

Table Of Content

PhysicsActions	2
PhysicsActions.accelerateBus	2
PhysicsActions.accelerateObject	3
PhysicsActions.loadNewFrame	4
PhysicsActions.moveBus	5
PhysicsActions.updateObjectPosition	5
PhysicsEngine	6
PhysicsFormulas	7
PhysicsFormulas.TooManyNullArgumentsException	9
EditObjectUI	10
MainWindow	11
ObjectUI	12
BusObject	13
ObjectManager	14
PhysicsObject	16
RelationshipManager	21
RelationshipTypes	22
RoundObject	23
SquareObject	24
Index	27

Class PhysicsActions

```
java.lang.Object
|
+--edu.vu.vuse.cs278.g3.engine.PhysicsActions
```

< [Constructors](#) >

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

Class that contains inner classes that implement the Runnable interface. All inner classes should implement the Runnable interface and be designed as one specific action for the PhysicsEngine work queue.

Author:

Matthew Shea

Constructors

PhysicsActions

```
public PhysicsActions()
```

Class PhysicsActions.accelerateBus

```
java.lang.Object
|
+--edu.vu.vuse.cs278.g3.engine.PhysicsActions.accelerateBus
```

All Implemented Interfaces:

java.lang.Runnable

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

```
public static class PhysicsActions.accelerateBus
extends java.lang.Object
implements java.lang.Runnable
```

Calculates the new speed for the bus based on the current speed and the acceleration.

Author:

Matthew Shea

Constructors

PhysicsActions.accelerateBus

```
public PhysicsActions.accelerateBus()
```

Methods

run

```
public void run()
```

Class PhysicsActions.accelerateObject

```
java.lang.Object
|
+--edu.vu.vuse.cs278.g3.engine.PhysicsActions.accelerateObject
```

All Implemented Interfaces:

java.lang.Runnable

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

```
public static class PhysicsActions.accelerateObject
extends java.lang.Object
implements java.lang.Runnable
```

Accelerates the object based on the calculated friction and speed of the object

Author:

Matthew Shea

Constructors

PhysicsActions.accelerateObject

```
public PhysicsActions.accelerateObject()
```

Methods

run

```
public void run()
```

Class PhysicsActions.loadNewFrame

```
java.lang.Object
|
+--edu.vu.vuse.cs278.g3.engine.PhysicsActions.loadNewFrame
```

All Implemented Interfaces:

java.lang.Runnable

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

```
public static class PhysicsActions.loadNewFrame
extends java.lang.Object
implements java.lang.Runnable
```

This Runnable adds the sequence of other runnables to the physics engine and then itself at the end. This is added to the PhysicsEngine's work queue on the enable() call.

Author:

Matthew Shea

Constructors

PhysicsActions.loadNewFrame

```
public PhysicsActions.loadNewFrame()
```

Methods

run

```
public void run()
```

Class PhysicsActions.moveBus

```
java.lang.Object
|
+--edu.vu.vuse.cs278.g3.engine.PhysicsActions.moveBus
```

All Implemented Interfaces:

java.lang.Runnable

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

```
public static class PhysicsActions.moveBus
extends java.lang.Object
implements java.lang.Runnable
```

The bus itself should never move on the screen. We will simulate movement by adjusting the position of the background.

Author:

Matthew Shea

Constructors

PhysicsActions.moveBus

```
public PhysicsActions.moveBus()
```

Methods

run

```
public void run()
```

Class PhysicsActions.updateObjectPosition

```
java.lang.Object
|
+--edu.vu.vuse.cs278.g3.engine.PhysicsActions.updateObjectPosition
```

All Implemented Interfaces:

java.lang.Runnable

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

```
public static class PhysicsActions.updateObjectPosition
```

extends java.lang.Object
implements java.lang.Runnable

Updates the object's position on the netlogo display

Author:

Matthew Shea

Constructors

PhysicsActions.updateObjectPosition

```
public PhysicsActions.updateObjectPosition()
```

Methods

run

```
public void run()
```

Class PhysicsEngine

```
java.lang.Object  
|  
+--edu.vu.vuse.cs278.g3.engine.PhysicsEngine
```

< [Methods](#) >

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

This class is a singleton wrapper around a QueueExecutor that runs the physics calculations and updates the display. Most of the work will be done on this QueueExecutor.

