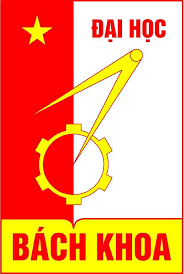
**HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY**

**

**REPORT**

MINI PROJECT

*for*

OBJECT-ORIENTED LANGUAGE & THEORY

TOPIC 2

**SUPERVISOR: DR. NGUYEN THI THU TRANG**

**NGUYEN MANH PHUC - 20176845**

**NGUYEN THI HONG ANH – 20176679**

**ICT04 – K62**

*Ha Noi, June 8th 2020*

Contents

[**DESCRIPTION** 3](#_Toc42724921)

[**Mini project requirements** 3](#_Toc42724922)

[Problem statement 3](#_Toc42724923)

[First law of motion 3](#_Toc42724924)

[Second law of motion 4](#_Toc42724925)

[Third law of motion 4](#_Toc42724926)

[**Use case diagram** 5](#_Toc42724927)

[**DESIGN IDEAS** 7](#_Toc42724928)

[**General class diagram** 7](#_Toc42724929)

[**Class diagram of package** 8](#_Toc42724930)

[Main 8](#_Toc42724931)

[Menu 8](#_Toc42724932)

[Introduction 10](#_Toc42724933)

[Firstlaw 11](#_Toc42724934)

[Secondlaw 14](#_Toc42724935)

[Thirdlaw 16](#_Toc42724936)

[Functional interfaces 18](#_Toc42724937)

[**ASSIGNMENT OF MEMBERS** 20](#_Toc42724938)

[**REFERENCE:** 21](#_Toc42724939)

# **DESCRIPTION**

## **Mini project requirements**

### Problem statement

Ever since its publification, ***Philosophiæ Naturalis Principia Mathematica*** (also called Principia) has proven itself to be one of the most important work in the history of science. The Principia states Newton’s laws of motion and sets the foundation for classical physics. The three laws describe the relationship between a body and the forces acting up on it, and its response according to those forces.

The first law of motion:

* In an inertial frame of reference, an object either remains at rest or continuously moves at a constant velocity, unless acted upon by a force.

The second law of motion:

* In an inertial frame of reference, the vector sum of forces F on an object is the product of mass m and acceleration a of that object.

The third law of motion

* When a body exerts force on a second body, that second body simultaneously exerts a force equal in magnitude and opposite in direction on the first body.

In this mini project, a visualization of Newton’s three laws of motion will be implemented using a ball’s motion

### First law of motion

In this first law, to represent the idea, the ball will be placed at rest unless user press button for it to move. It will move right or left depending on the user’s choice:

* When the user clicks *PushToTheRight* button, the ball will be pushed to the right and roll forever with a force level of 1. The ball will roll faster if user continue to click *PushToTheRight*  button and the force level will be increased by 1 (max is 10) after every click.
* Similarly, When the user clicks *PushToTheLeft* button, the ball will be pushed to the left and roll forever with a force level of 1. The ball will roll faster if user continue to click *PushToTheLeft*  button and the force level will be increased by 1 (max is 10) after every click.
* If the ball is rolling to the left or right with a force level *n* and user clicks *PushToTheRight*/*PushToTheLeft* button, which is the opposite direction of the rolling ball, the ball will either stop if n = 0 or roll slower with n = n – 1;
* The ball will be reset to initial position if user clicks *Reset* button

### Second law of motion

For this law, the ball will be under the influence of 3 forces:

* Gravitational force
* Normal force
* Frictional force

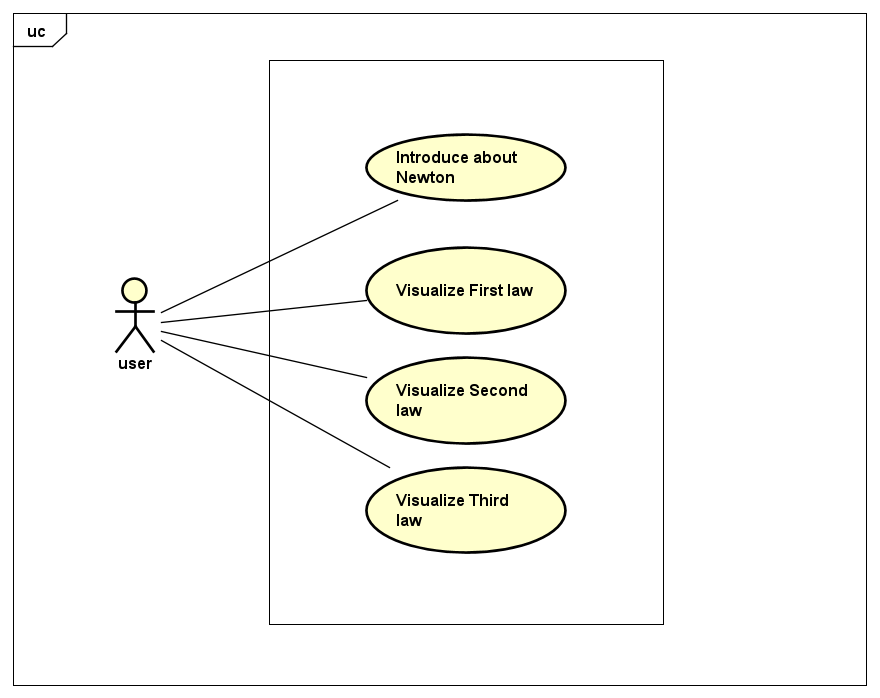
Since the ball is placed on ground, gravitational force is apparent. Another force called normal force also influence the ball. Since these two forces are equal in magnitude and opposite in direction, making the ball at its equilibrium, the force responsible for the ball’s motion is frictional force.

Assume a virtual initial force acts upon the ball to make it move. With a frictional force chosen by the user, the ball will move with a negative acceleration, and will eventually stop at a certain point at a certain time. Without the frictional force, it will move forever.

### Third law of motion

In this law, the ball is placed at a distance away from a wall. When it starts rolling toward the wall, the moment the ball touches the wall, the wall will exert a force of equivalent magnitude but opposite direction on the ball, making it bounce back to its original position.

