

FBLA Genie

Final Version

Generated by Doxygen 1.8.17



<b>1 Hierarchical Index</b>	<b>1</b>
1.1 Class Hierarchy	1
<b>2 Class Index</b>	<b>3</b>
2.1 Class List	3
<b>3 Class Documentation</b>	<b>5</b>
3.1 com.frontend.Add.AddHours.AddHours Class Reference	5
3.1.1 Detailed Description	5
3.2 com.frontend.Application Class Reference	5
3.2.1 Detailed Description	6
3.2.2 Member Function Documentation	6
3.2.2.1 main()	6
3.3 com.frontend.Charts Class Reference	6
3.3.1 Detailed Description	7
3.3.2 Member Function Documentation	7
3.3.2.1 achievedGoalPieChart()	7
3.3.2.2 barGraphCommunityServiceCategoryAchieved()	8
3.3.2.3 barGraphCommunityServiceCategoryGoals()	8
3.3.2.4 contributionTreeMapChart() [1/2]	9
3.3.2.5 contributionTreeMapChart() [2/2]	9
3.3.2.6 currentDivisionChart()	9
3.3.2.7 goalDivisionChart()	10
3.3.2.8 monthLineGraph()	10
3.3.2.9 solidGauge()	11
3.4 com.frontend.Security.ConfigureUIServiceInitListener Class Reference	11
3.4.1 Detailed Description	12
3.5 com.frontend.Add.CreateStudent.CreateStudent Class Reference	12
3.5.1 Detailed Description	12
3.6 com.backend.Date Class Reference	12
3.6.1 Detailed Description	14
3.6.2 Constructor & Destructor Documentation	14
3.6.2.1 Date() [1/2]	14
3.6.2.2 Date() [2/2]	14
3.6.3 Member Function Documentation	15
3.6.3.1 clone()	15
3.6.3.2 compare()	15
3.6.3.3 compareTo()	16
3.6.3.4 equals()	16
3.6.3.5 equalsNoDay()	16
3.6.3.6 fakeDate()	17
3.6.3.7 getDay()	17
3.6.3.8 getLocalDate()	17

3.6.3.9 <a href="#">getMonth()</a> [1/3]	18
3.6.3.10 <a href="#">getMonth()</a> [2/3]	18
3.6.3.11 <a href="#">getMonth()</a> [3/3]	18
3.6.3.12 <a href="#">getMonthString()</a>	19
3.6.3.13 <a href="#">getYear()</a>	19
3.6.3.14 <a href="#">optionToDate()</a>	19
3.6.3.15 <a href="#">setDate()</a> [1/2]	20
3.6.3.16 <a href="#">setDate()</a> [2/2]	20
3.6.3.17 <a href="#">setDay()</a>	20
3.6.3.18 <a href="#">setMonth()</a>	21
3.6.3.19 <a href="#">setYear()</a>	21
3.6.3.20 <a href="#">toString()</a>	21
3.6.3.21 <a href="#">toStringRegular()</a>	22
3.7 <a href="#">com.frontend.Documentation.Documentation Class Reference</a>	22
3.7.1 Detailed Description	22
3.8 <a href="#">com.backend.Event Class Reference</a>	22
3.8.1 Detailed Description	24
3.8.2 Constructor & Destructor Documentation	24
3.8.2.1 <a href="#">Event()</a> [1/2]	24
3.8.2.2 <a href="#">Event()</a> [2/2]	25
3.8.3 Member Function Documentation	25
3.8.3.1 <a href="#">compare()</a>	25
3.8.3.2 <a href="#">compareTo()</a>	26
3.8.3.3 <a href="#">getDate()</a>	26
3.8.3.4 <a href="#">getDay()</a>	26
3.8.3.5 <a href="#">getEventName()</a>	27
3.8.3.6 <a href="#">getHours()</a>	27
3.8.3.7 <a href="#">getLocalDate()</a>	27
3.8.3.8 <a href="#">getMonth()</a>	27
3.8.3.9 <a href="#">getMonthsWithYear()</a>	27
3.8.3.10 <a href="#">getTotalHours()</a>	28
3.8.3.11 <a href="#">getYear()</a>	28
3.8.3.12 <a href="#">monthsInRange()</a>	28
3.8.3.13 <a href="#">setDate()</a> [1/4]	29
3.8.3.14 <a href="#">setDate()</a> [2/4]	29
3.8.3.15 <a href="#">setDate()</a> [3/4]	30
3.8.3.16 <a href="#">setDate()</a> [4/4]	30
3.8.3.17 <a href="#">setDay()</a>	30
3.8.3.18 <a href="#">setEventName()</a>	31
3.8.3.19 <a href="#">setHours()</a> [1/2]	31
3.8.3.20 <a href="#">setHours()</a> [2/2]	31
3.8.3.21 <a href="#">setMonth()</a>	31

3.8.3.22 setYear()	33
3.8.3.23 toString()	33
3.8.3.24 updateEvent()	33
3.9 com.frontend.GetStudentInformation.GetStudentInformation Class Reference	34
3.9.1 Detailed Description	34
3.9.2 Member Function Documentation	34
3.9.2.1 deleteButton()	34
3.9.2.2 expandButton()	35
3.10 com.frontend.Home Class Reference	35
3.10.1 Detailed Description	36
3.11 com.frontend.Login.LoginView Class Reference	36
3.11.1 Detailed Description	36
3.11.2 Member Function Documentation	36
3.11.2.1 beforeEnter()	36
3.12 com.frontend.MainView Class Reference	37
3.12.1 Detailed Description	38
3.12.2 Member Function Documentation	38
3.12.2.1 getTabs()	38
3.13 com.backend.MySQLMethods Class Reference	38
3.13.1 Detailed Description	40
3.13.2 Member Function Documentation	41
3.13.2.1 addStudentHours()	41
3.13.2.2 addUserPass()	41
3.13.2.3 createDatabase()	42
3.13.2.4 createStudent()	42
3.13.2.5 createStudentTable()	42
3.13.2.6 delete() [1/2]	43
3.13.2.7 delete() [2/2]	43
3.13.2.8 getHoursAll()	44
3.13.2.9 getHoursMonth()	44
3.13.2.10 getHoursWeek()	44
3.13.2.11 getHoursYear()	45
3.13.2.12 getStudentData() [1/2]	45
3.13.2.13 getStudentData() [2/2]	45
3.13.2.14 getStudents()	46
3.13.2.15 makeName() [1/2]	46
3.13.2.16 makeName() [2/2]	47
3.13.2.17 numberOfRows()	47
3.13.2.18 round()	47
3.13.2.19 selectPass()	48
3.13.2.20 selectStudentEventsAsEvent() [1/2]	48
3.13.2.21 selectStudentEventsAsEvent() [2/2]	49

3.13.2.22 selectStudentEventsInRange()	49
3.13.2.23 selectTrackerAsStudent()	50
3.13.2.24 selectTrackerDouble()	50
3.13.2.25 selectTrackerInt()	50
3.13.2.26 selectTrackerShort()	52
3.13.2.27 selectTrackerString()	52
3.13.2.28 updateEvent()	53
3.13.2.29 updateFirstName()	53
3.13.2.30 updateLastName()	54
3.13.2.31 updateStudentID()	54
3.13.2.32 updateToCurrentDate()	54
3.13.2.33 updateTracker() [1/2]	55
3.13.2.34 updateTracker() [2/2]	55
3.14 com.backend.Percent Class Reference	56
3.14.1 Detailed Description	57
3.14.2 Constructor & Destructor Documentation	57
3.14.2.1 Percent() [1/2]	57
3.14.2.2 Percent() [2/2]	57
3.14.3 Member Function Documentation	58
3.14.3.1 doubleValue()	58
3.14.3.2 floatValue()	58
3.14.3.3 getPercent()	58
3.14.3.4 intValue()	59
3.14.3.5 longValue()	59
3.14.3.6 round()	59
3.14.3.7 toString()	60
3.15 com.frontend.Reports.Reports Class Reference	60
3.15.1 Detailed Description	61
3.15.2 Member Function Documentation	61
3.15.2.1 groupReport()	61
3.15.2.2 individualReport()	61
3.15.2.3 makeBulletList()	61
3.15.2.4 nextSection()	62
3.15.2.5 setText()	62
3.16 com.frontend.Security.SecurityConfiguration Class Reference	63
3.16.1 Detailed Description	63
3.16.2 Member Function Documentation	63
3.16.2.1 userDetailsService()	63
3.17 com.frontend.Security.SecurityUtils Class Reference	64
3.17.1 Detailed Description	64
3.18 com.backend.Student Class Reference	64
3.18.1 Detailed Description	65

3.18.2 Constructor & Destructor Documentation	65
3.18.2.1 Student() [1/2]	65
3.18.2.2 Student() [2/2]	66
3.18.3 Member Function Documentation	66
3.18.3.1 getCreateNewStudent()	66
3.18.3.2 getFirstName()	66
3.18.3.3 getFullName()	67
3.18.3.4 getLastName()	67
3.18.3.5 getStudent()	67
3.18.3.6 getStudentData()	67
3.18.3.7 getStudentID()	68
3.18.3.8 setFirstName()	68
3.18.3.9 setLastName()	68
3.18.3.10 setStudent()	68
3.18.3.11 setStudentID()	69
3.18.3.12 toString()	69
3.19 com.backend.StudentData Class Reference	69
3.19.1 Detailed Description	71
3.19.2 Constructor & Destructor Documentation	72
3.19.2.1 StudentData() [1/2]	72
3.19.2.2 StudentData() [2/2]	72
3.19.3 Member Function Documentation	72
3.19.3.1 achievedGoal()	73
3.19.3.2 getActiveCategory()	73
3.19.3.3 getActiveGoal()	73
3.19.3.4 getAverageCategory()	74
3.19.3.5 getAverageGoal()	74
3.19.3.6 getCommunityServiceCategory()	74
3.19.3.7 getCommunityServiceCategoryInt()	75
3.19.3.8 getCommunityServiceHours()	75
3.19.3.9 getCurrentCommunityServiceCategory()	75
3.19.3.10 getCurrentCommunityServiceCategoryInt()	76
3.19.3.11 getEmail()	76
3.19.3.12 getGrade()	76
3.19.3.13 getGradeInt()	77
3.19.3.14 getLastEdited()	77
3.19.3.15 getYearsDone()	77
3.19.3.16 isActive()	77
3.19.3.17 removeActive()	77
3.19.3.18 removeInactive()	78
3.19.3.19 setCommunityServiceCategory()	78
3.19.3.20 setCommunityServiceHours()	79

3.19.3.21 setCommunityServiceHoursFromSelect() [1/2]	79
3.19.3.22 setCommunityServiceHoursFromSelect() [2/2]	79
3.19.3.23 setCommunityServiceHoursTemp()	79
3.19.3.24 setEmail()	81
3.19.3.25 setFirstName()	81
3.19.3.26 setGrade() [1/2]	81
3.19.3.27 setGrade() [2/2]	82
3.19.3.28 setLastEdited()	82
3.19.3.29 setLastName()	82
3.19.3.30 setStudentID() [1/2]	83
3.19.3.31 setStudentID() [2/2]	83
3.19.3.32 setYearsDone() [1/2]	83
3.19.3.33 setYearsDone() [2/2]	83

<b>Index</b>	<b>87</b>
--------------	-----------



# Chapter 1

## Hierarchical Index

### 1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

com.frontend.Charts . . . . .	6
Cloneable	
com.backend.Date . . . . .	12
com.backend.Event . . . . .	22
com.backend.Percent . . . . .	56
com.backend.Student . . . . .	64
com.backend.Event . . . . .	22
com.backend.StudentData . . . . .	69
com.backend.StudentData . . . . .	69
Comparable	
com.backend.Date . . . . .	12
com.backend.Event . . . . .	22
com.backend.MySQLMethods . . . . .	38
Number	
com.backend.Percent . . . . .	56
com.frontend.Security.SecurityUtils . . . . .	64
AppLayout	
com.frontend.MainView . . . . .	37
BeforeEnterObserver	
com.frontend.Login.LoginView . . . . .	36
Comparator	
com.backend.Date . . . . .	12
com.backend.Event . . . . .	22
SpringBootServletInitializer	
com.frontend.Application . . . . .	5
VaadinServiceInitListener	
com.frontend.Security.ConfigureUIServiceInitListener . . . . .	11
VerticalLayout	
com.frontend.Add.AddHours.AddHours . . . . .	5
com.frontend.Add.CreateStudent.CreateStudent . . . . .	12
com.frontend.Documentation.Documentation . . . . .	22
com.frontend.GetStudentInformation.GetStudentInformation . . . . .	34
com.frontend.Home . . . . .	35
com.frontend.Login.LoginView . . . . .	36
com.frontend.Reports.Reports . . . . .	60
WebSecurityConfigurerAdapter	
com.frontend.Security.SecurityConfiguration . . . . .	63



## Chapter 2

# Class Index

### 2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">com.frontend.Add.AddHours.AddHours</a>	5
The Class for Adding Hours into a Student . . . . .	
<a href="#">com.frontend.Application</a>	5
The entry point of the Spring Boot application . . . . .	
<a href="#">com.frontend.Charts</a>	6
A Class to help make making classes much easier . . . . .	
<a href="#">com.frontend.Security.ConfigureUIServiceInitListener</a>	11
A Class which forces redirection to LoginView while someone is not logged in . . . . .	
<a href="#">com.frontend.Add.CreateStudent.CreateStudent</a>	12
The Class Used to Allow a User to Add a New Student to their List of Students . . . . .	
<a href="#">com.backend.Date</a>	12
Creates a date class to store the date . . . . .	
<a href="#">com.frontend.Documentation.Documentation</a>	22
<a href="#">com.backend.Event</a>	22
A Class to Store Events of Students . . . . .	
<a href="#">com.frontend.GetStudentInformation.GetStudentInformation</a>	34
Gets All Student's General Information and Allows Editing . . . . .	
<a href="#">com.frontend.Home</a>	35
Home is the home view for the App . . . . .	
<a href="#">com.frontend.Login.LoginView</a>	36
A Class that allows logging in to the system . . . . .	
<a href="#">com.frontend.MainView</a>	37
MainView is the main view of the entire app . . . . .	
<a href="#">com.backend.MySQLMethods</a>	38
This Class will hold all the methods to allow manipulation of the Data in the Database . . . . .	
<a href="#">com.backend.Percent</a>	56
A Class to Use for <a href="#">Percent</a> Data Types . . . . .	
<a href="#">com.frontend.Reports.Reports</a>	60
The main class for Generating <a href="#">Reports</a> . . . . .	
<a href="#">com.frontend.Security.SecurityConfiguration</a>	63
Configures security with username, password, and different redirection options . . . . .	
<a href="#">com.frontend.Security.SecurityUtils</a>	64
A Class for Util methods in security . . . . .	
<a href="#">com.backend.Student</a>	64
The General <a href="#">Student</a> Class to Store Basic <a href="#">Student</a> Information . . . . .	
<a href="#">com.backend.StudentData</a>	69
This is a class which helps in running MySQL Methods and takes in all the values needed for the main student . . . . .	