Author:

syddraf

Methods

addtoQueue

```
public void addToQueue(java.lang.Runnable runnable)
```

Adds an item to the queue for the thread to execute.

Parameters:

runnable - PhysicsAction for the engine to perform

disable

```
public void disable()
```

Calling this method will allow the engine to finish it's current operation, and then suspend and flush the queue.

enable

```
public void enable()
```

This method will allow the engine to perform operations in its run queue.

getInstance

```
public static PhysicsEngine getInstance()
```

This method returns the singleton PhysicsEngine object.

Returns:

The singleton PhysicsEngine

Class PhysicsFormulas

```
java.lang.Object
|
+--edu.vu.vuse.cs278.g3.engine.PhysicsFormulas
```

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

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

Constructors

PhysicsFormulas

```
public PhysicsFormulas()
```

Methods

frictionalForce

```
public static java.lang.Double frictionalForce(java.lang.Double f,  
                                              java.lang.Double m,  
                                              java.lang.Double u)  
                                              throws
```

[PhysicsFormulas.TooManyNullArgumentsException](#)

Implements the formula $F = um$; Pass one parameter as null and the calculated value of that parameter will be returned.

Parameters:

- f - The frictional force applied to the object
- m - The mass of the object
- u - The coefficient of friction

Returns:

The returned value is the calculated value of the parameter that was passed as null.

Throws:

`edu.vu.vuse.cs278.g3.engine.PhysicsFormulas.TooManyNullArgumentsException` -
Thrown if more than one argument is null.

momentum

```
public static java.lang.Double momentum(java.lang.Double P,  
                                       java.lang.Double m,  
                                       java.lang.Double v)  
                                       throws
```

[PhysicsFormulas.TooManyNullArgumentsException](#)

This function implements the formula for an objects momentum.

Parameters:

- P - The momentum of the object.
- m - The mass of the object.
- v - The velocity of the object.

Returns:

The value returned is the argument that was set to null in the arguments.

Throws:

`edu.vu.vuse.cs278.g3.engine.PhysicsFormulas.TooManyNullArgumentsException` -
Thrown if more than one argument is null.

Class

PhysicsFormulas.TooManyNullArgumentsException

```
java.lang.Object
|
+-- java.lang.Throwable
|
+-- java.lang.Exception
|
+-- edu.vu.vuse.cs278.g3.engine.PhysicsFormulas.TooManyNullArgumentsException
```

All Implemented Interfaces:

java.io.Serializable

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

```
public static class PhysicsFormulas.TooManyNullArgumentsException
extends java.lang.Exception
```

This exception indicates that too many null arguments were passed to one of the above PhysicsFormulas.

Author:

Matthew Shea

Constructors

PhysicsFormulas.TooManyNullArgumentsException

```
public PhysicsFormulas.TooManyNullArgumentsException(int number)
```

Methods

getError

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

Class EditObjectUI

```
java.lang.Object
|
+-- java.awt.Component
|   |
|   +-- java.awt.Container
|       |
|       +-- java.awt.Window
|           |
|           +-- java.awt.Frame
|               |
|               +-- javax.swing.JFrame
|                   |
|                   +-- edu.vu.vuse.cs278.g3.gui.EditObjectUI
```

All Implemented Interfaces:

java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable,
javax.accessibility.Accessible, javax.swing.RootPaneContainer,
javax.swing.TransferHandler.HasGetTransferHandler, javax.swing.WindowConstants

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

```
public class EditObjectUI
extends javax.swing.JFrame
```

This class holds the code for the Edit Object Dialog box.

Author:

Amber Maria

Constructors

EditObjectUI

```
public EditObjectUI()
```

The constructor for the class. Since this is Edit Object, we want to "fill in" the fields for the user to see what the current attributes are.

