```

Upgrades the x position of the alligator and defines which alligator image should be drawn by the render() method;

Overrides:

[tick](#) in class [Entity](#)

entities

Class AlligatorBank

```
java.lang.Object
|
+--Entity
    |
    +--entities.AlligatorBank
```

< [Constructors](#) > < [Methods](#) >

```
public class AlligatorBank
extends Entity
```

This is the class that defines every alligator that shows up in the river bank.
It extends the abstract class Entity.

Author:

Eder Paz ; Neil Blake ; Logan Wedel

Constructors

AlligatorBank

```
public AlligatorBank(Game game)
```

Defines the river bank alligator's width, height.

Defines the random initial position of the river bank alligator in one of the river banks.

Parameters:

game - Instance of the game so that the bank alligator can rely on the game variables.

Methods

render

```
public void render(java.awt.Graphics g)
```

Draws the river bank alligator on the screen according to its x, y position and the figure it is define by the tick() method.

Overrides:

[render](#) in class [Entity](#)

tick

```
public void tick()
```

Controls how much time the alligator is in one river bank and gives it a new random position.

Defines which alligator image should be drawn by the render() method.

Overrides:

[tick](#) in class [Entity](#)

entities

Class Bus

```
java.lang.Object
|
+-- Entity
|
+-- entities.Bus
```

< [Constructors](#) > < [Methods](#) >

```
public class Bus
extends Entity
```

This is the class that defines every Bus that shows up on the road.

It extends the abstract class Entity.

Author:

Eder Paz ; Neil Blake ; Logan Wedel

Constructors

Bus

```
public Bus(Game game,  
          int pos)
```

Defines the bus' width, height, initial position, speed and direction in which the bus has to move to.

Parameters:

game - Instance of the game so that the bus can rely on the game's variables.
pos - Defines the initial position of the bus on the road.

Methods

render

```
public void render(java.awt.Graphics g)
```

Draws the alligator on the screen according to its x, y position and the image defined the the constructor.

Overrides:

[render](#) in class [Entity](#)

tick

```
public void tick()
```

Upgrade the bus x position.

Overrides:

[tick](#) in class [Entity](#)

entities

Class Car

```
java.lang.Object  
|  
+--Entity  
|  
+--entities.Car
```

< [Constructors](#) > < [Methods](#) >

```
public class Car  
extends Entity
```

This is the class that defines every car that shows up on the road.

It extends the abstract class Entity.

Author:

Eder Paz ; Neil Blake ; Logan Wedel

Constructors

Car

```
public Car(Game game,  
          int pos)
```

Defines the car's width, height, initial position, speed and direction in which the bus has to move to.

Parameters:

game - Instance of the game so the car can rely on the game variables.

pos - Defines the initial position of the car on the road.

Methods

render

```
public void render(java.awt.Graphics g)
```

Draws the car on the screen according to its x, y position and the image defined by the constructor.

Overrides:

[render](#) in class [Entity](#)

tick

```
public void tick()
```

Upgrade the car's x position.

Overrides:

[tick](#) in class [Entity](#)

entities

Class Entity

```
java.lang.Object
|
+--entities.Entity
```

Direct Known Subclasses:

[Alligator](#), [AlligatorBank](#), [Bus](#), [Car](#), [Frog](#), [Log](#), [Taxi](#), [Truck](#), [Turtle](#)

< [Fields](#) > < [Constructors](#) > < [Methods](#) >

```
public abstract class Entity
extends java.lang.Object
```

This is the main class for the entities in the game, every entity that the game has will extend this one.

This class contains every variables, methods and objects that are common to every different entity object in the game.

Author:

Eder Paz ; Neil Blake ; Logan Wedel

Fields

alliBank_height

```
public static final int alliBank_height
    Default value for the river bank alligator height.
```

alliBank_width

```
public static final int alliBank_width
    Default value for the river bank alligator width.
```

alli_height

```
public static final int alli_height
    Default value for the river alligator width.
```

alli_width

```
public static final int alli_width
    Default value for the river alligator width.
```

car_height

```
public static final int car_height
```

Default value for the car height.

car_width

```
public static final int car_width
```

Default value for the car width.

log_height

```
public static final int log_height
```

Default value for log height.

player_height

```
public static final int player_height
```

Default value for the player height.

player_width

```
public static final int player_width
```

Default value for the player width.

truck_height

```
public static final int truck_height
```

Default value for truck and bus height.

truck_width

```
public static final int truck_width
```

Default value for truck and bus width.

turtle_height

```
public static final int turtle_height
```

Default value for the turtle height.

turtle_width

```
public static final int turtle_width
    Default value for the turtle width.
```

Constructors

Entity