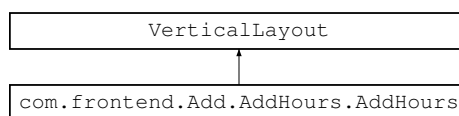
## Chapter 3

# Class Documentation

### 3.1 com.frontend.Add.AddHours.AddHours Class Reference

The Class for Adding Hours into a Student.

Inheritance diagram for com.frontend.Add.AddHours.AddHours:



#### Public Member Functions

- [AddHours \(\)](#)  
*The Add Hours View.*

#### 3.1.1 Detailed Description

The Class for Adding Hours into a Student.

Definition at line 37 of file `AddHours.java`.

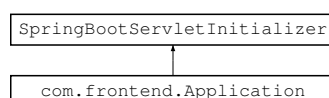
The documentation for this class was generated from the following file:

- `AddHours.java`

### 3.2 com.frontend.Application Class Reference

The entry point of the Spring Boot application.

Inheritance diagram for com.frontend.Application:



## Static Public Member Functions

- static void [main](#) (String[] args)  
*For Spring-Boot Running.*

### 3.2.1 Detailed Description

The entry point of the Spring Boot application.

Definition at line 12 of file Application.java.

### 3.2.2 Member Function Documentation

#### 3.2.2.1 main()

```
static void com.frontend.Application.main (
    String[] args ) [static]
```

For Spring-Boot Running.

#### Parameters

<i>args</i>	standard main method argument
-------------	-------------------------------

Definition at line 19 of file Application.java.

The documentation for this class was generated from the following file:

- Application.java

## 3.3 com.frontend.Charts Class Reference

A Class to help make making classes much easier.

### Static Public Member Functions

- static Chart [solidGauge](#) (double current, double max, int colorIndex)  
*Creates a solid gauge chart for the hours a student has compared to a certain category..*
- static Chart [monthLineGraph](#) (String title, List< [Event](#) > events)  
*Generates a Line Graph so that, for long-term reports, hours can be tracked over time to analyze consistent growth.*
- static Chart [contributionTreeMapChart](#) (List< [StudentData](#) > studentDataList, String chartType)  
*Generates a TreeMap chart to help divide which students contributed the most in a certain date range.*
- static Chart [contributionTreeMapChart](#) (String chartType)

- Generates a TreeMap chart to help divide which students contributed the most in a certain date range.*

  - static Chart `barGraphCommunityServiceCategoryGoals` (List< `StudentData` > studentDataList)

*Generates a bar graph with student's Community Service Category Goals.*
- static Chart `barGraphCommunityServiceCategoryAchieved` (List< `StudentData` > studentDataList)

*Generates a Bar Graph for the number of Community Service Award's achieved by students.*
- static Chart `achievedGoalPieChart` (List< `StudentData` > list, String title)

*A Pie Chart to show the number of student's who achieved their goals vs the number of those who didn't.*
- static Chart `goalDivisionChart` (List< `StudentData` > list, String title)

*Shows the different goals each student has in a Pie Chart format.*
- static Chart `currentDivisionChart` (List< `StudentData` > list, String title)

*A Pie Chart to show the division between the currently achieved goals of students.*

## Static Public Attributes

- static final String `WEEK_CHART` = "Week"
- A String identifying the option range of "Week".*
- static final String `MONTH_CHART` = "Month"
- A String identifying the option range of "Month".*
- static final String `YEAR_CHART` = "Year"
- A String identifying the option range of "Year".*
- static final String `ALL_TIME_CHART` = "All Time"
- A String identifying the option range of "All Time".*

### 3.3.1 Detailed Description

A Class to help make making classes much easier.

Definition at line 18 of file Charts.java.

### 3.3.2 Member Function Documentation

#### 3.3.2.1 achievedGoalPieChart()

```
static Chart com.frontend.Charts.achievedGoalPieChart (
    List< StudentData > list,
    String title ) [static]
```

A Pie Chart to show the number of student's who achieved their goals vs the number of those who didn't.

#### Parameters

<i>list</i>	All of the students
<i>title</i>	The title of the chart

**Returns**

The Pie Chart of Divisions

Definition at line 383 of file Charts.java.

**3.3.2.2 barGraphCommunityServiceCategoryAchieved()**

```
static Chart com.frontend.Charts.barGraphCommunityServiceCategoryAchieved (
    List< StudentData > studentDataList ) [static]
```

Generates a Bar Graph for the number of Community Service Award's achieved by students.

In this graph, if a student achieved a higher award, they are ignored in the lower award.

**Parameters**

<i>studentDataList</i>	The List of Students
------------------------	----------------------

**Returns**

A Bar Graph

Definition at line 316 of file Charts.java.

**3.3.2.3 barGraphCommunityServiceCategoryGoals()**

```
static Chart com.frontend.Charts.barGraphCommunityServiceCategoryGoals (
    List< StudentData > studentDataList ) [static]
```

Generates a bar graph with student's Community Service Category Goals.

**Parameters**

<i>studentDataList</i>	The List of Students
------------------------	----------------------

**Returns**

A Bar Graph

Definition at line 267 of file Charts.java.



### 3.3.2.4 contributionTreeMapChart() [1/2]

```
static Chart com.frontend.Charts.contributionTreeMapChart (
    List< StudentData > studentDataList,
    String chartType ) [static]
```

Generates a TreeMap chart to help divide which students contributed the most in a certain date range.

#### Parameters

<i>studentDataList</i>	The List of Students to be included in the chart
<i>chartType</i>	The type of chart, as in the date it covers

#### Returns

A TreeMap Char

Definition at line 167 of file Charts.java.

### 3.3.2.5 contributionTreeMapChart() [2/2]

```
static Chart com.frontend.Charts.contributionTreeMapChart (
    String chartType ) [static]
```

Generates a TreeMap chart to help divide which students contributed the most in a certain date range.

Automatically gets the Student Hour data based on the Chart Type Provided

#### Parameters

<i>chartType</i>	The type of chart, as in the date it covers
------------------	---

#### Returns

A TreeMap Char

Definition at line 207 of file Charts.java.

### 3.3.2.6 currentDivisionChart()

```
static Chart com.frontend.Charts.currentDivisionChart (
    List< StudentData > list,
    String title ) [static]
```

A Pie Chart to show the division between the currently achieved goals of students.

**Parameters**

<i>list</i>	The Students
<i>title</i>	The Title of the Chart

**Returns**

The Pie Chart

Definition at line 476 of file Charts.java.

**3.3.2.7 goalDivisionChart()**

```
static Chart com.frontend.Charts.goalDivisionChart (  
    List< StudentData > list,  
    String title ) [static]
```

Shows the different goals each student has in a Pie Chart format.

**Parameters**

<i>list</i>	The list of students
<i>title</i>	The title of the chart

**Returns**

The Pie Chart

Definition at line 422 of file Charts.java.

**3.3.2.8 monthLineGraph()**

```
static Chart com.frontend.Charts.monthLineGraph (  
    String title,  
    List< Event > events ) [static]
```

Generates a Line Graph so that, for long-term reports, hours can be tracked over time to analyze consistent growth.

**Parameters**

<i>title</i>	The Chart Title
<i>events</i>	The List of Events the chart should encompass

**Returns**

A Line Graph

Definition at line 111 of file Charts.java.

**3.3.2.9 solidGauge()**

```
static Chart com.frontend.Charts.solidGauge (
    double current,
    double max,
    int colorIndex ) [static]
```

Creates a solid gauge chart for the hours a student has compared to a certain category,.

**Parameters**

<i>current</i>	The Current Number of Hours
<i>max</i>	The Max Possible Number of Hours, or hours to achieve an award
<i>colorIndex</i>	The Color of the Chart

**Returns**

A Solid Gauge Chart

Definition at line 45 of file Charts.java.

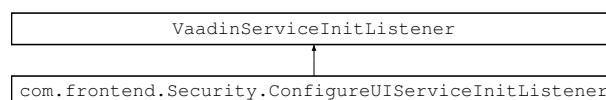
The documentation for this class was generated from the following file:

- Charts.java

## 3.4 com.frontend.Security.ConfigureUIServiceInitListener Class Reference

A Class which forces redirection to LoginView while someone is not logged in.

Inheritance diagram for com.frontend.Security.ConfigureUIServiceInitListener:

**Public Member Functions**

- void **serviceInit** (ServiceInitEvent event)

### 3.4.1 Detailed Description

A Class which forces redirection to LoginView while someone is not logged in.

This keeps anyone from accessing any part of the app without being validated

Definition at line 16 of file ConfigureUIServiceInitListener.java.

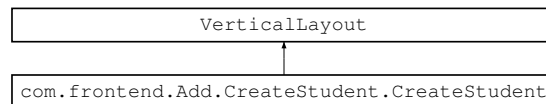
The documentation for this class was generated from the following file:

- ConfigureUIServiceInitListener.java

## 3.5 com.frontend.Add.CreateStudent.CreateStudent Class Reference

The Class Used to Allow a User to Add a New Student to their List of Students.

Inheritance diagram for com.frontend.Add.CreateStudent.CreateStudent:



### Public Member Functions

- [CreateStudent \(\)](#)  
*The Overall Path to create a new Student.*

### 3.5.1 Detailed Description

The Class Used to Allow a User to Add a New Student to their List of Students.

Definition at line 34 of file CreateStudent.java.

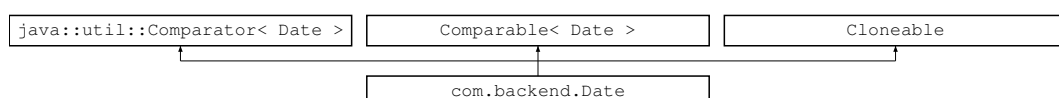
The documentation for this class was generated from the following file:

- CreateStudent.java

## 3.6 com.backend.Date Class Reference

Creates a date class to store the date.

Inheritance diagram for com.backend.Date:



## Public Member Functions

- [Date](#) (int year, int month, int day)  
*Constructs a date given all its information.*
- [Date](#) ()  
*Constructs a [Date](#) of the First Day of the Common Era, to use as a Minimum [Date](#).*
- [Date](#) (boolean noDate)  
*A [Date](#) which starts off as "Fake" because it hasn't been given any value.*
- boolean [fakeDate](#) ()  
*Gets the value of the "Fake Date" variable.*
- int [getYear](#) ()  
*Gets the Year.*
- void [setYear](#) (int year)  
*Changes the Year.*
- int [getMonth](#) ()  
*Gets the Month.*
- void [setMonth](#) (int month)  
*Changes the Month.*
- int [getDay](#) ()  
*Gets the Day.*
- void [setDay](#) (int day)  
*Changes the Day.*
- void [setDate](#) (String date)  
*Takes a [Date](#) in the String format and converts it into a [Date](#) object.*
- LocalDate [getLocalDate](#) ()  
*Returns the LocalDate of the Current [Date](#).*
- void [setDate](#) (LocalDate date)  
*Uses the modern version of [Date](#) in java, LocalDate, and turns it into the backend [Date](#).*
- String [getMonthString](#) ()  
*Returns the month value as its respective string without the need to direct to static methods.*
- String [toString](#) ()  
*Prints out the [Date](#) as a String in dd, MM, yyyy format.*
- String [toStringRegular](#) ()  
*Uses the standard format of a [Date](#), yyyy-mm-dd, and converts.*
- int [compareTo](#) ([Date](#) o)  
*Compares the current date with another to see which one comes first.*
- int [compare](#) ([Date](#) o1, [Date](#) o2)  
*Compares two dates, o1 and o2, to find which one comes first.*
- boolean [equals](#) ([Date](#) date)  
*Compares to see if two dates are the same.*
- boolean [equalsNoDay](#) ([Date](#) date)  
*Compares to see if two dates are equal, ignoring the day.*
- void [incrementDay](#) ()  
*Increments the Day by One, Changing the Month when necessary.*
- void [incrementByMonth](#) ()  
*Increments the month by one, changing the year if it hits 13.*
- void [incrementYear](#) ()  
*Increments the Year by 1.*

## Static Public Member Functions

- static [Date](#) [optionToDate](#) (String s)  
*Takes the options included in [Charts.java](#) and returns their related [Date](#) values.*
- static int [getMonth](#) (String month)  
*Converts a String of the month to its respective Integer values.*
- static String [getMonth](#) (int month)  
*Converts an Integer of the month to its respective String-based version.*

## Protected Member Functions

- [Date](#) [clone](#) ()  
*Returns a copy of the [Date](#) without changing the original.*

### 3.6.1 Detailed Description

Creates a date class to store the date.