Methods

main

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

Parameters:

args - the command line arguments

Class MainWindow

```
java.lang.Object
|
+-- java.awt.Component
|   |
|   +-- java.awt.Container
|       |
|       +-- java.awt.Window
|           |
|           +-- java.awt.Frame
|               |
|               +-- javax.swing.JFrame
|                   |
|                   +-- edu.vu.vuse.cs278.g3.gui.MainWindow
```

All Implemented Interfaces:

java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable,
javax.accessibility.Accessible, javax.swing.RootPaneContainer,
javax.swing.TransferHandler.HasGetTransferHandler, javax.swing.WindowConstants

< [Methods](#) >

```
public class MainWindow
extends javax.swing.JFrame
```

This class handles the main window of the program where the NetLogo is embedded.

Author:

Amber Maria

Methods

command

```
public void command(java.lang.String arg)
```

Sends arg to embedded NetLogo

Parameters:

arg -

getInstance

```
public static MainWindow getInstance()
```

Gets the single instance of the Main Window

Returns:

instance of Main Window

Class ObjectUI

```
java.lang.Object
|
+-- java.awt.Component
|   |
|   +-- java.awt.Container
|       |
|       +-- java.awt.Window
|           |
|           +-- java.awt.Frame
|               |
|               +-- javax.swing.JFrame
|                   |
|                   +-- edu.vu.vuse.cs278.g3.gui.ObjectUI
```

All Implemented Interfaces:

java.awt.MenuContainer, java.awt.image.ImageObserver, java.io.Serializable,
javax.accessibility.Accessible, javax.swing.RootPaneContainer,
javax.swing.TransferHandler.HasGetTransferHandler, javax.swing.WindowConstants

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

```
public class ObjectUI
extends javax.swing.JFrame
```

Class that handles the Add Object dialog

Author:

Amber

Constructors

ObjectUI

```
public ObjectUI()
```

Constructor. Calls initComponents to declare and initialize all fields of the dialog and sets the sliders to a neutral value.

Methods

main

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

Parameters:

args - the command line arguments

Class BusObject

```
java.lang.Object
|
+--PhysicsObject
|
+--edu.vu.vuse.cs278.g3.model.BusObject
```

< [Methods](#) >

```
public class BusObject
extends PhysicsObject
```

Methods

commit

```
public void commit()
```

Overrides:

[commit](#) in class [PhysicsObject](#)

getLength

```
public double getLength()
```

setShape

```
public void setShape(java.lang.String shape)
```

Overrides:

[setShape](#) in class [PhysicsObject](#)

updatePosition

```
public void updatePosition(double xcoord,
                           double ycoord)
```

Overrides:

[updatePosition](#) in class [PhysicsObject](#)

Class ObjectManager

```
java.lang.Object
|
+--edu.vu.vuse.cs278.g3.model.ObjectManager
```

< [Methods](#) >

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

The ObjectManager serves to manage all objects involved in the model. Its two primary functions are maintaining a map of all objects involved in the model and facilitating the creation of the objects.

Author:

Matthew Shea

Methods

addObject

```
public boolean addObject(java.lang.String id,
                        PhysicsObject object)
```

Adds an object to the manager with the specified id.

Parameters:

id - the UNIQUE identifier of the object
object - The object reference

Returns:

True if the object was successfully added and False if it already exists.

createBus

```
public BusObject createBus(double _xCoord,
                           double _yCoord,
                           double _length,
                           double mass)
```

Creates a BusObject with the specified parameters and returns the reference.

Parameters:

_xCoord - The x position of the bus
_yCoord - The y position of the bus
_length - The length of the bus

Returns:

Object reference

createCircle

```
public RoundObject createCircle(double _xCoord,  
                                   double _yCoord,  
                                   double _radius,  
                                   double mass)
```

Creates a RoundObject with the specified parameters and returns the reference.

Parameters:

_xCoord - The x position of the circle
_yCoord - The y position of the circle
_radius - The radius of the circle

Returns:

Object reference

createSquare

```
public SquareObject createSquare(double _xCoord,  
                                   double _yCoord,  
                                   double _width,  
                                   double _height,  
                                   double mass)
```

Creates a SquareObject with the specified parameters and returns the reference.