```
public Entity(Game game,
              float x,
              float y,
              int width,
              int height)
```

Defines the entity object initial position and size.

Creates the random object.

Creates the rectangle object used in the collision detection.

Records an instance of the game so that all the entities objects in the game can rely on the game variables.

Parameters:

game - Instance of the game.
x - Entity object initial x position on the screen.
y - Entity object initial y position on the screen.
width - Entity object width.
height - Entity object height.

Methods

getBounds

```
public java.awt.Rectangle getBounds()
```

Getter for the rectangle object used in collision detection.

Returns:

Entity's rectangle object that defines the are that the enemy holds in the screen.

getGame

```
public Game getGame()
```

Getter for the main game object.

Returns:

Instance of the game so the game private variables can be accessed.

getHeight

```
public int getHeight()
```

Getter for the entity's height.

Returns:

Entity object height

getSpeed

```
public float getSpeed()
```

Returns:

getWidth

```
public int getWidth()
```

Getter for the entity's width.

Returns:

Entity object width.

getX

```
public float getX()
```

Getter for the entity's x position.

Returns:

Entity object x position on the screen.

getY

```
public float getY()
```

Getter for the entity's y position.

Returns:

Entity object y position on the screen.

render

```
public abstract void render(java.awt.Graphics g)
```

Draws the entity object on the screen according to its position and respective image.

Parameters:

g - Graphic object used to draw the images.

tick

```
public abstract void tick()
```

Upgrade the entity object variables.

entities

Class Frog

```
java.lang.Object
|
+-- Entity
|
+-- entities.Frog
```

< [Constructors](#) > < [Methods](#) >

```
public class Frog
extends Entity
```

This is the class that defines the frogs that are used by the player.

It extends the abstract class entity.

Author:

Eder Paz ; Neil Blake ; Logan Wedel

Constructors

Frog

```
public Frog(Game game)
```

Defines the player size, initial position on the screen and initial life amount.

Parameters:

game - Instance of game so that the player can rely on the game variables.

Methods

goToInitialPosition

```
public void goToInitialPosition()
```

Define what happens when the player dies, either by being hit by a vehicle, sinking in the river or time's up.

goToPosition

```
public void goToPosition(float x,  
                          float y)
```

Parameters:

x -

y -

isStopped

```
public boolean isStopped()
```

Returns:

render

```
public void render(java.awt.Graphics g)
```

Draws the player on the screen according to its x and y position and the respective image according

to its movement defined by the tick() method.

Overrides:

[render](#) in class [Entity](#)

setX

```
public void setX(float speed)
```

Parameters:

x -

tick

```
public void tick()
```

Checks if one of the keys used to move the player have been pressed and move the player according to it.

Defines which frog image has to be drawn by the render() method the movement done as well.

Overrides:

[tick](#) in class [Entity](#)

entities

Class Log

```
java.lang.Object
|
+--Entity
|
+--entities.Log
```

< [Constructors](#) > < [Methods](#) >

```
public class Log
extends Entity
```

This is the class that defines every log that shows up in the river.

It extends the abstract class entity.

Author:

Eder Paz ; Neil Blake ; Logan Wedel

Constructors

Log

```
public Log(Game game,
           int pos,
           int width)
```

Defines the log's width, height, initial position, speed and direction in which the alligator has to move to.

Parameters:

game - Instance of the game so the log can rely on games variables.
pos - Defines the initial position of the log in the river.
width - Used to define the log's width.

Methods

render

```
public void render(java.awt.Graphics g)
```

Draws the log on the screen according to its x and y positions defined by the tick() method.

Overrides:

[render](#) in class [Entity](#)

tick

```
public void tick()
```

Upgrades the x position of the log on the screen;

Overrides:

[tick](#) in class [Entity](#)

entities

Class Taxi

```
java.lang.Object
|
+--Entity
    |
    +--entities.Taxi
```

< [Constructors](#) > < [Methods](#) >

```
public class Taxi
extends Entity
```

This is the class that defines every taxi that shows up on the road.

It extends the abstract class Entity.

Author:

Eder Paz ; Neil Blake ; Logan Wedel

Constructors

Taxi

```
public Taxi(Game game,  
           int pos)
```

Defines the taxi's width, height, initial position, speed and direction in which the bus has to move to.

Parameters:

game - Instance of game so that the taxi can rely on the game variables.
pos - Defines the initial position of the taxi on the screen.

Methods

render

```
public void render(java.awt.Graphics g)
```

Draws the taxi on the screen according to its x and y positions.

Overrides:

[render](#) in class [Entity](#)

tick