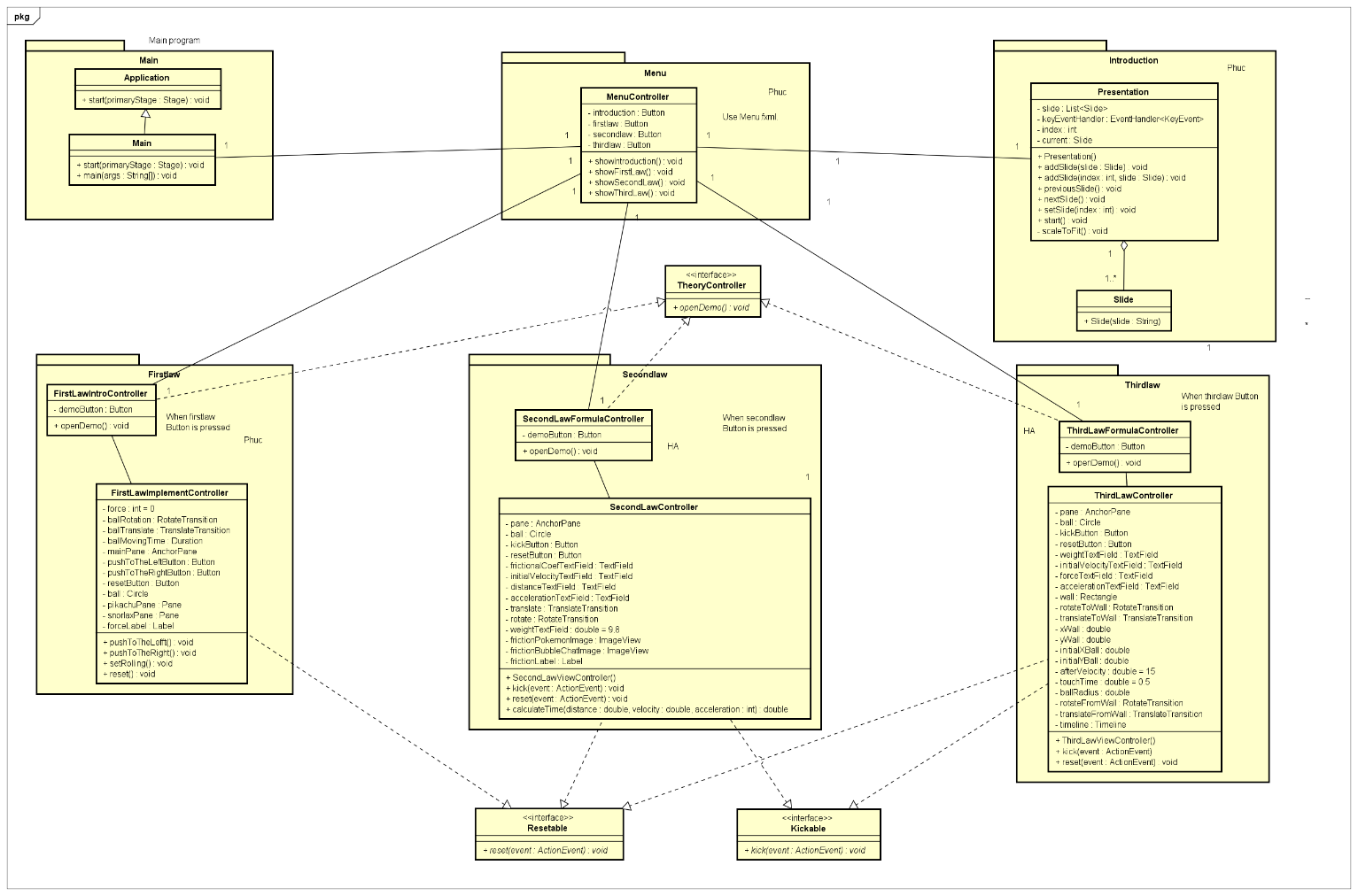
## **Use case diagram**



|  |  |
| --- | --- |
| USE CASE | DESCRIPTION |
| Introduce about Newton | Opens a new window of a brief presentation of Newton’s life since birth and his most famous work. The presentation will be a series of slides with image to show to user |
| Visualize First Law | Opens a new window to display visualization of first law. First window will be a summary of theory and then lead user to a demonstration of the first law. The user will click buttons to see how the law is illustrated |
| Visualize Second Law | Opens new windows to display visualization of second law. First window will be a summary of theory and then lead user to a demonstration of the second law. The user is prompt to enter some inputs for calculation, when the fields needed are filled and ball is chosen to be kick, the demonstration starts |
| Visualize Third Law | Opens new windows to display visualization of third law. First window will be a summary of theory and then lead user to a demonstration of the third law. The user is prompt to enter some inputs for calculation, when the fields needed are filled and ball is chosen to be kick, the demonstration starts |

# **DESIGN IDEAS**

## **General class diagram**



The overall design is simple. Classes are divided into equivalent packages with other related classes.

The four packages are:

* Main: responsible for the entire operation of the program. Contains the main function.
* Menu: Contains the menu for the program.
* Introduction: Package includes classes related to the presentation of Newton’s life.
* Firstlaw: Package includes classes related to the first law of motion.
* Secondlaw: Package includes classes related to the second law of motion.
* Thirdlaw: Package includes classes related to the third law of motion.
* Interfaces: These interfaces are functional interfaces, to provide polymorphism to the program.