Parameters:

_xCoord - The x position of the square
_yCoord - The y position of the square
_width - The width of the object
_height - The height of the object

Returns:

Object reference

getInstance

```
public static ObjectManager getInstance()
```

Returns the singleton instance of the ObjectManager.

Returns:

Singleton instance of the ObjectManager

getObject

```
public PhysicsObject getObject(java.lang.String id)
```

Returns the reference to the object associated with the id

Parameters:

id - identifier of the object

Returns:

The reference to the object or null if the object is not found.

removeObject

```
public void removeObject(java.lang.String id)
```

Remove the object associated with the specified unique id.

Parameters:

id - Unique ID of the object to remove.

updateObject

```
public void updateObject(java.lang.String id,  
                          PhysicsObject obj)
```

Replaces the specified key with the new object.

Parameters:

id - Identifier

obj - New Object

Class PhysicsObject

```
java.lang.Object  
|  
+--edu.vu.vuse.cs278.g3.model.PhysicsObject
```

Direct Known Subclasses:

[BusObject](#), [RoundObject](#), [SquareObject](#)

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

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


Fields

acceleration

protected double **acceleration**

The acceleration of the object, measured in pixels per frame per frame

array_num

protected int **array_num**

The number in the array of the Netlogo object

mass

protected double **mass**

The mass of the object in kg

speed

protected double **speed**

The speed of the object, measured in pixels per frame

xCoord

protected double **xCoord**

The x coordinate on the Netlogo display of the object

yCoord

protected double **yCoord**

The y coordinate on the Netlogo display of the object

Constructors

PhysicsObject

```
protected PhysicsObject(int arrayNum,  
                        double xcoord,  
                        double ycoord,  
                        double speed,  
                        double acceleration,  
                        double mass)
```

Creates a PhysicsObject with the specified parameters.

Parameters:

arrayNum - The number in the Netlogo array
xcoord - The xcoord in the Netlogo display
ycoord - The ycoord in the Netlogo display
speed - The speed of the object. Positive indicates rightward motion.
acceleration - The acceleration of the object.
mass - The mass of the object.

Methods

commit

```
public abstract void commit()
```

Commits the changes to the NetLogo backend to update the graphical display. You do not need to call this function unless the POSITION has changed.

getAcceleration

```
public double getAcceleration()
```

Returns the acceleration of the object

Returns:

The object's acceleration

getMass

```
public double getMass()
```

Returns the mass of the object

Returns:

The object's mass

getSpeed

```
public double getSpeed()
```

Returns the speed of the object

Returns:

The object's speed

getXCoord

```
public double getXCoord()
```

Returns the x coordinate of the object

Returns:

The object's X coordinate

getYCoord

```
public double getYCoord()
```

Returns the y coordinate of the object

Returns:

The object's Y coordinate

setAcceleration

```
public void setAcceleration(double acc)
```

Set's the object's acceleration

Parameters:

acc - A double indicating the acceleration of the object.

setMass

```
public void setMass(double mass)
```

Sets the mass of the object

Parameters:

mass - The new mass of the object

setShape

```
public abstract void setShape(java.lang.String shape)
```

Immediately changes the object's shape. Must exist in netlogo shapes library.

Parameters:

shape -

setSpeed

```
public void setSpeed(double iSpeed)
```

Sets the speed of the object.

Parameters:

iSpeed - The speed in pixels/frame

setXCoord

```
public void setXCoord(double xcoord)
```

Sets the x coordinate of the object

Parameters:

xcoord - x coordinate in pixels

setYCoord

```
public void setYCoord(double ycoord)
```

Sets the y coordinate of the object

Parameters:

ycoord - y coordinate in pixels

updatePosition

```
public void updatePosition(double xcoord,  
                           double ycoord)
```

Updates the position with the specified xcoords and ycoords

Parameters:

xcoord -

ycoord -

Class RelationshipManager

```
java.lang.Object
|
+--edu.vu.vuse.cs278.g3.model.RelationshipManager
```