```
public void tick()
```

Upgrade the taxi's x position.

Overrides:

[tick](#) in class [Entity](#)

entities

Class Truck

```
java.lang.Object  
|  
+--Entity  
|  
+--entities.Truck
```

< [Constructors](#) > < [Methods](#) >

```
public class Truck  
extends Entity
```

This is the class that defines every Truck that shows up on the road;

Author:

Eder Paz ; Neil Blake ; Logan Wedel

Constructors

Truck

```
public Truck(Game game,  
             int pos)
```

Defines the truck's width, height, initial position, speed and direction in which the bus has to move to.

Parameters:

game - Instance of game so that the truck can rely on the game variables.
pos - Defines the initial position of the truck on the screen.

Methods

render

```
public void render(java.awt.Graphics g)
```

Draws the truck image on the screen according to its x and y positions and its respective image defined by the tick() method.

Overrides:

[render](#) in class [Entity](#)

tick

```
public void tick()
```

Upgrade the truck's x position.

Overrides:

[tick](#) in class [Entity](#)

entities

Class Turtle

```
java.lang.Object
|
+--Entity
|
+--entities.Turtle
```

< [Constructors](#) > < [Methods](#) >

```
public class Turtle
extends Entity
```

This is the class that defines every turtle group that show up in the river.
It extends the abstract class Entity.

Author:

Eder Paz ; Neil Blake ; Logan Wedel

Constructors

Turtle

```
public Turtle(Game game,
             int pos,
             int amountTurtle)
```

Defines the turtles line width, height, initial position, speed and direction in which the alligator has to move to.

Adjust the width of the turtle object to how many turtles are being asked to be created.

Parameters:

game - Instance of game so that the turtle line can rely on the game variables.
pos - Defines the turtle line initial position on the screen.
amountTurtle - Defines how many turtles needs to be created in the turtle line.

Methods

render

```
public void render(java.awt.Graphics g)
```

Draws the turtles line on the screen according to its x and y position and the direction the are moving to.

Overrides:

[render](#) in class [Entity](#)

tick

```
public void tick()
```

Upgrade the turtles line x position and define the turtle image that has to be drawn by the render() method.

Overrides:

[tick](#) in class [Entity](#)

Package game

Class Summary

[Game](#)

This class is the base of the entire game.

It contains the methods to create a new game, opens the game windows and contains the game loop that keeps the game running.

[Launcher](#)

This program simulates an alternative version of the game Frogger, an Arcade game created in 1981.

The original game can be accessed on the web site:
<http://www.bigmoneyarcade.com/games/frogger>.

game

Class Game

```
java.lang.Object
|
+--game.Game
```

All Implemented Interfaces:

java.lang.Runnable

< [Constructors](#) > < [Methods](#) >

```
public class Game
extends java.lang.Object
implements java.lang.Runnable
```

This class is the base of the entire game.

It contains the methods to create a new game, opens the game windows and contains the game loop that keeps the game running.

Author:

Eder Paz; Neil Blake; Logan Wedel

Constructors

Game

```
public Game(java.lang.String title,  
            int width,  
            int height)
```

The game title, width and height are defined.

The objects for reading from the keyboard and the mouse are also initiated here.

Parameters:

title - Defines the name of the game

width - Defines the width of the game screen

height - Defines the height of the game screen

Methods

getDefaultSpeed

```
public float getDefaultSpeed()
```

Getter for the default speed of the game.

Returns:

Game default speed variable

getHeight

```
public int getHeight()
```

Getter for the game height.

Returns:

The height of the game window

getKeyManager

```
public KeyManager getKeyManager()
```

Getter for the object that reads the keyboard inputs.

Returns:

KeyManager class instance.

getManager

```
public MouseListener getManager()
```

Getter for the object that reads the mouse inputs.

Returns:

MouseListener class instance.

getWidth

```
public int getWidth()
```

Getter for the game width.

Returns:

The width of the game window

run

```
public void run()
```

This method is called when the game thread is started, right after the start() method.

It initiates the display, the assets and the states of the game.

It also contains the MAIN GAME LOOP;

setDefaultSpeed

```
public void setDefaultSpeed(float speed)
```

Setter for the default speed of the game.

Parameters:

speed - Value in which the game default speed is going to be set to.

start

```
public synchronized void start()
```

This methods actually starts the game.

It creates a new thread and starts it.

stop

```
public synchronized void stop()
```

This game ends the game by stopping the game thread.

game

Class Launcher

