Table Of Content

Package collisiondetection

Class Summary

Collision

This is the class that holds all the methods for collision detection using rectangles objects

collisiondetection

Class Collision

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

< Constructors > < Methods >

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

This is the class that holds all the methods for collision detection using rectangles objects

Author:

Eder Paz ; Neil Blake ; Logan Wedel

Constructors

Collision

```
public  Collision()
```

Methods

frogAndAlligatorBank

```
public boolean frogAndAlligatorBank(Frog frog,
                                    AlligatorBank alligator)
```

Checks if there is a collision between a frog and the alligator in the river banks.

Parameters:

frog - Current frog that the player is playing with
alligator - Alligator on the river bank

Returns:

True is there is a collision, false if there is not

frogAndAlligatorMouth

```
public boolean frogAndAlligatorMouth(Frog frog,
                                     java.util.ArrayList alligator)
```

Checks whether or not the frog lands on the alligators mouth and gets eaten.

Parameters:

frog - Current frog that the player is playing with
alligator - List of alligators of the game

Returns:

True is there is a collision, false if there is not

frogAndAlligators

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

Checks if there is a collision between a frog and the alligator's back.

Parameters:

frog - Current frog that the player is plying with
alligator - List of alligators of the game

Returns:

If there is a collision, return the index of the alligator that the frog is colliding with

frogAndLogs

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

Checks if there is a collision between a frog and the logs on the river.

Parameters:

frog - Current frog that the player is playing with
log - List of logs of the game

Returns:

If there is a collision, return the index of the log that the frog is colliding with

frogAndTurtles

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

Checks if there is a collision between a frog and the turtles on the river.

Parameters:

frog - Current frog that the player is playing with
turtle - List of turtles of the game

Returns:

If there is a collision, return the index of the turtles that the frog is colliding with

---

frogAndVehicles

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

Checks whether the frog collides with a vehicle.

Parameters:

frog - current frog the player is using
vehicles - List of all vehicles on the screen

Returns:

True if there is a collision, false if there is not

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

getHeadBounds

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

Getter for the rectangle object of the alligator head.

Returns:

Rectangle that represents the alligator head.

---

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

isInTheSurface

`public boolean` **`isInTheSurface`**`()`

Checks if the alligator is visible on the screen.

Returns:

True if the alligator is visible, false if it is not

---

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**()`

This method gets the speed of the entities in the game.

Returns:

Current speed of the entity.

---

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)
```

Takes the frog objects to the position define by the parameters x and y.

Parameters:

x - X position
y - Y position

---

isStopped

```
public boolean isStopped()
```

Test if the frog is in movement.

Returns:

True if the frog is moving, false if it is not

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 it movement defined by the tick() method.

Overrides:

render in class Entity

setX

`public void setX(float speed)`

Moves the frog according to the speed passed as a parameter.

Parameters:

speed - Speed in which the frog should move

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

FroggerLauncher

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

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.

---

game

Class FroggerLauncher

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

< Constructors > < Methods >

---

public class **FroggerLauncher**
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=PLWms45
- The you tube channel "New Beginner 2D Programming" - https://www.youtube.com/playlist?list=PLah6faXA

Author:

Eder Paz ; Neil Blake ; Logan Wedel

Version:

1.0 Created: 03/31/2016

Constructors

FroggerLauncher

```
public  FroggerLauncher()
```

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

---

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

GameOverState

```
public GameOver GameOverState()
```

Getter for the game over state.

Returns:

GameOverState as a GameOver State object

PlayingState

```
public Playing PlayingState()
```

Getter for the playing state

Returns:

PlayingState as a Playing State object

---

getDefaultSpeed