< [Methods](#) >

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

Singleton object to manage the list of Relationships

Author:

Matthew Shea

Methods

getInstance

```
public static RelationshipManager getInstance()
```

Returns the singleton instance of this type

Returns:

Singleton instance

getRelationship

```
public RelationshipTypes getRelationship(java.lang.String object1,
                                         java.lang.String object2)
```

This function fetches a relationship between two objects. The order of the objects is important. For example, getRelationship("a", "b") is not the same as getRelationship("b", "a")

Parameters:

object1 - The string id of the first object

object2 - The string id of the second object

Returns:

The relationshipType, if available, or NO_RELATIONSHIP if not.

setRelationship

```
public void setRelationship(java.lang.String object1,  
                           java.lang.String object2,  
                           RelationshipTypes type)
```

This function sets a relationship between two objects. The order of the objects is important. For example, `getRelationship("a", "b")` is not the same as `getRelationship("b", "a")`

Parameters:

object1 - The string id of the first object
object2 - The string id of the second object
type - The RelationshipType to define

Class RelationshipTypes

```
java.lang.Object  
|  
+-- java.lang.Enum  
|  
+-- edu.vu.vuse.cs278.g3.model.RelationshipTypes
```

All Implemented Interfaces:

java.io.Serializable, java.lang.Comparable

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

```
public final class RelationshipTypes  
extends java.lang.Enum
```

Enumeration that defines possible relationship types

Author:

Matthew Shea

Fields

ABOVE_RESTRAINED

```
public static final RelationshipTypes ABOVE_RESTRAINED
```

ABOVE_UNRESTRAINED

```
public static final RelationshipTypes ABOVE_UNRESTRAINED
```

BEHIND_ATTACHED

```
public static final RelationshipTypes BEHIND_ATTACHED
```

INSIDE_RESTRAINED

```
public static final RelationshipTypes INSIDE_RESTRAINED
```

INSIDE_UNRESTRAINED

```
public static final RelationshipTypes INSIDE_UNRESTRAINED
```

NO_RELATIONSHIP

```
public static final RelationshipTypes NO_RELATIONSHIP
```

Methods

valueOf

```
public static RelationshipTypes valueOf(java.lang.String name)
```

values

```
public static edu.vu.vuse.cs278.g3.model.RelationshipTypes[] values()
```

Class RoundObject

```
java.lang.Object
|
+--PhysicsObject
    |
    +--edu.vu.vuse.cs278.g3.model.RoundObject
```

< [Methods](#) >

```
public class RoundObject
extends PhysicsObject
```

Methods

commit

```
public void commit()
```

Overrides:

[commit](#) in class [PhysicsObject](#)

getRadius

```
public double getRadius()
```

setRadius

```
public void setRadius(double radius)
```

setShape

```
public void setShape(java.lang.String shape)
```

Overrides:

[setShape](#) in class [PhysicsObject](#)

updatePosition

```
public void updatePosition(double xcoord,  
                           double ycoord)
```

Overrides:

[updatePosition](#) in class [PhysicsObject](#)

Class SquareObject

```
java.lang.Object  
|  
+--PhysicsObject  
    |  
    +--edu.vu.vuse.cs278.g3.model.SquareObject
```

< [Methods](#) >

```
public class SquareObject  
extends PhysicsObject
```

Methods

commit

```
public void commit()
```

Overrides:

[commit](#) in class [PhysicsObject](#)

getHeight

```
public double getHeight()
```

getWidth

```
public double getWidth()
```

getXCoord

```
public double getXCoord()
```

Overrides:

[getXCoord](#) in class [PhysicsObject](#)

getYCoord

```
public double getYCoord()
```

Overrides:

[getYCoord](#) in class [PhysicsObject](#)

setHeight

```
public void setHeight(double height)
```

setShape

```
public void setShape(java.lang.String shape)
```

Overrides:

[setShape](#) in class [PhysicsObject](#)

setWidth

```
public void setWidth(double width)
```

updatePosition

```
public void updatePosition(double xcoord,  
                           double ycoord)
```