```
java.lang.Object
|
+--game.Launcher
```

< [Constructors](#) > < [Methods](#) >

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

This program simulates an alternative version of the game Frogger, an Arcade game created in 1981.

The original game can be accessed on the web site: <http://www.bigmoneyarcade.com/games/frogger>.

References: The game was created following:

- Subjects shown on the book "An Introduction to programming using java"
- The you tube channel "Java Game Development Series" - <https://www.youtube.com/playlist?list=PLWms450>
- The you tube channel "New Beginner 2D Programming" - <https://www.youtube.com/playlist?list=PLah6faXAg>

Author:

Eder Paz; Neil Blake; Logan Wedel

Version:

1.0 Created: 03/31/2016

Constructors

Launcher

```
public Launcher()
```

Methods

main

```
public static void main(java.lang.String[] args)
```

Launcher of the game.

Creates a new game object and start it.

Parameters:

args - Not used in the game

Package graphics

Class Summary

[Assets](#)

This is the class the loads and holds all the images used on the game.

[ImageLoader](#)

This is the class responsible to actually loads the image the the folder which contains the sprite sheets of the game.

[SpriteSheet](#)

This class holds the entire sprite sheet of the game.

graphics

Class Assets

```
java.lang.Object
|
+--graphics.Assets
```

< [Fields](#) > < [Constructors](#) > < [Methods](#) >

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

This is the class the loads and holds all the images used on the game.

Author:

Eder Paz ; Neil Blake ; Logan Wedel

Fields

alligator

```
public static java.awt.image.BufferedImage[][] alligator
    Images of the game alligators.
```

alligatorBank

```
public static java.awt.image.BufferedImage[] alligatorBank
    Images of the river bank alligator.
```

bgnd

```
public static java.awt.image.BufferedImage bgnd  
    Image of the background of the game.
```

bus

```
public static java.awt.image.BufferedImage[] bus  
    Images of the game buses.
```

car

```
public static java.awt.image.BufferedImage[] car  
    Images of the game cars.
```

frog

```
public static java.awt.image.BufferedImage[][] frog  
    Images of the frog.
```

log

```
public static java.awt.image.BufferedImage log  
    Image of the log.
```

taxi

```
public static java.awt.image.BufferedImage[] taxi  
    Images of the game taxis.
```

truck

```
public static java.awt.image.BufferedImage[] truck  
    Images of the game trucks.
```

turtle

```
public static java.awt.image.BufferedImage[][] turtle  
    Images of the game turtles.
```

Constructors

Assets

```
public Assets()
```

Methods

init

```
public static void init()
```

Load all the Sprite sheets objects and images that are doing to be used in the game.

graphics

Class ImageLoader

```
java.lang.Object  
|  
+--graphics.ImageLoader
```

< [Constructors](#) > < [Methods](#) >

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

This is the class responsible to actually loads the image the the folder which contains the sprite sheets of the game.

Author:

Eder Paz ; Neil Blake ; Logan Wedel

Constructors

ImageLoader

```
public ImageLoader()
```

Methods

loadImage

```
public static java.awt.image.BufferedImage loadImage(java.lang.String  
fileName)
```

Loads and returns the image that is saved in resource folder.

Parameters:

fileName - Name of the file that is in the resource folder.

Returns:

Image saved of the file.

graphics

Class SpriteSheet

```
java.lang.Object  
|  
+--graphics.SpriteSheet
```

< [Constructors](#) > < [Methods](#) >

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

This class holds the entire sprite sheet of the game.

Author:

Eder Paz ; Neil Blake ; Logan Wedel

Constructors

SpriteSheet

```
public SpriteSheet(java.awt.image.BufferedImage sheet)
```

Makes a copy of the buffered image passed to it to its own buffered image instance.

Parameters:

sheet - Buffered image to be saved.

Methods

crop

```
public java.awt.image.BufferedImage crop(int x,  
                                           int y,  
                                           int width,  
                                           int height)
```

Used to load an smaller image that is contained inside the sprite sheet object image.

Parameters:

- x - First x coordinate pixel of the image, at its top right.
- y - First y coordinate pixel of the image, at its top right.
- width - Width of the image that has to be cropped.
- height - Height of the image that has to be cropped.

Returns:

The sub Image the is saved inside the sprite sheet object within the coordinates passed.

Package input

Class Summary

[KeyManager](#)

This is the class that identifies all the actions that are performed on the keyboard.

[MouseListener](#)

This is the class that identifies all the actions that are performed by the mouse.

It holds boolean variables so that is easy to detected in the game if the mouse is used.

input

Class KeyManager

```
java.lang.Object
|
+--input.KeyManager
```

All Implemented Interfaces:

java.awt.event.KeyListener

< [Constructors](#) > < [Methods](#) >