This would allow easy storage and manipulation of the date

Definition at line 13 of file Date.java.

### 3.6.2 Constructor & Destructor Documentation

#### 3.6.2.1 [Date\(\)](#) [1/2]

```
com.backend.Date.Date (
    int year,
    int month,
    int day )
```

Constructs a date given all its information.

##### Parameters

<i>year</i>	Year
<i>month</i>	Month
<i>day</i>	Day

Definition at line 41 of file Date.java.

#### 3.6.2.2 [Date\(\)](#) [2/2]

```
com.backend.Date.Date (
```

```
boolean noDate )
```

A [Date](#) which starts off as "Fake" because it hasn't been given any value.

#### Parameters

<i>noDate</i>	A boolean telling whether or not the date is Fake
---------------	---

Definition at line 59 of file Date.java.

### 3.6.3 Member Function Documentation

#### 3.6.3.1 clone()

```
Date com.backend.Date.clone ( ) [protected]
```

Returns a copy of the [Date](#) without changing the original.

This overrides the super method of clone to work for [Date](#)

#### Returns

A copy of the [Date](#)

Definition at line 194 of file Date.java.

#### 3.6.3.2 compare()

```
int com.backend.Date.compare (
    Date o1,
    Date o2 )
```

Compares two dates, o1 and o2, to find which one comes first.

#### Parameters

<i>o1</i>	The First <a href="#">Date</a> Being compared
<i>o2</i>	The Second <a href="#">Date</a> Being Compared

#### Returns

An integer containing the comparison

Definition at line 376 of file Date.java.

### 3.6.3.3 compareTo()

```
int com.backend.Date.compareTo (
    Date o )
```

Compares the current date with another to see which one comes first.

#### Parameters

<i>o</i>	The date compare with
----------	-----------------------

#### Returns

An integer representing the difference in date values

Definition at line 353 of file Date.java.

### 3.6.3.4 equals()

```
boolean com.backend.Date.equals (
    Date date )
```

Compares to see if two dates are the same.

#### Parameters

<i>date</i>	the <a href="#">Date</a> compared
-------------	-----------------------------------

#### Returns

Whether they are equal or not

Definition at line 394 of file Date.java.

### 3.6.3.5 equalsNoDay()

```
boolean com.backend.Date.equalsNoDay (
    Date date )
```

Compares to see if two dates are equal, ignoring the day.

#### Parameters

<i>date</i>	The date being compared
-------------	-------------------------



**Returns**

Whether they are equal

Definition at line 404 of file Date.java.

**3.6.3.6 fakeDate()**

```
boolean com.backend.Date.fakeDate ( )
```

Gets the value of the "Fake Date" variable.

**Returns**

The "Fake Date" value

Definition at line 208 of file Date.java.

**3.6.3.7 getDay()**

```
int com.backend.Date.getDay ( )
```

Gets the Day.

**Returns**

The current Day

Definition at line 253 of file Date.java.

**3.6.3.8 getLocalDate()**

```
LocalDate com.backend.Date.getLocalDate ( )
```

Returns the LocalDate of the Current [Date](#).

**Returns**

The LocalDate of the [Date](#)

Definition at line 289 of file Date.java.

### 3.6.3.9 `getMonth()` [1/3]

```
int com.backend.Date.getMonth ( )
```

Gets the Month.

#### Returns

The current Month

Definition at line 235 of file Date.java.

### 3.6.3.10 `getMonth()` [2/3]

```
static String com.backend.Date.getMonth (
    int month ) [static]
```

Converts an Integer of the month to its respective String-based version.

This method allows the use of String-based months alongside their Integer-based counterparts, allowing easy conversion in one direction.

#### Parameters

<i>month</i>	The month as an Integer
--------------	-------------------------

#### Returns

The String value of the month

Definition at line 155 of file Date.java.

### 3.6.3.11 `getMonth()` [3/3]

```
static int com.backend.Date.getMonth (
    String month ) [static]
```

Converts a String of the month to its respective Integer values.

This method allows the use of String-based months alongside their Integer-based counterparts, allowing easy conversion in one direction.

#### Parameters

<i>month</i>	The month as a string
--------------	-----------------------

**Returns**

The integer value of the month

Definition at line 100 of file Date.java.

**3.6.3.12 getMonthString()**

```
String com.backend.Date.getMonthString ( )
```

Returns the month value as its respective string without the need to direct to static methods.

**Returns**

The String value of the Month

Definition at line 309 of file Date.java.

**3.6.3.13 getYear()**

```
int com.backend.Date.getYear ( )
```

Gets the Year.

**Returns**

the current Year

Definition at line 217 of file Date.java.

**3.6.3.14 optionToDate()**

```
static Date com.backend.Date.optionToDate (
    String s ) [static]
```

Takes the options included in [Charts.java](#) and returns their related [Date](#) values.

This is used for Dates which are based on the String Values, such as Week, Month, and Year. This typically is useful in reports when the user is choosing a date range

**Parameters**

s	The Option Selected
---	---------------------

#### Returns

A [Date](#) with the constraints put on by the Option

Definition at line 72 of file Date.java.

#### 3.6.3.15 setDate() [1/2]

```
void com.backend.Date.setDate (
    LocalDate date )
```

Uses the modern version of [Date](#) in java, LocalDate, and turns it into the backend [Date](#).

#### Parameters

<i>date</i>	The LocalDate being converted.
-------------	--------------------------------

Definition at line 298 of file Date.java.

#### 3.6.3.16 setDate() [2/2]

```
void com.backend.Date.setDate (
    String date )
```

Takes a [Date](#) in the String format and converts it into a [Date](#) object.

#### Parameters

<i>date</i>	the <a href="#">Date</a> as a String @apiNote For this method to work, the date has to be in the format of yyyy-mm-dd
-------------	---

Definition at line 272 of file Date.java.

#### 3.6.3.17 setDay()

```
void com.backend.Date.setDay (
    int day )
```

Changes the Day.

#### Parameters

<i>day</i>	The new Day
------------	-------------

Definition at line 262 of file Date.java.

#### 3.6.3.18 setMonth()

```
void com.backend.Date.setMonth (
    int month )
```

Changes the Month.

##### Parameters

<i>month</i>	The new Month
--------------	---------------

Definition at line 244 of file Date.java.

#### 3.6.3.19 setYear()

```
void com.backend.Date.setYear (
    int year )
```

Changes the Year.

##### Parameters

<i>year</i>	The New Year
-------------	--------------

Definition at line 226 of file Date.java.

#### 3.6.3.20 toString()

```
String com.backend.Date.toString ( )
```

Prints out the [Date](#) as a String in dd, MM, yyyy format.

##### Returns

The Entire [Date](#) as a String

Definition at line 319 of file Date.java.

### 3.6.3.21 toStringRegular()

```
String com.backend.Date.toStringRegular ( )
```

Uses the standard format of a [Date](#), yyyy-mm-dd, and converts.

#### Returns

A String with the [Date](#) in the Standard Format

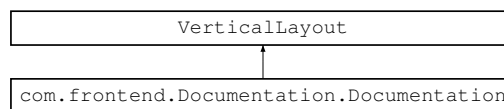
Definition at line 328 of file Date.java.

The documentation for this class was generated from the following file:

- Date.java

## 3.7 com.frontend.Documentation.Documentation Class Reference

Inheritance diagram for com.frontend.Documentation.Documentation:



### 3.7.1 Detailed Description

Definition at line 12 of file Documentation.java.

The documentation for this class was generated from the following file:

- Documentation.java

## 3.8 com.backend.Event Class Reference

A Class to Store Events of Students.

Inheritance diagram for com.backend.Event:



## Public Member Functions

- [Event](#) (String firstName, String lastName, int studentID, String eventName, double hours, int year, int month, int day)  
*Event Constructor with Each Value Individually.*
- [Event](#) ([Student](#) student, String eventName, double hours, [Date](#) date)  
*Event Constructor with Each Value Individually.*
- [Event](#) ()  
*Creates an empty event, only initializing the date to avoid NullPointerExceptions.*
- String [getEventName](#) ()  
*Gets the name of the Event.*
- void [setEventName](#) (String eventName)  
*Sets the name of the event, changing in the Database if necessary.*
- double [getHours](#) ()  
*Gets the length of the Event.*
- void [setHours](#) (double hours)  
*Changes the length of the Event, updating database if necessary.*
- void [setHours](#) (String hours)  
*Changes the length of the Event given a String, changing backend if necessary.*
- int [getYear](#) ()  
*Gets the Year.*
- void [setYear](#) (int year)  
*Changes the Year.*
- int [getMonth](#) ()  
*Gets the Month.*
- void [setMonth](#) (int month)  
*Changes the Month.*
- int [getDay](#) ()  
*Gets the Day.*
- void [setDay](#) (int day)  
*Changes the Day.*
- void [setDate](#) (int year, int month, int day)  
*Sets the date, changing the database if necessary.*
- [Date](#) [getDate](#) ()  
*Gets the Date.*
- void [setDate](#) (String date)  
*Sets the Date using a String of the new Date, depending on the format of the String.*
- void [setDate](#) (LocalDate localDate)  
*Sets the Date of the Event based on the LocalDate, changing the Database if necessary.*
- void [setDate](#) ([Date](#) date)  
*Sets the date given another Date.*
- void [setDateNoUpdate](#) (LocalDate localDate)
- LocalDate [getLocalDate](#) ()  
*Gets the LocalDate (the new version of java.util.Date) of the current Date.*
- void [addEvent](#) ()  
*Adds the Current Event to the database, taking all changes made.*
- void [delete](#) ()  
*This deletes the current event from the database, removing all traces.*
- String [toString](#) ()  
*Formats the Event in an easy-to-read manner.*
- void [updateEvent](#) ([Event](#) oldEvent, [Event](#) newEvent)

*Updates the event with all the new values in the backend database.*

- int [compareTo](#) ([Event](#) o)

*Compares the dates of two events.*

- int [compare](#) ([Event](#) o1, [Event](#) o2)

*Compares the dates of two events.*

## Static Public Member Functions

- static String[] [getMonthsWithYear](#) ([Date](#) startDateIn, [Date](#) endDateIn)

*Gets an Array of all the Months in the range between two months, including the start and the end.*

- static int [monthsInRange](#) ([Date](#) startDate, [Date](#) endDate)

*Gets the number of months in a range between two months.*

- static double [getTotalHours](#) (List< [Event](#) > eventList)

*Gets the total hours in a list of Events, especially useful when calculating the hours a student achieved in a certain range.*

## Additional Inherited Members

### 3.8.1 Detailed Description

A Class to Store Events of Students.

Definition at line 13 of file Event.java.

### 3.8.2 Constructor & Destructor Documentation

#### 3.8.2.1 [Event\(\)](#) [1/2]

```
com.backend.Event.Event (
    String firstName,
    String lastName,
    int studentID,
    String eventName,
    double hours,
    int year,
    int month,
    int day )
```

[Event](#) Constructor with Each Value Individually.

#### Parameters

<i>firstName</i>	The First Name of the <a href="#">Student</a>
<i>lastName</i>	The Last Name of the <a href="#">Student</a>
<i>studentID</i>	The <a href="#">Student</a> ID of the <a href="#">Student</a>
<i>eventName</i>	The Name of the <a href="#">Event</a>
<i>hours</i>	The Length of the <a href="#">Event</a>
<i>year</i>	The Year of the <a href="#">Event</a>
<i>month</i>	The Month of the <a href="#">Event</a>
<i>day</i>	The Day of the <a href="#">Event</a>



Definition at line 41 of file Event.java.

### 3.8.2.2 Event() [2/2]

```
com.backend.Event.Event (
    Student student,
    String eventName,
    double hours,
    Date date )
```

[Event](#) Constructor with Each Value Individually.

#### Parameters

<i>student</i>	The <a href="#">Student</a> doing the <a href="#">Event</a>
<i>eventName</i>	The Name of the <a href="#">Event</a>
<i>hours</i>	The Length of the <a href="#">Event</a>
<i>date</i>	The <a href="#">Date</a> of the <a href="#">Event</a>

Definition at line 57 of file Event.java.