## **Class diagram of package**

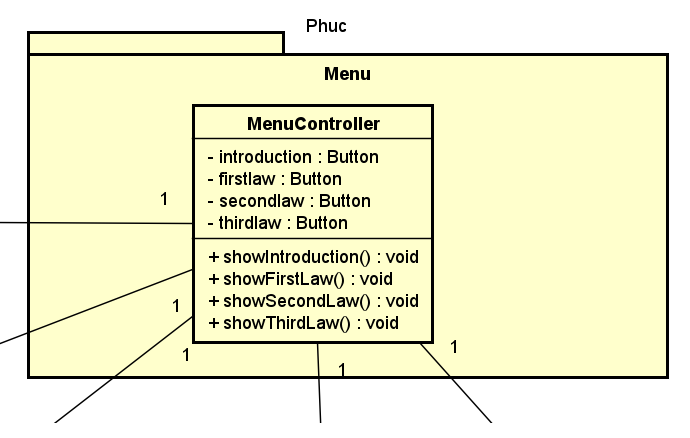
### Main

A screenshot of a cell phone

Description automatically generated

This package contains only one class that is the Main class. This class extends from the abstract class Application and inherits its start method. When overriding start, Main will add necessary components to get the program running

### Menu



This package includes the *MenuController* class, which is responsible for the Menu representation of the program. There are four buttons with four corresponding methods for opening equivalent windows when a certain button is pressed.

Detail description of the flow:

* If user press *Introduction* Button, this component will call *showIntroduction* method to render equivalent FXML file and redirect user to introduction window.
* If user press *first law* Button, this component will call *showFirstLaw* method to render equivalent FXML file and redirect user to first law window.
* If user press *second law* Button, this component will call *showSecondLaw* method to render equivalent FXML file and redirect user to second law window.
* If user press *third law* Button, this component will call *showThirdLaw* method to render equivalent FXML file and redirect user to third law window.

### Introduction

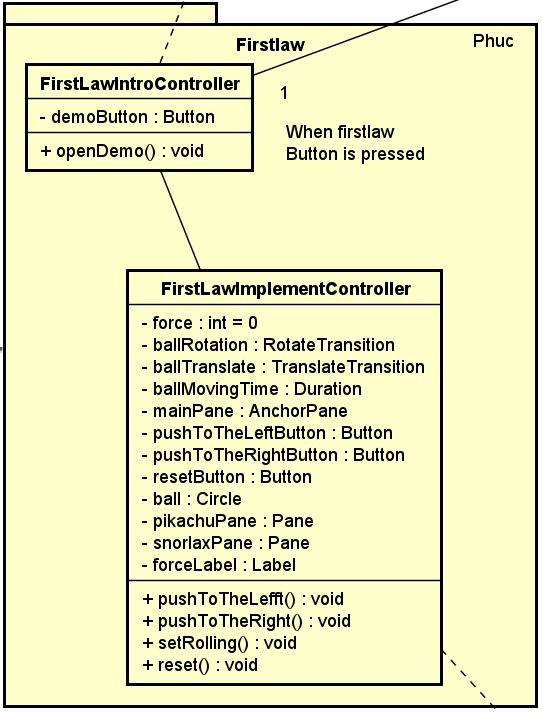
A screenshot of a cell phone

Description automatically generated

In this package, there are two classes:

* Slide : The class is responsible for loading FXML file ( which is a slide )
* Presentation: The class is responsible for adding slide to a presentation, controlling slide change by pressing left and right arrow button on keyboard, fit slide to screen

### Firstlaw



In this package, there are two classes:

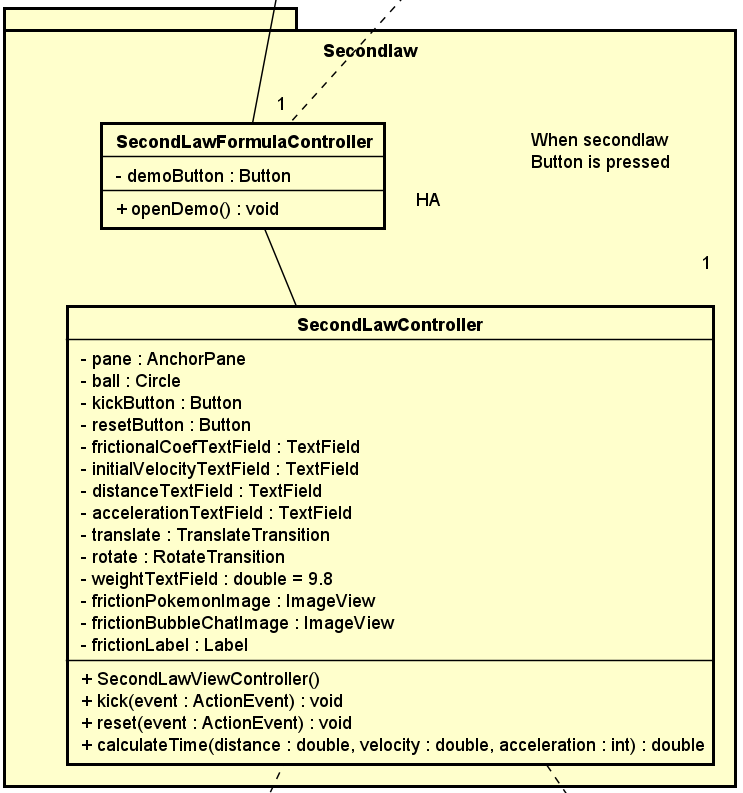
* FirstLawIntroController:

This class controller is responsible for rendering the First law theory GUI file in a new window. Within this window, there is a *demoButton* to redirect user to a new window for the law’s demonstration. When this button is pressed, FirstLawIntroController class will call *openDemo* method and do the below action.

* FirstLawImplementController:

|  |  |  |
| --- | --- | --- |
| ATTRIBUTE | METHOD | FLOW |
| force |  | The level of force the ball will roll with |
| ball |  | Circle filled with image: the main component for visualization |
| ballRotation |  | Represent rotation and translate transition of the ball |
| ballTranslate |  |  |
| ballMovingTime |  | Duration to control time of translate transition |
| mainPane |  | AnchorPane will be used to get its property for indefinite translate transition of ball |
| pikachuPane |  | Pane showing information on ball rolling status |
| snorlaxPane |  | Pane showing information on ball resting status |
| forceLabel |  | Label showing force level of rolling ball |
| PushToTheLeftButton | PushToTheLeft( )  PushToTheRight( ) | When one of these two buttons is pressed, the corresponding method will be called and *forceLevel* will either be increased by 1 ( push to the right ) or decreased by 1 ( push to the left ) and call *setRolling( )* method to make the ball roll |
| PushToTheRightButton |
|  | setRolling( ) | This method is used to initialize rotate and translate transition on the ball, and update related values ( ballMovingTime, angle, setFromX, setToX, playFrom):   * When ball is either null or not initialized, this method will set rotate and translate for ball and ball will roll from the middle of the scene * When ball is not null anymore, this method will update values to make the ball roll with different rotate and translate duration according to the corresponding force level * This method also change Pane to show corresponding information on the ball rolling status |
| resetButton | reset( ) | This method sets the ball rotate, translate transition ballMovingTime to null value, force to 0, and set the ball to be at the original position . |

### Secondlaw



In this package there are two classes:

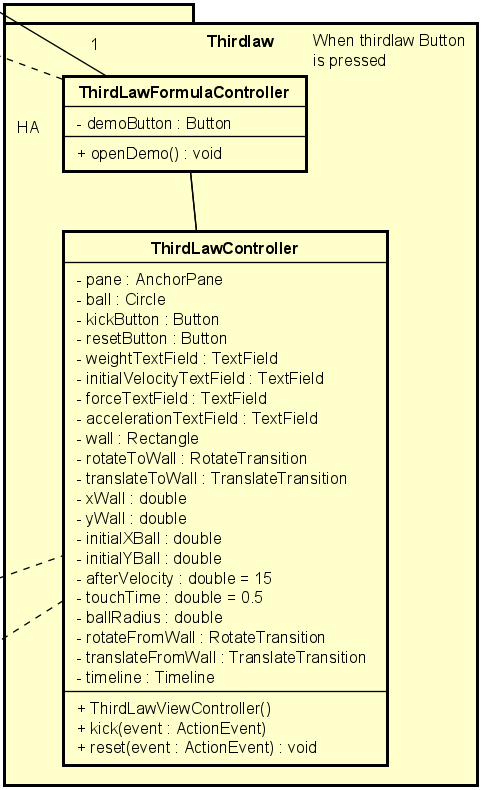
* SecondLawFormulaController
* SecondLawController
* SecondLawFormulaController:

This class controller is responsible for rendering the Second formula theory GUI file in a new window. Within this window, there is a *demoButton* to redirect user to a new window for the law’s demonstration. When this button is pressed, SecondLawFormulaController class will call *openDemo* method and do the below action.

* SecondLawController:

|  |  |  |
| --- | --- | --- |
| COMPONENT | METHOD | FLOW |
| ball |  | Main component for visualization. Will be applied translation and rotation to imitate a ball rolling. |
| kickButton | kick | When *kickButton* is pressed, the program will take user input from *frictionalCoefTextField* and *initialVelocityTextField* to calculate the distance and acceleration that the ball travels with and display them. |
| frictionalCoefTextField |
| initialVelocityTextField |
| distanceTextField |  | These two fields are used to display the result distance and acceleration of the ball calculated in the *kick* method. |
| accelerationTextField |
| rotate |  | When *kick* method is called, it will use the calculated parameters to move the ball accordingly. *Rotate* is for rotation, translate is for translation, thus, when both animations are played, they will mimic the ball’s movement. |
| translate |
|  | reset | This function sets all components back to their original status:  + Textfields: clear all inputs  + Ball: put it back to its original position  + Transition: stop all transitions and make them null |
|  | calculate | Utility function to help kick calculate time the ball travel, thus will help find the distance and the acceleration |

### Thirdlaw



In this package there are two classes:

* ThirdFormulaController
* ThirdLawController
* ThirdLawFormulaController:

This class controller is responsible for rendering the third formula theory GUI file in a new window. Within this window, there is a *demoButton* to redirect user to a new window for the law’s demonstration. When this button is pressed, *ThirdLawFormulaController* class will call *openDemo* method and do the below action.

* ThirdLawController:

|  |  |  |
| --- | --- | --- |
| COMPONENT | METHOD | FLOW |
| ball |  | Main component for visualization. Will be applied translation and rotation to imitate a ball rolling. |
| initialXBall |  | Initial X coordinate of the ball on the screen |
| initialYBall |  | Initial Y coordinate of the ball on the screen |
| wall |  | A rectangle to represents a wall for the ball to bounce back from |
| ballRadius |  | Radius of the ball |
| xWall |  | Upper-left corner’s X coordinate of the wall |
| afterVelocity |  | Constant predefined velocity of the ball when bounce back from the wall |
| touchTime |  | The duration in which the ball is in contact with the wall during collision |
| kickButton | kick | When *kickButton* is pressed, the program will take user input from weightTextField and *initialVelocityTextField* to calculate the ball’s traveling time and distance. |
| weightTextField |
| initialVelocityTextField |
| forceTextField |  | These two fields are used to display the result force exerted upon the ball by the wall and acceleration of the ball calculated in the *kick* method. |
| accelerationTextField |
| rotateToWall |  | These animations are used to represent the ball’s movement when moving towards a wall and from the wall outwards. They are all included in the timeline for a continuous process.  rotateToWall and translateToWall are incorporated to mimic the ball’s moving to the wall while rotateFromWall and translateFromWall do the opposite |
| translateToWall |
| rotateFromWall |
| translateFromWall |
| timeline |
|  | reset | This function sets all components back to their original status:  + Textfields: clear all inputs  + Ball: put it back to its original position  + Transition: stop all transitions and make them null |