```
public class KeyManager
extends java.lang.Object
implements java.awt.event.KeyListener
```

This is the class that identifies all the actions that are performed on the keyboard.

Author:

Eder Paz ; Neil Blake ; Logan Wedel

Constructors

KeyManager

```
public KeyManager()
```

Initiate the boolean array that is used to store which key that action took place on.

Methods

Down

```
public boolean Down()
```

Getter for the boolean variable that indicates to move down.

Returns:

Boolean variable down.

Left

```
public boolean Left()
```

Getter for the boolean variable that indicates to move left.

Returns:

Boolean variable left.

Right

```
public boolean Right()
```

Getter for the boolean variable that indicates to move right.

Returns:

Boolean variable right.

Up

```
public boolean Up()
```

Getter for the boolean variable that indicates to move up.

Returns:

Boolean variable up.

keyPressed

```
public void keyPressed(java.awt.event.KeyEvent e)
```

Runs every time a key is pressed on the keyboard and stores its respective boolean variable to true.

keyReleased

```
public void keyReleased(java.awt.event.KeyEvent e)
```

Runs every time a key is released on the keyboard and stores its respective boolean variable to true.

keyTyped

```
public void keyTyped(java.awt.event.KeyEvent e)
```

Runs every time a key is typed on the keyboard.
Not used in the game.

tick

```
public void tick()
```

Upgrades the variables that the game uses.
Set them to its respective value on the boolean array.

input

Class MouseManager

```
java.lang.Object  
|  
+--input.MouseManager
```

All Implemented Interfaces:

java.awt.event.MouseListener, java.awt.event.MouseMotionListener

< [Constructors](#) > < [Methods](#) >

```
public class MouseManager  
extends java.lang.Object  
implements java.awt.event.MouseListener, java.awt.event.MouseMotionListener
```

This is the class that identifies all the actions that are performed by the mouse.
It holds boolean variables so that is easy to detected in the game if the mouse is used.

Author:

Eder Paz ; Neil Blake ; Logan Wedel

Constructors

MouseListener

```
public MouseListener()
```

Methods

getX

```
public int getX()
```

Getter for the mouse position on the screen.

Returns:

Mouse x position on the screen.

getY

```
public int getY()
```

Getter for the mouse position on the screen.

Returns:

Mouse y position on the screen.

isLeftPressed

```
public boolean isLeftPressed()
```

Getter for the boolean variable that says if the right button of the mouse is pressed or not

Returns:

True for pressed, false for not pressed.

isRightPressed

```
public boolean isRightPressed()
```

Getter for the boolean variable that says if the left button of the mouse is pressed or not.

Returns:

True for pressed, false for not pressed.

mouseClicked

```
public void mouseClicked(java.awt.event.MouseEvent e)
```

Runs every time the mouse is clicked, that means pressing and releasing it.

mouseDragged

```
public void mouseDragged(java.awt.event.MouseEvent e)
```

Runs every time the mouse click and drags.

Not used in the game.

mouseEntered

```
public void mouseEntered(java.awt.event.MouseEvent e)
```

Runs every time the mouse enters the screen.

Not used in the game.

mouseExited

```
public void mouseExited(java.awt.event.MouseEvent e)
```

Runs every time the mouse exits the screen.

Not used in the game.

mouseMoved

```
public void mouseMoved(java.awt.event.MouseEvent e)
```

Stores the mouse x and y position every time the mouse moves on the screen.

mousePressed

```
public void mousePressed(java.awt.event.MouseEvent e)
```

Set the respective mouse button boolean variable to true.

mouseReleased

```
public void mouseReleased(java.awt.event.MouseEvent e)
```

Set the respective mouse button boolean variable to true.

Package objectsarrays

Class Summary

[Player](#)

[RiverItems](#)

This class holds the linked lists that helps the creation of the objects of the game.

[Vehicles](#)

This class holds the linked lists that helps the creation of the objects of the game.

objectsarrays

Class Player

```
java.lang.Object
|
+--objectsarrays.Player
```

< [Constructors](#) > < [Methods](#) >

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

Author:

Eder

Constructors

Player

```
public Player(Game game)
```

Parameters:

game -

Methods

Death

```
public void Death()
```

getFrog

```
public Frog getFrog(int index)
```

Parameters:

index -

Returns:

objectsarrays

Class RiverItems

```
java.lang.Object
|
+--objectsarrays.RiverItems
```

< [Constructors](#) > < [Methods](#) >

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

This class holds the linked lists that helps the creation of the objects of the game.

Author:

Eder Paz ; Neil Blake ; Logan Wedel

Constructors

RiverItems

```
public RiverItems()
```

Methods

addAlligator

```
public void addAlligator(Alligator alligator)
```

Adds a new alligator to the alligator to the alligator linked list.

Parameters:

alligator - Alligator object to be added to the list.

addLog