## 3.8.3 Member Function Documentation

### 3.8.3.1 compare()

```
int com.backend.Event.compare (
    Event o1,
    Event o2 )
```

Compares the dates of two events.

#### Parameters

<i>o1</i>	The First <a href="#">Event</a>
<i>o2</i>	The Second <a href="#">Event</a>

#### Returns

An integer containing the difference

Definition at line 386 of file Event.java.

### 3.8.3.2 compareTo()

```
int com.backend.Event.compareTo (
    Event o )
```

Compares the dates of two events.

#### Parameters

<i>o</i>	The Second <a href="#">Event</a>
----------	----------------------------------

#### Returns

An integer representing the difference

Definition at line 374 of file Event.java.

### 3.8.3.3 getDate()

```
Date com.backend.Event.getDate ( )
```

Gets the [Date](#).

#### Returns

The [Date](#)

Definition at line 254 of file Event.java.

### 3.8.3.4 getDay()

```
int com.backend.Event.getDay ( )
```

Gets the Day.

#### Returns

The current Day

Definition at line 222 of file Event.java.

### 3.8.3.5 getEventName()

```
String com.backend.Event.getEventName ( )
```

Gets the name of the [Event](#).

#### Returns

The Name of the [Event](#)

Definition at line 135 of file Event.java.

### 3.8.3.6 getHours()

```
double com.backend.Event.getHours ( )
```

Gets the length of the [Event](#).

#### Returns

The Length of the [Event](#)

Definition at line 156 of file Event.java.

### 3.8.3.7 getLocalDate()

```
LocalDate com.backend.Event.getLocalDate ( )
```

Gets the LocalDate (the new version of java.util.Date) of the current [Date](#).

#### Returns

the LocalDate

Definition at line 322 of file Event.java.

### 3.8.3.8 getMonth()

```
int com.backend.Event.getMonth ( )
```

Gets the Month.

#### Returns

The current Month

Definition at line 204 of file Event.java.

### 3.8.3.9 getMonthsWithYear()

```
static String [] com.backend.Event.getMonthsWithYear (
    Date startDateIn,
    Date endDateIn ) [static]
```

Gets an Array of all the Months in the range between two months, including the start and the end.

This is used for labeling graphs, particularly the Line Graph in the Individual [Student](#) Reports.

**Parameters**

<i>start↵ DateIn</i>	The Start <a href="#">Date</a> in the Range
<i>endDateIn</i>	The End <a href="#">Date</a> in the Range

**Returns**

An Array of Strings containing all Labels of the dates in the range.

Definition at line 80 of file Event.java.

**3.8.3.10 getTotalHours()**

```
static double com.backend.Event.getTotalHours (
    List< Event > eventList ) [static]
```

Gets the total hours in a list of Events, especially useful when calculating the hours a student achieved in a certain range.

**Parameters**

<i>eventList</i>	The List of Events
------------------	--------------------

**Returns**

the total time spent volunteering

Definition at line 121 of file Event.java.

**3.8.3.11 getYear()**

```
int com.backend.Event.getYear ( )
```

Gets the Year.

**Returns**

the current Year

Definition at line 186 of file Event.java.

**3.8.3.12 monthsInRange()**

```
static int com.backend.Event.monthsInRange (
    Date startDate,
    Date endDate ) [static]
```

Gets the number of months in a range between two months.

## Parameters

<i>startDate</i>	The start date of the Range
<i>endDate</i>	The end date of the range

## Returns

the range

Definition at line 110 of file Event.java.

**3.8.3.13 setDate()** [1/4]

```
void com.backend.Event.setDate (
    Date date )
```

Sets the date given another [Date](#).

## Parameters

<i>date</i>	The new <a href="#">Date</a>
-------------	------------------------------

Definition at line 304 of file Event.java.

**3.8.3.14 setDate()** [2/4]

```
void com.backend.Event.setDate (
    int year,
    int month,
    int day )
```

Sets the date, changing the database if necessary.

## Parameters

<i>year</i>	The New Year
<i>month</i>	The New Month
<i>day</i>	The New Day

Definition at line 242 of file Event.java.

### 3.8.3.15 setDate() [3/4]

```
void com.backend.Event.setDate (
    LocalDate localDate )
```

Sets the [Date](#) of the [Event](#) based on the `LocalDate`, changing the Database if necessary.

#### Parameters

<i>localDate</i>	The New <a href="#">Date</a>
------------------	------------------------------

Definition at line 289 of file `Event.java`.

### 3.8.3.16 setDate() [4/4]

```
void com.backend.Event.setDate (
    String date )
```

Sets the [Date](#) using a `String` of the new [Date](#), depending on the format of the `String`.

Updates Database if necessary

#### Parameters

<i>date</i>	The new <a href="#">Date</a> as a <code>String</code>
-------------	---

Definition at line 264 of file `Event.java`.

### 3.8.3.17 setDay()

```
void com.backend.Event.setDay (
    int day )
```

Changes the Day.

#### Parameters

<i>day</i>	The new Day
------------	-------------

Definition at line 231 of file `Event.java`.

### 3.8.3.18 setEventName()

```
void com.backend.Event.setEventName (
    String eventName )
```

Sets the name of the event, changing in the Database if necessary.

#### Parameters

<i>eventName</i>	The new Name of the <a href="#">Event</a>
------------------	---

Definition at line 144 of file Event.java.

### 3.8.3.19 setHours() [1/2]

```
void com.backend.Event.setHours (
    double hours )
```

Changes the length of the [Event](#), updating database if necessary.

#### Parameters

<i>hours</i>	The new Length of the <a href="#">Event</a>
--------------	---

Definition at line 165 of file Event.java.

### 3.8.3.20 setHours() [2/2]

```
void com.backend.Event.setHours (
    String hours )
```

Changes the length of the [Event](#) given a String, changing backend if necessary.

#### Parameters

<i>hours</i>	The new Length (as a String)
--------------	------------------------------

Definition at line 177 of file Event.java.

### 3.8.3.21 setMonth()

```
void com.backend.Event.setMonth (
    int month )
```

Changes the Month.



**Parameters**

<i>month</i>	The new Month
--------------	---------------

Definition at line 213 of file Event.java.

**3.8.3.22 setYear()**

```
void com.backend.Event.setYear (
    int year )
```

Changes the Year.

**Parameters**

<i>year</i>	The New Year
-------------	--------------

Definition at line 195 of file Event.java.

**3.8.3.23 toString()**

```
String com.backend.Event.toString ( )
```

Formats the [Event](#) in an easy-to-read manner.

**Returns**

A String with the event Information

Reimplemented from [com.backend.Student](#).

Definition at line 353 of file Event.java.

**3.8.3.24 updateEvent()**

```
void com.backend.Event.updateEvent (
    Event oldEvent,
    Event newEvent )
```

Updates the event with all the new values in the backend database.

## Parameters

<i>oldEvent</i>	the Old <a href="#">Event</a>
<i>newEvent</i>	the New <a href="#">Event</a>

Definition at line 363 of file Event.java.

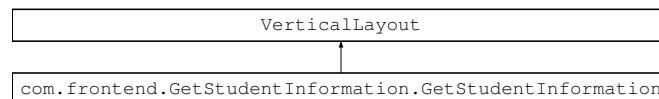
The documentation for this class was generated from the following file:

- Event.java

### 3.9 com.frontend.GetStudentInformation.GetStudentInformation Class Reference

Gets All Student's General Information and Allows Editing.

Inheritance diagram for com.frontend.GetStudentInformation.GetStudentInformation:



#### Public Member Functions

- [GetStudentInformation](#) ()  
*The Constructor, just runs the Main Table Method.*
- void [mainTable](#) ()  
*Creates the main table, which shows student data and allows editing of it.*
- Button [deleteButton](#) ([Student](#) student)  
*Creates the delete button, which deletes a student if clicked.*
- Button [expandButton](#) ([StudentData](#) student)  
*Creates the expand button, which reveals more specific student information, namely the events.*

#### 3.9.1 Detailed Description

Gets All Student's General Information and Allows Editing.

Also Allows viewing more specific events.

Definition at line 34 of file GetStudentInformation.java.

#### 3.9.2 Member Function Documentation

##### 3.9.2.1 deleteButton()

```
Button com.frontend.GetStudentInformation.GetStudentInformation.deleteButton (
    Student student )
```

Creates the delete button, which deletes a student if clicked.

## Parameters

<i>student</i>	The Student to create a delete button for
----------------	---

## Returns

The Delete Button

Definition at line 82 of file GetStudentInformation.java.

### 3.9.2.2 expandButton()

```
Button com.frontend.GetStudentInformation.GetStudentInformation.expandButton (
    StudentData student )
```

Creates the expand button, which reveals more specific student information, namely the events.

This method creates a dialog box (as a notification) which has all of a student's events and data, allowing for a more in-depth review of a student

## Parameters

<i>student</i>	The Student to Expand
----------------	-----------------------

## Returns

The Expand Button

Definition at line 118 of file GetStudentInformation.java.

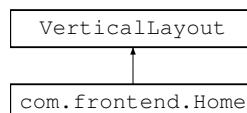
The documentation for this class was generated from the following file:

- GetStudentInformation.java

## 3.10 com.frontend.Home Class Reference

[Home](#) is the home view for the App.

Inheritance diagram for com.frontend.Home:



## Public Member Functions

- [Home](#) ()  
*Creates the [Home](#) Screen.*

### 3.10.1 Detailed Description

[Home](#) is the home view for the App.

This view has a brief introduction to the project and allows navigation to every other part of the app.

#### Author

Shourya Bansal

Definition at line 31 of file Home.java.

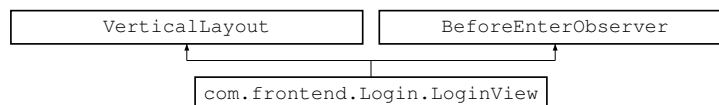
The documentation for this class was generated from the following file:

- Home.java

## 3.11 com.frontend.Login.LoginView Class Reference

A Class that allows logging in to the system.

Inheritance diagram for com.frontend.Login.LoginView:



## Public Member Functions

- [LoginView](#) ()  
*Sets up the Login View and adds it to the its page.*
- void [beforeEnter](#) (BeforeEnterEvent beforeEnterEvent)  
*Turns on security, preventing access to the rest of the app before login is complete.*

### 3.11.1 Detailed Description

A Class that allows logging in to the system.

Huge potential for security, currently not using its full capabilities for demonstration purposes.

Definition at line 20 of file LoginView.java.

### 3.11.2 Member Function Documentation

#### 3.11.2.1 beforeEnter()

```
void com.frontend.Login.LoginView.beforeEnter (
    BeforeEnterEvent beforeEnterEvent )
```

Turns on security, preventing access to the rest of the app before login is complete.

## Parameters

<i>beforeEnterEvent</i>	A lambda event that is used by the login view to validate if the app should be open or remain closed
-------------------------	--

Definition at line 52 of file LoginView.java.

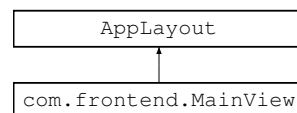
The documentation for this class was generated from the following file:

- LoginView.java

## 3.12 com.frontend.MainView Class Reference

[MainView](#) is the main view of the entire app.

Inheritance diagram for com.frontend.MainView:



### Public Member Functions

- [MainView](#) ()  
*Creates the Main View that formats the rest of the App.*

### Static Public Member Functions

- static Tabs [getTabs](#) ()  
*Generates the tabs for the Navbar.*

### Static Public Attributes

- static final Tab **HOME\_TAB** = createTab(VaadinIcon.HOME, "Home", Home.class)
- static final Tab **ADD\_STUDENT\_TAB** = createTab(VaadinIcon.FILE\_ADD, "Add a Student", CreateStudent.class)
- static final Tab **ADD\_HOURS\_TAB** = createTab(VaadinIcon.EDIT, "Add Hours to Student", AddHours.class)
- static final Tab **VIEW\_EDIT\_TAB** = createTab(VaadinIcon.EYE, "View and Edit Students", GetStudentInformation.class)
- static final Tab **REPORT\_TAB** = createTab(VaadinIcon.RECORDS, "Generate [Reports](#)", Reports.class)
- static final Tab **DOC\_TAB** = createTab(VaadinIcon.QUESTION, "Documentation and FAQs", Documentation.class)
- static final Tabs **tabs** = [getTabs](#)()

### 3.12.1 Detailed Description

[MainView](#) is the main view of the entire app.

Most pages use [MainView](#) as the base to format themselves, keeping everything uniform

#### Author

Shourya Bansal

Definition at line 49 of file MainView.java.

### 3.12.2 Member Function Documentation

#### 3.12.2.1 `getTabs()`

```
static Tabs com.frontend.MainView.getTabs ( ) [static]
```

Generates the tabs for the Navbar.

#### Returns

The Tabs for the navBar

Definition at line 111 of file MainView.java.

The documentation for this class was generated from the following file:

- MainView.java

## 3.13 `com.backend.MySQLMethods` Class Reference

This Class will hold all the methods to allow manipulation of the Data in the Database.