### Functional interfaces

There are three functional interfaces in the program, each having only one method:

* Resetable
* Kickable
* TheoryController

Detailed description of each interface is as follow:

* TheoryController

A close up of a sign

Description automatically generated

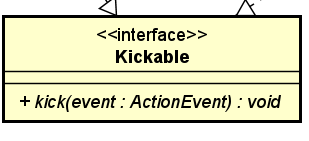
This interface is a common blueprint for classes that inherit from it. The only function that it has is *openDemo().*

There are three classes that are its children, namely:

* FirstLawIntroController
* SecondLawFormulaController
* ThirdLawFormulaController

Each of the above class implements their own version of openDemo and as the method’s name suggested, will open a demonstration of the corresponding law.

* Kickable



This interface is a common blueprint for classes that inherit from it. The only function that it has is *kick(ActionEvent event).*

There are two classes that are its children, namely:

* SecondLawViewController
* ThirdLawViewController

Each of the above class implements their own version of *kick* as has been discussed above.

* Resetable

A picture containing bird

Description automatically generated

This interface is a common blueprint for classes that inherit from it. The only function that it has is *reset(ActionEvent event).*

There are two classes that are its children, namely:

* FirstLawImplementController
* SecondLawViewController
* ThirdLawViewController

Each of the above class implements their own version of *reset* as has been discussed above.

# **ASSIGNMENT OF MEMBERS**

|  |  |  |  |
| --- | --- | --- | --- |
| **Package** | **Component/Method** | **Phuc** | **Hong Anh** |
| Menu | MenuController.java | x |  |
| Menu.fxml | x |  |
| Introduction | Presentation.java | x |  |
| Slide.java | x |  |
| Slide1 🡪 Slide4.fxml | x |  |
| Firstlaw | FirstLawImplementController.java | x |  |
| FirstLawIntroController.java | x |  |
| FirstLawImplement.fxml | x |  |
| FirstLawIntro.fxml | x |  |
| Secondlaw | SecondLawFormulaController.java |  | x |
| SecondLawController.java |  | x |
| SecondLawFormulaView.fxml |  | x |
| SecondLawView.fxml |  | x |
| Thirdlaw | ThirdLawFormulaController.java |  | x |
| ThirdLawController.java |  | x |
| ThirdLawFormulaView.fxml |  | x |
| ThirdLawView.fxml |  | x |
| Interfaces | Kickable, Resetable, TheoryController | x | x |
| GUI Ideas | Menu GUI | x |  |
| Demo GUI | x |  |
| Theory GUI | x |  |
| Design | Class Diagram | x | x |
| Use Case Diagram |  | x |
| Problem | Report | x | x |
| Presentation | PowerPoint slide | x |  |
| Demo video | Demo video | x |  |

# **REFERENCE:**

* For Presentation.java and Slide.java : Copy with modify source code from: [*https://www.javacodegeeks.com/2013/02/a-minimal-javafx-presentation-in-javafx.html*](https://www.javacodegeeks.com/2013/02/a-minimal-javafx-presentation-in-javafx.html)
* Connecting custom FXML file to controller: <https://docs.oracle.com/javafx/2/fxml_get_started/custom_control.htm>
* For limiting upper bound of Frictional and Velocity text field in SecondLawViewController.java: Idea from <https://stackoverflow.com/a/41514286>
* Timeline animation in ThirdLawViewController.java: <https://docs.oracle.com/javafx/2/animations/basics.htm>