```
public void addLog(Log log)
```

Adds a new log on the log linked list.

Parameters:

log - Log object to be added to the list.

addTurtle

```
public void addTurtle(Turtle turtle)
```

Adds a new turtle line to the turtle linked list.

Parameters:

turtle - Turtle object to be added to the list.

getAlligators

```
public java.util.ArrayList getAlligators()
```

Returns:

getLogs

```
public java.util.ArrayList getLogs()
```

Returns:

getTurtles

```
public java.util.ArrayList getTurtles()
```

Returns:

removeAlligator

```
public void removeAlligator(Alligator alligator)
```

Removes an alligator from the alligator linked list.

Parameters:

alligator - Alligator to be removed from the linked list.

removeLog

```
public void removeLog(Log log)
```

Removes a log from the log linked list.

Parameters:

log - Log to be removed from the link.

removeTurtle

```
public void removeTurtle(Turtle turtle)
```

Removes a turtle line from the turtle linked list.

Parameters:

turtle - Turtle list to be removed from the link.

render

```
public void render(java.awt.Graphics g)
```

Goes through every position available on the linked lists and call the objects render() method.

Parameters:

g - Graphics object used to draw images on the game window.

tick

```
public void tick()
```

Goes through every position available on the linked links and tests the position of the object on the screen.

If it is already completely out of the screen, the respective object is removed, if it is not, its tick() method is called

objectsarrays

Class Vehicles

```
java.lang.Object
|
+--objectsarrays.Vehicles
```

< [Constructors](#) > < [Methods](#) >

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

This class holds the linked lists that helps the creation of the objects of the game.

Author:

Eder Paz ; Neil Blake ; Logan Wedel

Constructors

Vehicles

```
public Vehicles()
```

Methods

addBus

```
public void addBus(Bus bus)
```

Adds a new bus to the bus linked list.

Parameters:

bus - Bus object to be added to the list.

addCar

```
public void addCar(Car car)
```

Adds a new car the the car linked list.

Parameters:

car - Car object to be added to the list.

addTaxi

```
public void addTaxi(Taxi taxi)
```

Adds a new taxi to the taxi linked list.

Parameters:

taxi - Taxi object to be added to the list.

addTruck

```
public void addTruck(Truck truck)
```

Adds a new truck to the truck linked list.

Parameters:

truck - Truck object to be added to the list.

getBuses

```
public java.util.ArrayList getBuses()
```

Returns:

getCars

```
public java.util.ArrayList getCars()
```

Returns:

getTaxis

```
public java.util.ArrayList getTaxis()
```

Returns:

getTrucks

```
public java.util.ArrayList getTrucks()
```

Returns:

removeBus

```
public void removeBus(Bus bus)
```

Removes a bus from the bus linked list.

Parameters:

bus - Bus object to be removed from the linked list.

removeCar

```
public void removeCar(Car car)
```

Removes a car from car linked list.

Parameters:

car - Car object to be removed from the linked list.

removeTaxi

```
public void removeTaxi(Taxi taxi)
```

Removes a taxi from the taxi linked list.

Parameters:

taxi - Taxi to be removed from the linked list.

removeTruck

```
public void removeTruck(Truck truck)
```

Removes a truck from the truck linked list.

Parameters:

truck - Truck object to be removed from the linked list.

render

```
public void render(java.awt.Graphics g)
```

Goes through every position available on the linked lists and call the objects render() method.

Parameters:

g - Graphics object used to draw images on the game window.

tick

```
public void tick()
```

Goes through every position available on the linked links and tests the position of the object on the screen.

If it is already completely out of the screen, the respective object is removed, if it is not, its tick() method is called

Package score

Class Summary

[FileReader](#)

This class is responsible for loading the .txt file which contains the player's initials and scores.

[Score](#)

This class holds the arrays that store the name and the score of the five high scores of the game.

score

Class FileReader

```
java.lang.Object
|
+--score.FileReader
```

< [Constructors](#) > < [Methods](#) >

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

This class is responsible for loading the .txt file which contains the player's initials and scores.

Author:

Eder Paz ; Neil Blake ; Logan Wedel

Constructors

FileReader

```
public FileReader(java.lang.String txtName)
```

Loads and stores as a scanner the text file saved in the resource folder with the name specified by txtName.

Parameters:

txtName - Name of the file saved in the resource folder.

Methods

close

```
public void close()
```

Closes the scanner object, used in the end of the game when no reading and writing is necessary anymore.

getFile

```
public java.util.Scanner getFile()
```

Returns the scores file to be read and or written.

Returns:

File which stores the high scores of the game.

score

Class Score