## Static Public Member Functions

- static void `createDatabase` ()  
*This method runs at the start of any Query.*
- static void `createTable` ()  
*This method creates the main table which would store student information and current hours.*
- static void `createPasswordTable` ()  
*This method creates the password Table which will be implemented in the future to allow multiple users to access different layers of security.*
- static void `createStudentTable` (String firstName, String lastName, int studentID)  
*This method creates a specific student table.*
- static void `createStudent` (String firstName, String lastName, int studentID, short grade, String community← ServiceCategory, String email, short yearsDone)  
*This method makes it very easy to create a student in the main tracker table.*
- static void `addStudentHours` (Student student, String eventName, double hours, int year, int month, int day) throws Exception  
*This adds events and hours into a students unique table.*
- static void `addUserPass` (String username, String hashedPassword) throws Exception  
*This stores the password in a database.*
- static String `selectPass` (String username)  
*Gets the hashed password given a username.*
- static int `selectTrackerInt` (String firstName, String lastName, int studentID, String data)  
*This allows someone to get specific integer-type data from the Main Table.*
- static String `selectTrackerString` (String firstName, String lastName, int studentID, String data)  
*This allows someone to get Specific String-type data from the Main table.*
- static double `selectTrackerDouble` (String firstName, String lastName, int studentID, String data)  
*This allows someone to get specific Double-Type data from the Main Table.*
- static short `selectTrackerShort` (String firstName, String lastName, int studentID, String data)  
*This allows someone to get specific short-type data from the Main Table.*
- static `StudentData` `selectTrackerAsStudent` (Student student)  
*This method gets the data of a student with the information listed, but returns it all in its special class as a `StudentData` object.*
- static List< `Student` > `getStudents` ()  
*Generates a list of all students being tracked.*
- static List< `StudentData` > `getStudentData` ()  
*Generates a List of All Students being tracked, as well as their hours and other information.*
- static List< `Event` > `selectStudentEventsAsEvent` (String firstName, String lastName, int studentID)  
*This allows access to all of a student's events.*
- static List< `Event` > `selectStudentEventsAsEvent` (Student student)  
*This allows access to all of a student's events.*
- static List< `Event` > `selectStudentEventsInRange` (Student student, Date startDate)  
*This is mainly used in reports, where we want to get dates in a certain range.*
- static List< `StudentData` > `getStudentData` (String range)  
*A List of All Students with their data, but limiting hours to only those earned in a specified range.*
- static double `getHoursWeek` (Student student)  
*Gets the hours a student earned in the past week.*
- static double `getHoursMonth` (Student student)  
*Gets the number of hours a student earned in a Month.*
- static double `getHoursYear` (Student student)  
*Gets the hours a student earned in a year.*
- static double `getHoursAll` (Student student)  
*Gets the the total hours a student has earned their entire time in FBLA.*

- static void `updateTracker` (String firstName, String lastName, int studentID, String dataType, String newData) throws Exception  
*This updates a value in the Main Table.*
- static void `updateFirstName` (String firstName, String lastName, int studentID, String newFirstName)  
*Updates the first name of a `Student` in case that needs to be changed.*
- static void `updateLastName` (String firstName, String lastName, int studentID, String newLastName)  
*Updates the last Name of a `Student` in case that needs to be done.*
- static void `updateStudentID` (String firstName, String lastName, int studentID, int newStudentID)  
*Updates the `Student` ID of a `Student` just in case that changes.*
- static void `updateTracker` (`Student` initialStudent, `StudentData` newData) throws Exception  
*Updates the main, general tracking table.*
- static void `updateEvent` (`Student` student, `Event` oldEvent, `Event` newEvent)  
*Updates a student's event.*
- static void `delete` (`Student` student)  
*This method deletes a student and his or her respective data table.*
- static void `delete` (`Event` event)  
*Deletes an `Event`.*
- static String `makeName` (String firstName, String lastName, int studentID)  
*This method would be used to make student table names in MySQL (firstName\_lastName\_studentID)*
- static String `makeName` (`Student` student)  
*Formats the name of a student.*
- static int `numberOfRows` (String tableName) throws Exception  
*Returns the number of rows in a table.*
- static void `updateToDate` (String firstName, String lastName, int studentID)  
*Updates the date in the tracker to the current date for lastEdited.*
- static double `round` (double input)  
*Rounds the double to the nearest hundredth.*
- static void `setUp` ()  
*Runs the necessary methods to set up the tables in case this is the first time the app is running.*

## Static Public Attributes

- static final String `pwTableName` = "passwords"  
*The Name of the Passwords Table.*
- static final String `DATABASE_NAME` = "student\_data"  
*The Name of the Database.*

### 3.13.1 Detailed Description

This Class will hold all the methods to allow manipulation of the Data in the Database.

This uses MySQL to actively store and edit data.

#### Author

Shourya Bansal

Definition at line 16 of file MySQLMethods.java.



## 3.13.2 Member Function Documentation

### 3.13.2.1 addStudentHours()

```
static void com.backend.MySQLMethods.addStudentHours (
    Student student,
    String eventName,
    double hours,
    int year,
    int month,
    int day ) throws Exception [static]
```

This adds events and hours into a students unique table.

At the same time, this updates the hours in the main table

#### Parameters

<i>student</i>	The student getting hours
<i>eventName</i>	The Name of the <a href="#">Event</a> Completed
<i>hours</i>	The Length of the <a href="#">Event</a>
<i>year</i>	The Year the <a href="#">Event</a> Was Done
<i>month</i>	The Month of the <a href="#">Event</a>
<i>day</i>	The Day of the <a href="#">Event</a>

#### Exceptions

<i>Exception</i>	This is throws in case the Database is not found
------------------	--

Definition at line 258 of file MySQLMethods.java.

### 3.13.2.2 addUserPass()

```
static void com.backend.MySQLMethods.addUserPass (
    String username,
    String hashedPassword ) throws Exception [static]
```

This stores the password in a database.

However, to secure them, they use a hashing function (explained in the documentation) which securely converts the password into an impenetrable String. It is a one way function, thus stopping anyone from figuring out what the password is.

#### Parameters

<i>username</i>	the Username of the user being added
<i>hashedPassword</i>	the password after hash encryption

### Exceptions

<i>Exception</i>	This is throws in case the Database is not found
------------------	--

Definition at line 306 of file MySQLMethods.java.

#### 3.13.2.3 createDatabase()

```
static void com.backend.MySQLMethods.createDatabase ( ) [static]
```

This method runs at the start of any Query.

It checks if the database has already been created and, if it hasn't, creates it.

Definition at line 64 of file MySQLMethods.java.

#### 3.13.2.4 createStudent()

```
static void com.backend.MySQLMethods.createStudent (
    String firstName,
    String lastName,
    int studentID,
    short grade,
    String communityServiceCategory,
    String email,
    short yearsDone ) [static]
```

This method makes it very easy to create a student in the main tracker table.

#### Parameters

<i>firstName</i>	The First Name of the <a href="#">Student</a>
<i>lastName</i>	The Last Name of the <a href="#">Student</a>
<i>studentID</i>	The <a href="#">Student's</a> <a href="#">Student</a> ID Number
<i>grade</i>	The <a href="#">Student's</a> Grade Number
<i>communityServiceCategory</i>	The Category of Award Aimed for by the <a href="#">Student</a>
<i>email</i>	The <a href="#">Student's</a> Email Address
<i>yearsDone</i>	The number of years of FBLA Completed

Definition at line 208 of file MySQLMethods.java.

#### 3.13.2.5 createStudentTable()

```
static void com.backend.MySQLMethods.createStudentTable (
    String firstName,
```

```
String lastName,  
int studentID ) [static]
```

This method creates a specific student table.

This would allow more specific storage of data, such as data of specific events and such and make it much easier to analyze a students data.

#### Parameters

<i>firstName</i>	This is the student's First Name
<i>lastName</i>	This is the student's Last Name
<i>studentID</i>	The <a href="#">Student's Student</a> ID Number

Definition at line 166 of file MySQLMethods.java.

#### 3.13.2.6 delete() [1/2]

```
static void com.backend.MySQLMethods.delete (  
    Event event ) [static]
```

Deletes an [Event](#).

#### Parameters

<i>event</i>	the <a href="#">Event</a>
--------------	---------------------------

Definition at line 1208 of file MySQLMethods.java.

#### 3.13.2.7 delete() [2/2]

```
static void com.backend.MySQLMethods.delete (  
    Student student ) [static]
```

This method deletes a student and his or her respective data table.

#### Parameters

<i>student</i>	The Deleted <a href="#">Student</a>
----------------	-------------------------------------

Definition at line 1175 of file MySQLMethods.java.

### 3.13.2.8 getHoursAll()

```
static double com.backend.MySQLMethods.getHoursAll (
    Student student ) [static]
```

Gets the the total hours a student has earned their entire time in FBLA.

#### Parameters

<i>student</i>	The <a href="#">Student</a> being tracked
----------------	---

#### Returns

The total hours a student earned

Definition at line 899 of file MySQLMethods.java.

### 3.13.2.9 getHoursMonth()

```
static double com.backend.MySQLMethods.getHoursMonth (
    Student student ) [static]
```

Gets the number of hours a student earned in a Month.

#### Parameters

<i>student</i>	The <a href="#">Student</a> being tracked
----------------	---

#### Returns

The hours the student earned in a month

Definition at line 835 of file MySQLMethods.java.

### 3.13.2.10 getHoursWeek()

```
static double com.backend.MySQLMethods.getHoursWeek (
    Student student ) [static]
```

Gets the hours a student earned in the past week.

#### Parameters

<i>student</i>	The <a href="#">Student</a> being found
----------------	---

**Returns**

The hours the student earned in a week

Definition at line 803 of file MySQLMethods.java.

**3.13.2.11 getHoursYear()**

```
static double com.backend.MySQLMethods.getHoursYear (
    Student student ) [static]
```

Gets the hours a student earned in a year.

**Parameters**

<i>student</i>	The student being tracked
----------------	---------------------------

**Returns**

The hours earned in a year

Definition at line 867 of file MySQLMethods.java.

**3.13.2.12 getStudentData() [1/2]**

```
static List<StudentData> com.backend.MySQLMethods.getStudentData ( ) [static]
```

Generates a List of All Students being tracked, as well as their hours and other information.

**Returns**

A List of [StudentData](#) Objects

Definition at line 632 of file MySQLMethods.java.

**3.13.2.13 getStudentData() [2/2]**

```
static List<StudentData> com.backend.MySQLMethods.getStudentData (
    String range ) [static]
```

A List of All Students with their data, but limiting hours to only those earned in a specified range.

**Parameters**

<i>range</i>	The range of time from where hours should be received
--------------	---

**Returns**

A list of [StudentData](#) objects

Definition at line 773 of file MySQLMethods.java.

**3.13.2.14 getStudents()**

```
static List<Student> com.backend.MySQLMethods.getStudents ( ) [static]
```

Generates a list of all students being tracked.

**Returns**

a List of Students

Definition at line 593 of file MySQLMethods.java.

**3.13.2.15 makeName() [1/2]**

```
static String com.backend.MySQLMethods.makeName (
    String firstName,
    String lastName,
    int studentID ) [static]
```

This method would be used to make student table names in MySQL (*firstName\_lastName\_studentID*)

**Parameters**

<i>firstName</i>	First Name
<i>lastName</i>	Last Name
<i>studentID</i>	<a href="#">Student</a> ID

**Returns**

formatted Name

Definition at line 1250 of file MySQLMethods.java.

### 3.13.2.16 makeName() [2/2]

```
static String com.backend.MySQLMethods.makeName (  
    Student student ) [static]
```

Formats the name of a student.

#### Parameters

<i>student</i>	The <a href="#">Student</a> whose name is being formatted
----------------	---

#### Returns

the name in the format of firstName\_lastName\_studentID

Definition at line 1260 of file MySQLMethods.java.

### 3.13.2.17 numberOfRows()

```
static int com.backend.MySQLMethods.numberOfRows (  
    String tableName ) throws Exception [static]
```

Returns the number of rows in a table.

This is extremely useful in knowing the number of times to loop a certain event.

#### Parameters

<i>tableName</i>	the name of the table being counted
------------------	-------------------------------------

#### Returns

the number of rows

#### Exceptions

<i>Exception</i>	for SQL Errors
------------------	----------------

Definition at line 1285 of file MySQLMethods.java.

### 3.13.2.18 round()

```
static double com.backend.MySQLMethods.round (  
    double input ) [static]
```

Rounds the double to the nearest hundredth.

**Parameters**

<i>input</i>	the double being rounded
--------------	--------------------------

**Returns**

the rounded double

Definition at line 1332 of file MySQLMethods.java.

**3.13.2.19 selectPass()**

```
static String com.backend.MySQLMethods.selectPass (  
    String username ) [static]
```

Gets the hashed password given a username.

**Parameters**

<i>username</i>	the Username for which a password is being found
-----------------	--

**Returns**

the hashed password

Definition at line 333 of file MySQLMethods.java.

**3.13.2.20 selectStudentEventsAsEvent() [1/2]**

```
static List<Event> com.backend.MySQLMethods.selectStudentEventsAsEvent (  
    String firstName,  
    String lastName,  
    int studentID ) [static]
```

This allows access to all of a student's events.

**Parameters**

<i>firstName</i>	the student's first name
<i>lastName</i>	the student's last name
<i>studentID</i>	The <a href="#">Student's Student</a> ID Number



**Returns**

an array of Strings containing all of the Events and their data

Definition at line 673 of file MySQLMethods.java.