```
public float getDefaultSpeed()
```

Getter for the default speed of the game.

Returns:

Game default speed variable

---

getGameOverState

`public GameStates getGameOverState()`

Getter for the game over state.

Returns:

GameOverState as a GameStates object

---

getHeight

`public int getHeight()`

Getter for the game height.

Returns:

The height of the game window

---

getHighScoreState

`public GameStates getHighScoreState()`

Getter for the high score state.

Returns:

HighScoreState as a GameStates object

---

getKeyManager

`public KeyManager getKeyManager()`

Getter for the object that reads the keyboard inputs.

Returns:

KeyManager class instance.

---

getMenuState

`public GameStates getMenuState()`

Getter for the menu state.

Returns:

MenuState as a GameStates object

---

getMouseManager

`public MouseManager getMouseManager()`

Getter for the object that reads the mouse inputs.

Returns:

MouseManager class instance.

---

getPlayingState

`public GameStates getPlayingState()`

Getter for the playing state.

Returns:

PlayingState as a GameStates object

---

getWidht

`public int getWidht()`

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.

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.

MouseManager

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`**`(Game game)`

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

copies and instance of the game object so that it is possible to tests in with state the game is.

Parameters:

game - Instance of Game

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.

---

getInitials

`public java.lang.String **getInitials**()`

Getter for the player's initials

Returns:

Initials of the player as a string object

---

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.

Concatenates the key pressed to the string builder object.

---

resetInitials

`public void` **`resetInitials`**`()`

Resets the string builder object so that new initials can be typed when the game ends.

---

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

MouseManager

```
public  MouseManager()
```

Methods

getMouseX

```
public int getMouseX()
```

Getter for the mouse position on the screen.

Returns:

Mouse x position on the screen.

---

getMouseY

```
public int getMouseY()
```

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.

Not used in the game.

---

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

This is a class sets up the frogs array for the player in the game.

RiverItems

This class holds the arrays lists of all the river objects of the game.

Vehicles

This class holds the array lists of all the vehicles of the game.

---

objectsarrays

Class Player

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

< Constructors > < Methods >

---

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

This is a class sets up the frogs array for the player in the game.

Author:

Eder Paz ; Neil Blake ; Logan Wedel

Constructors

Player

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

Adds five frog objects to the player's list of frogs.

Parameters:

game - Game instance so that the player can rely on the game's variables

Methods

Death

`public void **Death**()`

Sets all the player frogs to the initial position.

---

getFrog

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

Getter for an specific frog in the frog list.

Parameters:

index - Index of the frog in the list to be returned

Returns:

Frog located on the index passed as a parameter

---

objectsarrays

Class RiverItems

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

< Constructors > < Methods >

---

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

This class holds the arrays lists of all the river 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.

---

clear

`public void` **`clear`**`()`

Remove all the existent river items objects from their respective lists.

---

getAlligators

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

Returns a list of alligators that are in the window.

Returns:

ArrayLists of alligators

---

getLogs

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

Returns a list of logs that are in the window.

Returns:

ArrayList of logs

---

getTurtles

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

Returns a list of turtles that are in the window.

Returns:

ArrayList of turtles

---

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 the 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 array 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 array lists and tests the position of the object on the screen.

If it is already completely out of the screen, the respective object is removed, is 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 array lists of all the vehicles 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.

---

clear

`public void **clear**()`

Remove all the existent vehicles from their respective lists.

---

getBuses

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

Returns a list of buses that are in the window.

Returns:

ArrayList of buses

---

getCars

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

Returns a list of cars that are in the window.

Returns:

ArrayList of cars

---

getTaxis

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

Returns a list of taxis that are in the window.

Returns:

ArrayList of taxis

---

getTrucks

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

Returns a list of trucks that are in the window.

Returns:

ArrayList of trucks

---

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 arrays 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 arrays lists and tests the position of the object on the screen.

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

Package score

Class Summary

FileHandler

This class is responsible for loading and writting in 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 FileHandler

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

< Constructors > < Methods >

---

public class **FileHandler**
extends java.lang.Object

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

Author:

Eder Paz ; Neil Blake ; Logan Wedel

Constructors

FileHandler

```
public  FileHandler(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.