```
java.lang.Object  
|  
+--score.Score
```

< [Fields](#) > < [Constructors](#) > < [Methods](#) >

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

This class holds the arrays that store the name and the score of the five high scores of the game.

Author:

Eder Paz ; Neil Blake ; Logan Wedel

Fields

initials

```
public static java.lang.String[] initials
```

score

```
public static int[] score
```

Constructors

Score

```
public Score()
```

Methods

close

```
public static void close()
```

Closes the scanner object used to read from the file.

init

```
public static void init()
```

Initiate the high scores of the game according to the data saved in the HighScores file in the resource folder.

Package states

Class Summary

[GameStates](#)

This is the class that summarizes all the information about the states of the game.

A "game state" is a single situation that the game might go to, like each menu, each phase of the game.

[Playing](#)

This is the class that all the actual game information, everything that happens on the screen when the game is running is defined here.

states

Class GameStates

```
java.lang.Object
|
+--states.GameStates
```

Direct Known Subclasses:

[Playing](#)

< [Constructors](#) > < [Methods](#) >

```
public abstract class GameStates
extends java.lang.Object
```

This is the class that summarizes all the information about the states of the game.

A "game state" is a single situation that the game might go to, like each menu, each phase of the game.

Author:

Eder Paz ; Neil Blake ; Logan Wedel

Constructors

GameStates

```
public GameStates(Game game)
```

Creates a new instance of the game to each game state so the the state can rely on the game variables.

Parameters:

game - Instance of the game so that the game state can rely on the game variables.

Methods

changeState

```
public static boolean changeState()
```

Getter for the boolean variable used to allow or not the game to change its state.

Returns:

Boolean variable.

getState

```
public static GameStates getState()
```

Getter for the current game state.

Returns:

Current running game state.

render

```
public abstract void render(java.awt.Graphics g)
```

Draw the game state on the screen.

Parameters:

g - Graphics object used to draw images on the screen.

setChangeState

```
public static void setChangeState(boolean b)
```

Setter for the boolean variable used to allow or not the game to change its state.

Parameters:

b - Boolean value in which the variable will be set to.

setState

```
public static void setState(GameStates state)
```

Sets the current state of the game.

Parameters:

state - State that the current game state has to be set to.

tick

```
public abstract void tick()
```

Upgrade the game state variables.

states

Class Playing

```
java.lang.Object
|
+--GameStates
|
+--states.Playing
```

< [Constructors](#) > < [Methods](#) >

```
public class Playing
extends GameStates
```

This is the class that all the actual game information, everything that happens on the screen when the game is running is defined here.

Author:

Eder Paz ; Neil Blake ; Logan Wedel

Constructors

Playing

```
public Playing(Game game)
```

Creates a new instance of all objects and instantiate the game object passed to it.

Parameters:

game - Game instance so that the game state can rely on the game variables.

Methods

render

```
public void render(java.awt.Graphics g)
```

Draws the player score and life on the screen.

Call the render() method of all objects in the game state.

Overrides:

[render](#) in class [GameStates](#)

tick