**3.13.2.21 selectStudentEventsAsEvent() [2/2]**

```
static List<Event> com.backend.MySQLMethods.selectStudentEventsAsEvent (
    Student student ) [static]
```

This allows access to all of a student's events.

**Parameters**

<i>student</i>	the selected student
----------------	----------------------

**Returns**

an array of Strings containing all of the Events and their data

Definition at line 720 of file MySQLMethods.java.

**3.13.2.22 selectStudentEventsInRange()**

```
static List<Event> com.backend.MySQLMethods.selectStudentEventsInRange (
    Student student,
    Date startDate ) [static]
```

This is mainly used in reports, where we want to get dates in a certain range.

Thus, this method gets a start date and adds a limitation on what to select from the student table.

**Parameters**

<i>student</i>	The <a href="#">Student</a> whose events we need
<i>startDate</i>	The Start <a href="#">Date</a> of the Events

**Returns**

A List containing the events being used

Definition at line 732 of file MySQLMethods.java.

### 3.13.2.23 selectTrackerAsStudent()

```
static StudentData com.backend.MySQLMethods.selectTrackerAsStudent (
    Student student ) [static]
```

This method gets the data of a student with the information listed, but returns it all in its special class as a [StudentData](#) object.

#### Parameters

<i>student</i>	The <a href="#">Student</a> who is being queried
----------------	--

#### Returns

A [StudentData](#) Object with all of the [Student](#)'s Information

Definition at line 545 of file MySQLMethods.java.

### 3.13.2.24 selectTrackerDouble()

```
static double com.backend.MySQLMethods.selectTrackerDouble (
    String firstName,
    String lastName,
    int studentID,
    String data ) [static]
```

This allows someone to get specific Double-Type data from the Main Table.

#### Parameters

<i>firstName</i>	The <a href="#">Student</a> 's First Name
<i>lastName</i>	The <a href="#">Student</a> 's Last name
<i>studentID</i>	The <a href="#">Student</a> 's <a href="#">Student</a> ID Number
<i>data</i>	The data field being accessed

#### Returns

A double containing the data

Definition at line 465 of file MySQLMethods.java.

### 3.13.2.25 selectTrackerInt()

```
static int com.backend.MySQLMethods.selectTrackerInt (
    String firstName,
```

```
String lastName,  
int studentID,  
String data ) [static]
```

This allows someone to get specific integer-type data from the Main Table.

**Parameters**

<i>firstName</i>	The <a href="#">Student</a> 's first name
<i>lastName</i>	the student's last name
<i>studentID</i>	The <a href="#">Student</a> 's <a href="#">Student</a> ID Number
<i>data</i>	the data field being accessed

**Returns**

an integer containing the data

Definition at line 378 of file MySQLMethods.java.

**3.13.2.26 selectTrackerShort()**

```
static short com.backend.MySQLMethods.selectTrackerShort (
    String firstName,
    String lastName,
    int studentID,
    String data ) [static]
```

This allows someone to get specific short-type data from the Main Table.

**Parameters**

<i>firstName</i>	the student's first name
<i>lastName</i>	the student's last name
<i>studentID</i>	The <a href="#">Student</a> 's <a href="#">Student</a> ID Number
<i>data</i>	the data field being accessed

**Returns**

a short containing the data

Definition at line 506 of file MySQLMethods.java.

**3.13.2.27 selectTrackerString()**

```
static String com.backend.MySQLMethods.selectTrackerString (
    String firstName,
    String lastName,
    int studentID,
    String data ) [static]
```

This allows someone to get Specific String-type data from the Main table.

## Parameters

<i>firstName</i>	The student's First Name
<i>lastName</i>	The student's Last Name
<i>studentID</i>	The <a href="#">Student's Student</a> ID Number
<i>data</i>	The data field being accessed

## Returns

the String Type data

Definition at line 419 of file MySQLMethods.java.

**3.13.2.28 updateEvent()**

```
static void com.backend.MySQLMethods.updateEvent (
    Student student,
    Event oldEvent,
    Event newEvent ) [static]
```

Updates a student's event.

## Parameters

<i>student</i>	The student being updated
<i>oldEvent</i>	The old event information
<i>newEvent</i>	The new event details

Definition at line 1132 of file MySQLMethods.java.

**3.13.2.29 updateFirstName()**

```
static void com.backend.MySQLMethods.updateFirstName (
    String firstName,
    String lastName,
    int studentID,
    String newFirstName ) [static]
```

Updates the first name of a [Student](#) in case that needs to be changed.

## Parameters

<i>firstName</i>	The Old First Name of the <a href="#">Student</a>
<i>lastName</i>	The <a href="#">Student's</a> last Name
<i>studentID</i>	The <a href="#">Student's Student</a> ID Number
<i>newFirstName</i>	The <a href="#">Student's</a> new First Name

Definition at line 981 of file MySQLMethods.java.

### 3.13.2.30 updateLastName()

```
static void com.backend.MySQLMethods.updateLastName (
    String firstName,
    String lastName,
    int studentID,
    String newLastName ) [static]
```

Updates the last Name of a [Student](#) in case that needs to be done.

#### Parameters

<i>firstName</i>	The First name of the <a href="#">Student</a>
<i>lastName</i>	The Old Last name of the <a href="#">Student</a>
<i>studentID</i>	The <a href="#">Student's</a> <a href="#">Student</a> ID Number
<i>newLastName</i>	The New Last name of the <a href="#">Student</a>

Definition at line 1017 of file MySQLMethods.java.

### 3.13.2.31 updateStudentID()

```
static void com.backend.MySQLMethods.updateStudentID (
    String firstName,
    String lastName,
    int studentID,
    int newStudentID ) [static]
```

Updates the [Student](#) ID of a [Student](#) just in case that changes.

#### Parameters

<i>firstName</i>	The <a href="#">Student's</a> First Name
<i>lastName</i>	The <a href="#">Student's</a> Last Name
<i>studentID</i>	The <a href="#">Student's</a> Old <a href="#">Student</a> ID
<i>newStudentID</i>	The <a href="#">Student's</a> New <a href="#">Student</a> ID

Definition at line 1054 of file MySQLMethods.java.

### 3.13.2.32 updateToDate()

```
static void com.backend.MySQLMethods.updateToDate (
    String firstName,
```

```
String lastName,
int studentID ) [static]
```

Updates the date in the tracker to the current date for lastEdited.

#### Parameters

<i>firstName</i>	The First Name of the <a href="#">Student</a> to know who to update
<i>lastName</i>	The Last Name of the <a href="#">Student</a> to Update
<i>studentID</i>	The <a href="#">Student's Student</a> ID Number

Definition at line 1317 of file MySQLMethods.java.

#### 3.13.2.33 updateTracker() [1/2]

```
static void com.backend.MySQLMethods.updateTracker (
    String firstName,
    String lastName,
    int studentID,
    String dataType,
    String newData ) throws Exception [static]
```

This updates a value in the Main Table.

#### Parameters

<i>firstName</i>	the student's first name
<i>lastName</i>	the student's last name
<i>studentID</i>	The <a href="#">Student's Student</a> ID Number
<i>dataType</i>	the data field being changed
<i>newData</i>	a new data value (in string form) for the location

#### Exceptions

<i>Exception</i>	for SQL Errors
------------------	----------------

Definition at line 934 of file MySQLMethods.java.

#### 3.13.2.34 updateTracker() [2/2]

```
static void com.backend.MySQLMethods.updateTracker (
    Student initialStudent,
    StudentData newData ) throws Exception [static]
```

Updates the main, general tracking table.

There can be no changes in First Name, Last Name, or [Student](#) ID in these two pieces of data for the change to work successfully

## Parameters

<i>initialStudent</i>	The <a href="#">Student</a> Being updated
<i>newData</i>	The Data being inserted into the student

## Exceptions

<i>Exception</i>	For MySQL Exceptions
------------------	----------------------

Definition at line 1093 of file MySQLMethods.java.

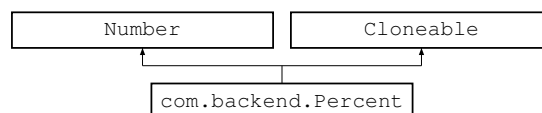
The documentation for this class was generated from the following file:

- MySQLMethods.java

### 3.14 com.backend.Percent Class Reference

A Class to Use for [Percent](#) Data Types.

Inheritance diagram for com.backend.Percent:



#### Public Member Functions

- [Percent](#) (int current, int max)  
*A Constructor that takes in integers and generates a percentage based on them.*
- [Percent](#) (double current, double max)  
*A Constructor that takes in doubles and generates a percentage based on them.*
- double [getPercent](#) ()  
*Returns the [Percent](#) as a Double.*
- String [toString](#) ()  
*Allows Printing as a String.*
- int [intValue](#) ()  
*Gets the integer value of the percentage.*
- long [longValue](#) ()  
*Gets the long value of the percentage.*
- float [floatValue](#) ()  
*Gets the float value of the percentage.*
- double [doubleValue](#) ()  
*Just returns the double value of the percentage.*



## Static Public Member Functions

- static double `round` (double toBeRounded)  
*Rounds to the nearest integer, as none of the used percentages require much precision.*

## Protected Member Functions

- `Percent clone ()` throws CloneNotSupportedException

### 3.14.1 Detailed Description

A Class to Use for `Percent` Data Types.

Definition at line 8 of file Percent.java.

### 3.14.2 Constructor & Destructor Documentation

#### 3.14.2.1 `Percent()` [1/2]

```
com.backend.Percent.Percent (
    int current,
    int max )
```

A Constructor that takes in integers and generates a percentage based on them.

It calculates percentage in the form  $\text{current} / \text{total}$

##### Parameters

<i>current</i>	The Current Value
<i>max</i>	The Total Value

Definition at line 23 of file Percent.java.

#### 3.14.2.2 `Percent()` [2/2]

```
com.backend.Percent.Percent (
    double current,
    double max )
```

A Constructor that takes in doubles and generates a percentage based on them.

It calculates percentage in the form  $\text{current} / \text{total}$

**Parameters**

<i>current</i>	The Current Value
<i>max</i>	The Total Value

Definition at line 36 of file Percent.java.

### 3.14.3 Member Function Documentation

#### 3.14.3.1 doubleValue()

```
double com.backend.Percent.doubleValue ( )
```

Just returns the double value of the percentage.

**Returns**

the double value of the percentage

Definition at line 106 of file Percent.java.

#### 3.14.3.2 floatValue()

```
float com.backend.Percent.floatValue ( )
```

Gets the float value of the percentage.

**Returns**

the float value of the percentage

Definition at line 96 of file Percent.java.

#### 3.14.3.3 getPercent()

```
double com.backend.Percent.getPercent ( )
```

Returns the [Percent](#) as a Double.

**Returns**

the [Percent](#) as a Double

Definition at line 56 of file Percent.java.

### 3.14.3.4 intValue()

```
int com.backend.Percent.intValue ( )
```

Gets the integer value of the percentage.

#### Returns

the integer value of the percentage

Definition at line 76 of file Percent.java.

### 3.14.3.5 longValue()

```
long com.backend.Percent.longValue ( )
```

Gets the long value of the percentage.

#### Returns

the long value of the percentage

Definition at line 86 of file Percent.java.

### 3.14.3.6 round()

```
static double com.backend.Percent.round (
    double toBeRounded ) [static]
```

Rounds to the nearest integer, as none of the used percentages require much precision.

#### Parameters

<i>toBeRounded</i>	the double before rounding
--------------------	----------------------------

#### Returns

The rounded double

Definition at line 47 of file Percent.java.

### 3.14.3.7 toString()

```
String com.backend.Percent.toString ( )
```

Allows Printing as a String.

#### Returns

A String with the correct formatting

Definition at line 66 of file Percent.java.

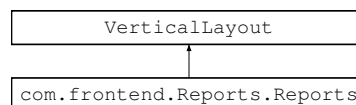
The documentation for this class was generated from the following file:

- Percent.java

## 3.15 com.frontend.Reports.Reports Class Reference

The main class for Generating [Reports](#).

Inheritance diagram for com.frontend.Reports.Reports:



### Public Member Functions

- [Reports](#) ()  
*Creates a form which allows the option between generating a group and individual report.*
- void [generateIndividualReport](#) ()  
*Creates a form to configure an individual student report.*
- void [generateGroupReport](#) ()  
*Creates the form to Generate the group report.*
- void [individualReport](#) ([Student](#) student, [Date](#) startDate)  
*Using the given constraints, generate an individual report.*
- void [groupReport](#) (String option)  
*The group report after it has been configured.*

### Static Public Member Functions

- static Div [setText](#) (String header, String text)  
*This method returns a the pre-formatted data of a student.*
- static void [nextSection](#) (Board dataBoard)  
*Prints out two lines in the data board to symbolize the next section.*
- static Div [makeBulletList](#) (String label, List< [StudentData](#) > list)  
*Makes a bulleted list, primarily used for the inactive students list in the Group [Reports](#).*

### 3.15.1 Detailed Description

The main class for Generating [Reports](#).

Definition at line 36 of file Reports.java.

### 3.15.2 Member Function Documentation

#### 3.15.2.1 groupReport()

```
void com.frontend.Reports.Reports.groupReport (
    String option )
```

The group report after it has been configured.