Overrides:

[updatePosition](#) in class [PhysicsObject](#)

INDEX

A

[acceleration](#) ... 17
[addObject](#) ... 14
[addtoQueue](#) ... 7
[array_num](#) ... 17
[ABOVE RESTRAINED](#) ... 22
[ABOVE UNRESTRAINED](#) ... 22

B

[BEHIND ATTACHED](#) ... 22
[BusObject](#) ... 13

C

[command](#) ... 11
[commit](#) ... 13
[commit](#) ... 18
[commit](#) ... 24
[commit](#) ... 25
[createBus](#) ... 14
[createCircle](#) ... 15
[createSquare](#) ... 15

D

[disable](#) ... 7

E

[enable](#) ... 7
[EditObjectUI](#) ... 10
[EditObjectUI](#) ... 10

F

[frictionalForce](#) ... 8

G

[getAcceleration](#) ... 18
[getError](#) ... 9
[getHeight](#) ... 25
[getInstance](#) ... 7
[getInstance](#) ... 11
[getInstance](#) ... 15
[getInstance](#) ... 21
[getLength](#) ... 13
[getMass](#) ... 18
[getObject](#) ... 16
[getRadius](#) ... 24
[getRelationship](#) ... 21
[getSpeed](#) ... 19
[getWidth](#) ... 25
[getXCoord](#) ... 19
[getXCoord](#) ... 25
[getYCoord](#) ... 19
[getYCoord](#) ... 25

I

[INSIDE RESTRAINED](#) ... 23
[INSIDE UNRESTRAINED](#) ... 23

M

[main](#) ... 10
[main](#) ... 12
[mass](#) ... 17
[momentum](#) ... 8
[MainWindow](#) ... 11

N

[NO RELATIONSHIP](#) ... 23

O

[ObjectManager](#) ... 14
[ObjectUI](#) ... 12
[ObjectUI](#) ... 12

P

[PhysicsActions](#) ... 2
[PhysicsActions](#) ... 2
[PhysicsActions.accelerateBus](#) ... 2
[PhysicsActions.accelerateBus](#) ... 3
[PhysicsActions.accelerateObject](#) ... 3
[PhysicsActions.accelerateObject](#) ... 3
[PhysicsActions.loadNewFrame](#) ... 4
[PhysicsActions.loadNewFrame](#) ... 4
[PhysicsActions.moveBus](#) ... 5
[PhysicsActions.moveBus](#) ... 5
[PhysicsActions.updateObjectPosition](#) ... 5
[PhysicsActions.updateObjectPosition](#) ... 6
[PhysicsEngine](#) ... 6
[PhysicsFormulas](#) ... 7
[PhysicsFormulas](#) ... 8
[PhysicsFormulas.TooManyNullArgumentsException](#) ... 9
[PhysicsFormulas.TooManyNullArgumentsException](#) ... 9
[PhysicsObject](#) ... 16
[PhysicsObject](#) ... 18

R

[removeObject](#) ... 16
[run](#) ... 3
[run](#) ... 4
[run](#) ... 4
[run](#) ... 5
[run](#) ... 6
[RelationshipManager](#) ... 21
[RelationshipTypes](#) ... 22
[RoundObject](#) ... 23

S

[setAcceleration](#) ... 19
[setHeight](#) ... 26
[setMass](#) ... 19
[setRadius](#) ... 24
[setRelationship](#) ... 22
[setShape](#) ... 13
[setShape](#) ... 20
[setShape](#) ... 24
[setShape](#) ... 26
[setSpeed](#) ... 20
[setWidth](#) ... 26
[setXCoord](#) ... 20
[setYCoord](#) ... 20
[speed](#) ... 17
[SquareObject](#) ... 24

U

[updateObject](#) ... 16
[updatePosition](#) ... 13
[updatePosition](#) ... 20
[updatePosition](#) ... 24
[updatePosition](#) ... 26

V

[valueOf](#) ... 23
[values](#) ... 23

X

[xCoord](#) ... 17

Y

[yCoord](#) ... 17