```
public void tick()
```

Generates random and automatically all the objects that cross the screen according to time.

Keeps track of and updates the player score.

Calls the tick() methods of all the object of the game state.

Overrides:

[tick](#) in class [GameStates](#)

INDEX

A

[addAlligator](#) ... 41
[addBus](#) ... 44
[addCar](#) ... 44
[addLog](#) ... 42
[addTaxi](#) ... 45
[addTruck](#) ... 45
[addTurtle](#) ... 42
[alli_height](#) ... 12
[alli_width](#) ... 12
[alliBank_height](#) ... 12
[alliBank_width](#) ... 12
[alligator](#) ... 29
[alligatorBank](#) ... 29
[Alligator](#) ... 7
[Alligator](#) ... 7
[AlligatorBank](#) ... 8
[AlligatorBank](#) ... 8
[Assets](#) ... 29
[Assets](#) ... 31

B

[bgnd](#) ... 30
[bus](#) ... 30
[Bus](#) ... 9
[Bus](#) ... 10

C

[car](#) ... 30
[car_height](#) ... 13
[car_width](#) ... 13
[changeState](#) ... 52
[close](#) ... 4
[close](#) ... 49
[close](#) ... 50
[createDisplay](#) ... 5
[crop](#) ... 33
[Car](#) ... 10
[Car](#) ... 11
[Collision](#) ... 2
[Collision](#) ... 2

D

[Death](#) ... 40
[Display](#) ... 4
[Display](#) ... 4
[Down](#) ... 35

E

[Entity](#) ... 12
[Entity](#) ... 14

F

[frog](#) ... 30
[frogAndAlligators](#) ... 2
[frogAndLogs](#) ... 3
[frogAndTurtles](#) ... 3
[frogAndVehicles](#) ... 3
[FileReader](#) ... 48
[FileReader](#) ... 48
[Frog](#) ... 16
[Frog](#) ... 16

G

[getAlligators](#) ... 42
[getBounds](#) ... 14
[getBuses](#) ... 45
[getCanvas](#) ... 5
[getCars](#) ... 45
[getDefaultSpeed](#) ... 25
[getFile](#) ... 49
[getFrame](#) ... 5
[getFrog](#) ... 41
[getGame](#) ... 14
[getHeight](#) ... 15
[getHeight](#) ... 25
[getKeyManager](#) ... 25
[getLogs](#) ... 42
[getMouseManager](#) ... 26
[getMouseX](#) ... 37
[getMouseY](#) ... 37
[getSpeed](#) ... 15
[getState](#) ... 52
[getTaxis](#) ... 45
[getTrucks](#) ... 45
[getTurtles](#) ... 42
[getWidht](#) ... 26
[getWidth](#) ... 15
[getX](#) ... 15
[getY](#) ... 15
[goToInitialPosition](#) ... 17
[goToPosition](#) ... 17
[Game](#) ... 24
[Game](#) ... 25
[GameStates](#) ... 51
[GameStates](#) ... 51

I

[init](#) ... 31
[init](#) ... 50
[initials](#) ... 49
[isLeftPressed](#) ... 37
[isRightPressed](#) ... 37
[isStopped](#) ... 17
[ImageLoader](#) ... 31
[ImageLoader](#) ... 31

K

[keyPressed](#) ... 35
[keyReleased](#) ... 36
[keyTyped](#) ... 36
[KeyManager](#) ... 34
[KeyManager](#) ... 34

L

[loadImage](#) ... 32
[log](#) ... 30
[log_height](#) ... 13
[Launcher](#) ... 27
[Launcher](#) ... 27
[Left](#) ... 35
[Log](#) ... 18
[Log](#) ... 18

M

[main](#) ... 28
[mouseClicked](#) ... 38
[mouseDragged](#) ... 38
[mouseEntered](#) ... 38
[mouseExited](#) ... 38
[mouseMoved](#) ... 38
[mousePressed](#) ... 38
[mouseReleased](#) ... 39
[MouseManager](#) ... 36
[MouseManager](#) ... 37

P

[player_height](#) ... 13
[player_width](#) ... 13
[Player](#) ... 40
[Player](#) ... 40
[Playing](#) ... 53
[Playing](#) ... 53

R

[removeAlligator](#) ... 43
[removeBus](#) ... 46
[removeCar](#) ... 46
[removeLog](#) ... 43
[removeTaxi](#) ... 46
[removeTruck](#) ... 46
[removeTurtle](#) ... 43
[render](#) ... 7
[render](#) ... 9
[render](#) ... 10
[render](#) ... 11
[render](#) ... 16
[render](#) ... 17
[render](#) ... 19
[render](#) ... 20
[render](#) ... 21
[render](#) ... 22
[render](#) ... 43
[render](#) ... 46
[render](#) ... 52
[render](#) ... 54
[run](#) ... 26
[Right](#) ... 35
[RiverItems](#) ... 41
[RiverItems](#) ... 41

S

[score](#) ... 49
[setChangeState](#) ... 52
[setDefaultSpeed](#) ... 26
[setState](#) ... 52
[setX](#) ... 17
[start](#) ... 26
[stop](#) ... 27
[Score](#) ... 49
[Score](#) ... 50
[SpriteSheet](#) ... 32
[SpriteSheet](#) ... 32

T

[taxi](#) ... 30
[tick](#) ... 8
[tick](#) ... 9
[tick](#) ... 10
[tick](#) ... 11
[tick](#) ... 16
[tick](#) ... 18
[tick](#) ... 19
[tick](#) ... 20
[tick](#) ... 21
[tick](#) ... 23
[tick](#) ... 36
[tick](#) ... 43
[tick](#) ... 47
[tick](#) ... 53
[tick](#) ... 54
[truck](#) ... 30
[truck_height](#) ... 13
[truck_width](#) ... 13
[turtle](#) ... 30
[turtle_height](#) ... 13
[turtle_width](#) ... 14
[Taxi](#) ... 19
[Taxi](#) ... 20
[Truck](#) ... 20
[Truck](#) ... 21
[Turtle](#) ... 22
[Turtle](#) ... 22

U

[Up](#) ... 35

V

[Vehicles](#) ... 44
[Vehicles](#) ... 44