##### Parameters

<i>option</i>	The range of time
---------------	-------------------

Definition at line 391 of file Reports.java.

#### 3.15.2.2 individualReport()

```
void com.frontend.Reports.Reports.individualReport (
    Student student,
    Date startingDate )
```

Using the given constraints, generate an individual report.

##### Parameters

<i>student</i>	The Student of the Report
<i>startingDate</i>	The report's starting date

Definition at line 265 of file Reports.java.

#### 3.15.2.3 makeBulletList()

```
static Div com.frontend.Reports.Reports.makeBulletList (
    String label,
    List< StudentData > list ) [static]
```

Makes a bulleted list, primarily used for the inactive students list in the Group [Reports](#).

**Parameters**

<i>label</i>	The label of the List
<i>list</i>	The bulleted items in the list

**Returns**

A Div containing the Bulleted List

Definition at line 124 of file Reports.java.

**3.15.2.4 nextSection()**

```
static void com.frontend.Reports.Reports.nextSection (
    Board dataBoard ) [static]
```

Prints out two lines in the data board to symbolize the next section.

**Parameters**

<i>dataBoard</i>	The databoard with the data
------------------	-----------------------------

Definition at line 89 of file Reports.java.

**3.15.2.5 setText()**

```
static Div com.frontend.Reports.Reports.setText (
    String header,
    String text ) [static]
```

This method returns a the pre-formatted data of a student.

This simplifies the job significantly when writing multiple pieces of data

**Parameters**

<i>header</i>	The heading text of the piece of data
<i>text</i>	The piece of data

**Returns**

A Div containing the formatted Student Data

Definition at line 76 of file Reports.java.

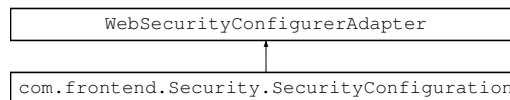
The documentation for this class was generated from the following file:

- Reports.java

## 3.16 com.frontend.Security.SecurityConfiguration Class Reference

Configures security with username, password, and different redirection options.

Inheritance diagram for com.frontend.Security.SecurityConfiguration:



### Public Member Functions

- UserDetailsService [userDetailsService](#) ()  
*Configures the Login Information and credentials to the app.*
- void [configure](#) (WebSecurity web)  
*Creates a configuration which allows usage of certain things while offline.*

### Protected Member Functions

- void [configure](#) (HttpSecurity http) throws Exception

#### 3.16.1 Detailed Description

Configures security with username, password, and different redirection options.

Definition at line 19 of file SecurityConfiguration.java.

#### 3.16.2 Member Function Documentation

##### 3.16.2.1 userDetailsService()

```
UserDetailsService com.frontend.Security.SecurityConfiguration.userDetailsService ( )
```

Configures the Login Information and credentials to the app.

Currently using temporary security for demonstration purposes.

Definition at line 53 of file SecurityConfiguration.java.

The documentation for this class was generated from the following file:

- SecurityConfiguration.java

## 3.17 com.frontend.Security.SecurityUtils Class Reference

A Class for Util methods in security.

### 3.17.1 Detailed Description

A Class for Util methods in security.

Definition at line 15 of file SecurityUtils.java.

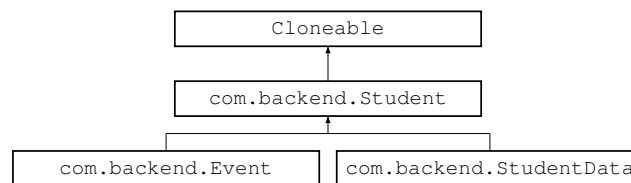
The documentation for this class was generated from the following file:

- SecurityUtils.java

## 3.18 com.backend.Student Class Reference

The General [Student](#) Class to Store Basic [Student](#) Information.

Inheritance diagram for com.backend.Student:



### Public Member Functions

- [Student](#) ()  
*Creates an empty student to allow using the other methods to manipulate it.*
- [Student](#) (String firstName, String lastName, int studentID)  
*Creates a student with the given variables.*
- [Student](#) (boolean createNewStudent)  
*Creates a new student with the createNewStudent variable, used to link a student with the Add [Student](#) feature in Select, ComboBox, and others.*
- String [getFirstName](#) ()  
*Gets the First.*
- void [setFirstName](#) (String firstName)  
*Changes the first name.*
- String [getLastName](#) ()  
*Gets the last name.*
- void [setLastName](#) (String lastName)  
*Changes the Last name.*
- String [getFullName](#) ()  
*Gets the full name in a specific format.*
- int [getStudentID](#) ()



- Gets the student ID.*
- void `setStudentID` (int studentID)
- Changes the StudentID.*
- `Student` `getStudent` ()
- Returns itself, mainly for the child methods.*
- void `setStudent` (`Student` newStudent)
- Changes all values of a student to be equal to a different student datatype.*
- boolean `getCreateNewStudent` ()
- Returns the boolean telling whether the object is meant to symbolize creating a new student.*
- `StudentData` `getStudentData` ()
- Uses the `MySQLMethods` to get the `StudentData` of the given `Student`.*
- void `delete` ()
- Deletes this student from the backend database.*
- String `toString` ()
- Converts The `Student` to a String.*

## Protected Member Functions

- `Student` `clone` () throws CloneNotSupportedException

### 3.18.1 Detailed Description

The General `Student` Class to Store Basic `Student` Information.

The Parent of `StudentData` and `Event`

Definition at line 10 of file Student.java.

### 3.18.2 Constructor & Destructor Documentation

#### 3.18.2.1 `Student()` [1/2]

```
com.backend.Student.Student (
    String firstName,
    String lastName,
    int studentID )
```

Creates a student with the given variables.

Parameters

<i>firstName</i>	The First Name
<i>lastName</i>	The Last Name
<i>studentID</i>	The <code>Student</code> ID

Definition at line 42 of file Student.java.

### 3.18.2.2 Student() [2/2]

```
com.backend.Student.Student (
    boolean createNewStudent )
```

Creates a new student with the createNewStudent variable, used to link a student with the Add [Student](#) feature in Select, ComboBox, and others.

#### Parameters

<i>createNewStudent</i>	The value of createNewStudent, typically when this method is used;
-------------------------	--

Definition at line 54 of file Student.java.

## 3.18.3 Member Function Documentation

### 3.18.3.1 getCreateNewStudent()

```
boolean com.backend.Student.getCreateNewStudent ( )
```

Returns the boolean telling whether the object is meant to symbolize creating a new student.

#### Returns

Whether the student symbolizes creating a new [Student](#)

Definition at line 148 of file Student.java.

### 3.18.3.2 getFirstName()

```
String com.backend.Student.getFirstName ( )
```

Gets the First.

#### Returns

the First Name

Definition at line 63 of file Student.java.

### 3.18.3.3 getFullName()

```
String com.backend.Student.getFullName ( )
```

Gets the full name in a specific format.

Uses the format firstName\_lastName\_studentID

#### Returns

The Name in said format

Definition at line 101 of file Student.java.

### 3.18.3.4 getLastName()

```
String com.backend.Student.getLastName ( )
```

Gets the last name.

#### Returns

the last name

Definition at line 81 of file Student.java.

### 3.18.3.5 getStudent()

```
Student com.backend.Student.getStudent ( )
```

Returns itself, mainly for the child methods.

#### Returns

itself, a [Student](#)

Definition at line 128 of file Student.java.

### 3.18.3.6 getStudentData()

```
StudentData com.backend.Student.getStudentData ( )
```

Uses the [MySQLMethods](#) to get the [StudentData](#) of the given [Student](#).

#### Returns

The full [StudentData](#) of the [Student](#)

Definition at line 157 of file Student.java.

### 3.18.3.7 `getStudentID()`

```
int com.backend.Student.getStudentID ( )
```

Gets the student ID.

#### Returns

the [Student](#) ID

Definition at line 110 of file Student.java.

### 3.18.3.8 `setFirstName()`

```
void com.backend.Student.setFirstName (
    String firstName )
```

Changes the first name.

#### Parameters

<i>firstName</i>	the new first name
------------------	--------------------

Definition at line 72 of file Student.java.

### 3.18.3.9 `setLastName()`

```
void com.backend.Student.setLastName (
    String lastName )
```

Changes the Last name.

#### Parameters

<i>lastName</i>	the new Last Name
-----------------	-------------------

Definition at line 90 of file Student.java.

### 3.18.3.10 `setStudent()`

```
void com.backend.Student.setStudent (
    Student newStudent )
```

Changes all values of a student to be equal to a different student datatype.

## Parameters

<i>newStudent</i>	The Values of the New <a href="#">Student</a>
-------------------	---

Definition at line 137 of file Student.java.

**3.18.3.11 setStudentID()**

```
void com.backend.Student.setStudentID (
    int studentID )
```

Changes the StudentID.

## Parameters

<i>studentID</i>	the new <a href="#">Student</a> ID
------------------	------------------------------------

Definition at line 119 of file Student.java.

**3.18.3.12 toString()**

```
String com.backend.Student.toString ( )
```

Converts The [Student](#) to a String.

## Returns

A String representing the [Student](#)

Reimplemented in [com.backend.Event](#).

Definition at line 180 of file Student.java.

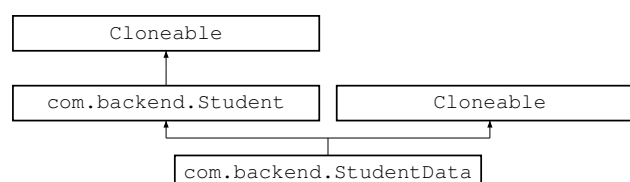
The documentation for this class was generated from the following file:

- Student.java

**3.19 com.backend.StudentData Class Reference**

This is a class which helps in running MySQL Methods and takes in all the values needed for the main student.

Inheritance diagram for com.backend.StudentData:



## Public Member Functions

- [StudentData](#) (boolean fromSelect)  
*A Constructor which allows the use of.*
- [StudentData](#) ()  
*A Simple Constructor for the creation of a [StudentData](#) Object.*
- short [getGrade](#) ()  
*Gets the student's current grade level.*
- void [setGrade](#) (int grade)  
*Sets the Grade Level of the [Student](#), changing the backend database if necessary.*
- void [setGrade](#) (String grade)  
*Sets the Grade Level of the [Student](#), changing the backend database if necessary.*
- int [getGradeInt](#) ()  
*Gets the student's current grade level.*
- double [getCommunityServiceHours](#) ()  
*Gets the Amount of Community Service Hours the [Student](#) has participated in.*
- void [setCommunityServiceHours](#) (double communityServiceHours)  
*Changes the number of Community Service Hours a student participated in, updating the backend if necessary.*
- void [setCommunityServiceHoursFromSelect](#) (double communityServiceHours)  
*Changes the number of Community Service Hours a student participated in and updating the backend.*
- void [setCommunityServiceHoursFromSelect](#) (String communityServiceHours)  
*Changes the number of Community Service Hours a student participated in and updating the backend.*
- String [getCommunityServiceCategory](#) ()  
*Gets the Community Service Award Category Goal of the [Student](#).*
- void [setCommunityServiceCategory](#) (String communityServiceCategoryIn)  
*Changes the Community Service Award Category, updating the backend if necessary.*
- int [getCommunityServiceCategoryInt](#) ()  
*Gets each Community Service Categories respective integer value.*
- String [getCurrentCommunityServiceCategory](#) ()  
*Gets the [Student](#)'s current Community Service Award Category depending on the hours the student has done.*
- int [getCurrentCommunityServiceCategoryInt](#) ()  
*Gets the student's current achieved category and returns its respective integer.*
- String [getEmail](#) ()  
*Gets the student's email address.*
- void [setEmail](#) (@NotNull final String email)  
*Changes the student's email address, updating the backend if necessary.*
- short [getYearsDone](#) ()  
*Gets the number of years a student has participated in FBLA.*
- void [setYearsDone](#) (@NotNull final short yearsDone)  
*Changes the value of the number of years a student has participated in FBLA, updating the database if necessary.*
- void [setYearsDone](#) (String yearsDone)  
*Changes the value of the number of years a student has participated in FBLA, updating the database if necessary.*
- boolean [isFreshman](#) ()
- void [setFreshman](#) (boolean freshman)
- boolean [isSophomore](#) ()
- void [setSophomore](#) (boolean sophomore)
- boolean [isJunior](#) ()
- void [setJunior](#) (boolean junior)
- boolean [isSenior](#) ()
- void [setSenior](#) (boolean senior)
- [Date](#) [getLastEdited](#) ()

- Gets the date any of a [Student](#)'s information was last edited.*

  - void [setLastEdited](#) (String lastEdited)

*Changes the last-edited date of a [Student](#)'s information.*
- boolean [achievedGoal](#) ()

*Gets whether or not a student achieved their Community Service Award Category Goal.*
- void [createStudent](#) ()

*Creates this student and adds it to database.*
- void [setCommunityServiceHoursTemp](#) (double communityServiceHours)

*Temporarily sets the community Service hours of a student, typically used when looking at the hours in a range.*
- boolean [isActive](#) ()

*Returns whether or not the student has any hours.*
- void [setFirstName](#) (@NotNull final String firstName)

*Changes the [Student](#)'s First Name, checking to see if the backend database must be changes as well.*
- void [setLastName](#) (@NotNull final String lastName)

*Changes the last name of a student and updates the database if necessary.*
- void [setStudentID](#) (@NotNull final int studentID)

*Changes the StudentID of a student, updating the database if necessary.*
- void [setStudentID](#) (String studentID)

*Changes the StudentID of a student, updating the database if necessary.*

### Static Public Member Functions

- static String [getAverageCategory](#) (List< [StudentData](#) > studentDataList)

*Calculates the average achieved category in a list of students.*
- static String [getAverageGoal](#) (List< [StudentData](#) > studentDataList)

*Calculates the Average Category Goal of a List of Students.*
- static String [getActiveCategory](#) (List< [StudentData](#) > studentDataListIn)

*Takes a List of Students and returns the value of their average current category discounting students with no hours.*
- static String [getActiveGoal](#) (List< [StudentData](#) > studentDataListIn)

*Takes a List of Students and returns the value of their average Community Service Goal discounting the students with no hours.*
- static List< [StudentData](#) > [removeInactive](#) (List< [StudentData](#) > studentDataListIn)

*Takes a list of Students and removes all of the students with no hours.*
- static List< [StudentData](#) > [removeActive](#) (List< [StudentData](#) > studentDataListIn)

*Returns a new list with only the inactive students.*

### Protected Member Functions

- [StudentData](#) [clone](#) () throws CloneNotSupportedException

#### 3.19.1 Detailed Description

This is a class which helps in running MySQL Methods and takes in all the values needed for the main student.