---

write

```
public void write(java.lang.String[] initials,
                  int[] scores)
```

Rewrites the high scores files with the new high scores information.

Parameters:

initials - Initials of the player's high scores
scores - Scores of the player's high scores

---

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.

---

storeHighScores

```
public static void storeHighScores()
```

Rewrites a the high score file with the new high score data.

---

updateHighScores

```
public static void updateHighScores(java.lang.String newInitials,
                                    int newScore)
```

Test if the list of highest scores needs to be upgraded and if it does, upgrade it.

Package states

Class Summary

GameOver

This is the class that defines the game over screen.

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.

HighScores

This is the class that defines the screen that shows the highest scores of the game.

MainMenu

This is the class that defines the main menu 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 GameOver

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

< Constructors > < Methods >

---

public class **GameOver**
extends GameStates

This is the class that defines the game over screen.

Author:

Eder Paz ; Logan Wedel ; Neil Blake

Constructors

GameOver

`public` **`GameOver`**`(Game game)`

Makes a copy of the game object so that the state can rely on the game variables.

Parameters:

game - Game instance

Methods

checkScore

`public void` **`checkScore`**`(int score)`

Check if the score sent to it is within the high score values recorded in the high scores file.

---

render

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

Draws the the menus and the player's initials that are being typed, if the player reached a high score.

Overrides:

render in class GameStates

---

tick

`public void` **`tick`**`()`

Evaluates the position where the mouse was clicked on the game window and change the state of the game according to it.

Overrides:

tick in class GameStates

states

Class GameStates

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

Direct Known Subclasses:
GameOver, HighScores, MainMenu, 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

backToLastState

`public static void` **`backToLastState`**`()`

This brings the player to the screen that was the last state of the player in the game.

---

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.

---

setGameStateTo

`public static void setGameStateTo(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 HighScores

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

< Constructors > < Methods >

---

public class **HighScores**
extends GameStates

This is the class that defines the screen that shows the highest scores of the game.

Author:

Eder Paz ; Neil Blake ; Logan Wedel

Constructors

HighScores

`public HighScores(Game game)`

Makes a copy of the game objects so that the state can rely on the game variables.

Parameters:

game - Game instance.

Methods

render

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

Draw the list of the highest scores with initials and scores.

Draw the buttons and titles as well.

Overrides:

render in class GameStates

tick

`public void tick()`

Evaluates the position where the mouse was clicked on the game window and change the state of the game according to it.

Overrides:

tick in class GameStates

---

states

Class MainMenu

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

< Constructors > < Methods >

---

public class **MainMenu**
extends GameStates

This is the class that defines the main menu of the game.

Author:

Eder Paz ; Neil Blake ; Logan Wedel

Constructors

MainMenu

`public MainMenu(Game game)`

Makes a copy of the game object so that the state can rely on the game variables.

Parameters:

game - Game instance

Methods

render

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

Draw the buttons on the screen.

Overrides:

render in class GameStates

---

tick

`public void tick()`

Evaluates the position where the mouse was clicked on the game window and change the state of the game according to it.

Overrides:

tick in class GameStates

---

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.

Initiate all the objects needed or the game.

Parameters:

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

Methods

getScore

`public int` **`getScore`**`()`

Getter for the player score.

Returns:

Player score value.

---

levelBegin

`public void` **`levelBegin`**`()`

Starts the game.

Initialize the variables to record the player score correctly.

Initialize the timer, life and score variables.

Set the frogs to their initial position.

Clear all the objects that have already been created.

---

render

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

Draws the player score and life on the screen as well as the highest score already achieved and the timer bar.

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

Overrides:

render in class GameStates

---

tick

```
public void tick()
```

Keeps track of everything that can happen in the game.

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

Overrides:

tick in class GameStates

INDEX

H

I

K

L

M

P

R

S

T

U

V

W