Author

Shourya Bansal

See also

[Student](#)

Definition at line 15 of file StudentData.java.

### 3.19.2 Constructor & Destructor Documentation

#### 3.19.2.1 `StudentData()` [1/2]

```
com.backend.StudentData.StudentData (
    boolean fromSelect )
```

A Constructor which allows the use of.

##### Parameters

<i>fromSelect</i>	the fromSelect value, typically true if this method is used.
-------------------	--

Definition at line 56 of file StudentData.java.

#### 3.19.2.2 `StudentData()` [2/2]

```
com.backend.StudentData.StudentData ( )
```

A Simple Constructor for the creation of a [StudentData](#) Object.

However, the [StudentData](#) Object can not be used unless the requisite setter methods are used

##### See also

- [setGrade\(int\)](#)
- [#setYearsDone\(short\)](#)
- [setStudentID\(int\)](#)
- [setStudent\(Student\)](#)
- [setCommunityServiceHours\(double\)](#)
- [setCommunityServiceCategory\(String\)](#)

Definition at line 73 of file StudentData.java.

### 3.19.3 Member Function Documentation



### 3.19.3.1 achievedGoal()

```
boolean com.backend.StudentData.achievedGoal ( )
```

Gets whether or not a student achieved their Community Service Award Category Goal.

#### Returns

a boolean of whether or not a student achieved his or her goal

Definition at line 499 of file StudentData.java.

### 3.19.3.2 getActiveCategory()

```
static String com.backend.StudentData.getActiveCategory (
    List< StudentData > studentDataListIn ) [static]
```

Takes a List of Students and returns the value of their average current category discounting students with no hours.

#### Parameters

<i>studentData↔ ListIn</i>	A List of Students which contains both active and inactive students
--------------------------------	---

#### Returns

The Average Category of only the Active Students

Definition at line 126 of file StudentData.java.

### 3.19.3.3 getActiveGoal()

```
static String com.backend.StudentData.getActiveGoal (
    List< StudentData > studentDataListIn ) [static]
```

Takes a List of Students and returns the value of their average Community Service Goal discounting the students with no hours.

#### Parameters

<i>studentData↔ ListIn</i>	A List of Students which contains both active and inactive students
--------------------------------	---

**Returns**

The Average Category Goal of only the Active Students

Definition at line 141 of file StudentData.java.

**3.19.3.4 getAverageCategory()**

```
static String com.backend.StudentData.getAverageCategory (
    List< StudentData > studentDataList ) [static]
```

Calculates the average achieved category in a list of students.

**Parameters**

<i>studentDataList</i>	The List of Students
------------------------	----------------------

**Returns**

The Average Category

Definition at line 84 of file StudentData.java.

**3.19.3.5 getAverageGoal()**

```
static String com.backend.StudentData.getAverageGoal (
    List< StudentData > studentDataList ) [static]
```

Calculates the Average Category Goal of a List of Students.

**Parameters**

<i>studentDataList</i>	The List of Students
------------------------	----------------------

**Returns**

The Average Category Goal

Definition at line 106 of file StudentData.java.

**3.19.3.6 getCommunityServiceCategory()**

```
String com.backend.StudentData.getCommunityServiceCategory ( )
```

Gets the Community Service Award Category Goal of the [Student](#).

**Returns**

The Community Service Award Category Goal

Definition at line 287 of file StudentData.java.

**3.19.3.7 getCommunityServiceCategoryInt()**

```
int com.backend.StudentData.getCommunityServiceCategoryInt ( )
```

Gets each Community Service Categories respective integer value.

**Returns**

the respective integer value of the [Student's](#) Category goal

Definition at line 319 of file StudentData.java.

**3.19.3.8 getCommunityServiceHours()**

```
double com.backend.StudentData.getCommunityServiceHours ( )
```

Gets the Amount of Community Service Hours the [Student](#) has participated in.

**Returns**

The Number of hours

Definition at line 248 of file StudentData.java.

**3.19.3.9 getCurrentCommunityServiceCategory()**

```
String com.backend.StudentData.getCurrentCommunityServiceCategory ( )
```

Gets the [Student's](#) current Community Service Award Category depending on the hours the student has done.

**Returns**

The Category

Definition at line 335 of file StudentData.java.

### 3.19.3.10 `getCurrentCommunityServiceCategoryInt()`

```
int com.backend.StudentData.getCurrentCommunityServiceCategoryInt ( )
```

Gets the student's current achieved category and returns its respective integer.

#### Returns

An integer representing the student's current category

#### See also

[getCurrentCommunityServiceCategory\(\)](#)

Definition at line 348 of file StudentData.java.

### 3.19.3.11 `getEmail()`

```
String com.backend.StudentData.getEmail ( )
```

Gets the student's email address.

#### Returns

The [Student](#)'s email address

Definition at line 364 of file StudentData.java.

### 3.19.3.12 `getGrade()`

```
short com.backend.StudentData.getGrade ( )
```

Gets the student's current grade level.

#### Returns

the current grade level

Definition at line 202 of file StudentData.java.

### 3.19.3.13 getGradeInt()

```
int com.backend.StudentData.getGradeInt ( )
```

Gets the student's current grade level.

#### Returns

the current grade level

Definition at line 239 of file StudentData.java.

### 3.19.3.14 getLastEdited()

```
Date com.backend.StudentData.getLastEdited ( )
```

Gets the date any of a [Student](#)'s information was last edited.

#### Returns

A [Date](#) containing the last date a value was changed

Definition at line 473 of file StudentData.java.

### 3.19.3.15 getYearsDone()

```
short com.backend.StudentData.getYearsDone ( )
```

Gets the number of years a student has participated in FBLA.

#### Returns

The Number of Years a [Student](#) Has Participated in FBLA

Definition at line 391 of file StudentData.java.

### 3.19.3.16 isActive()

```
boolean com.backend.StudentData.isActive ( )
```

Returns whether or not the student has any hours.

If a [Student](#) has no hours, the student is considered inactive, so some tables will ignore them when showing data

#### Returns

whether or not a student is active

Definition at line 527 of file StudentData.java.

### 3.19.3.17 removeActive()

```
static List<StudentData> com.backend.StudentData.removeActive (
    List< StudentData > studentDataListIn ) [static]
```

Returns a new list with only the inactive students.

**Parameters**

<i>studentData</i> ↔ <i>ListIn</i>	A List containing both active and inactive students
---------------------------------------	---

**Returns**

A new list with only inactive students

Definition at line 173 of file StudentData.java.

**3.19.3.18 removeInactive()**

```
static List<StudentData> com.backend.StudentData.removeInactive (
    List< StudentData > studentDataListIn ) [static]
```

Takes a list of Students and removes all of the students with no hours.

No changes are made to the original List to ensure that both the old and new list are still accessible by the User

**Parameters**

<i>studentData</i> ↔ <i>ListIn</i>	A List of Students with both active and inactive students
---------------------------------------	---

**Returns**

A new list with only the active students

Definition at line 157 of file StudentData.java.

**3.19.3.19 setCommunityServiceCategory()**

```
void com.backend.StudentData.setCommunityServiceCategory (
    String communityServiceCategoryIn )
```

Changes the Community Service Award Category, updating the backend if necessary.

**Parameters**

<i>communityService</i> ↔ <i>CategoryIn</i>	The new Community Service Award Category
--	--

Definition at line 296 of file StudentData.java.

**3.19.3.20 setCommunityServiceHours()**

```
void com.backend.StudentData.setCommunityServiceHours (
    double communityServiceHours )
```

Changes the number of Community Service Hours a student participated in, updating the backend if necessary.

**Parameters**

<i>communityServiceHours</i>	The new amount of community Service Hours
------------------------------	---

Definition at line 257 of file StudentData.java.

**3.19.3.21 setCommunityServiceHoursFromSelect() [1/2]**

```
void com.backend.StudentData.setCommunityServiceHoursFromSelect (
    double communityServiceHours )
```

Changes the number of Community Service Hours a student participated in and updating the backend.

**Parameters**

<i>communityServiceHours</i>	The new amount of community Service Hours
------------------------------	---

Definition at line 268 of file StudentData.java.

**3.19.3.22 setCommunityServiceHoursFromSelect() [2/2]**

```
void com.backend.StudentData.setCommunityServiceHoursFromSelect (
    String communityServiceHours )
```

Changes the number of Community Service Hours a student participated in and updating the backend.

**Parameters**

<i>communityServiceHours</i>	The new amount of community Service Hours
------------------------------	---

Definition at line 278 of file StudentData.java.

**3.19.3.23 setCommunityServiceHoursTemp()**

```
void com.backend.StudentData.setCommunityServiceHoursTemp (
    double communityServiceHours )
```

Temporarily sets the community Service hours of a student, typically used when looking at the hours in a range.



## Parameters

<i>communityServiceHours</i>	the temporary amount of hours of a student
------------------------------	--

Definition at line 516 of file StudentData.java.

### 3.19.3.24 setEmail()

```
void com.backend.StudentData.setEmail (
    @NotNull final String email )
```

Changes the student's email address, updating the backend if necessary.

## Parameters

<i>email</i>	The New email address
--------------	-----------------------

Definition at line 373 of file StudentData.java.

### 3.19.3.25 setFirstName()

```
void com.backend.StudentData.setFirstName (
    @NotNull final String firstName )
```

Changes the [Student](#)'s First Name, checking to see if the backend database must be changes as well.

## Parameters

<i>firstName</i>	the new first name
------------------	--------------------

Definition at line 537 of file StudentData.java.

### 3.19.3.26 setGrade() [1/2]

```
void com.backend.StudentData.setGrade (
    int grade )
```

Sets the Grade Level of the [Student](#), changing the backend database if necessary.

## Parameters

<i>grade</i>	The New Grade level
--------------	---------------------

Definition at line 211 of file StudentData.java.

### 3.19.3.27 setGrade() [2/2]

```
void com.backend.StudentData.setGrade (
    String grade )
```

Sets the Grade Level of the [Student](#), changing the backend database if necessary.

#### Parameters

<i>grade</i>	The New Grade level
--------------	---------------------

Definition at line 230 of file StudentData.java.

### 3.19.3.28 setLastEdited()

```
void com.backend.StudentData.setLastEdited (
    String lastEdited )
```

Changes the last-edited date of a [Student](#)'s information.

#### Parameters

<i>lastEdited</i>	The new lastEdited <a href="#">Date</a>
-------------------	---

Definition at line 482 of file StudentData.java.

### 3.19.3.29 setLastName()

```
void com.backend.StudentData.setLastName (
    @NotNull final String lastName )
```

Changes the last name of a student and updates the database if necessary.

#### Parameters

<i>lastName</i>	the new Last Name
-----------------	-------------------

Definition at line 557 of file StudentData.java.

**3.19.3.30 setStudentID()** [1/2]

```
void com.backend.StudentData.setStudentID (
    @NotNull final int studentID )
```

Changes the StudentID of a student, updating the database if necessary.

**Parameters**

<i>studentID</i>	the new <a href="#">Student</a> ID
------------------	------------------------------------

Definition at line 577 of file StudentData.java.

**3.19.3.31 setStudentID()** [2/2]

```
void com.backend.StudentData.setStudentID (
    String studentID )
```

Changes the StudentID of a student, updating the database if necessary.

**Parameters**

<i>studentID</i>	the new <a href="#">Student</a> ID
------------------	------------------------------------

Definition at line 594 of file StudentData.java.

**3.19.3.32 setYearsDone()** [1/2]

```
void com.backend.StudentData.setYearsDone (
    @NotNull final short yearsDone )
```

Changes the value of the number of years a student has participated in FBLA, updating the database if necessary.

**Parameters**

<i>yearsDone</i>	The new number of years done
------------------	------------------------------

Definition at line 401 of file StudentData.java.

**3.19.3.33 setYearsDone()** [2/2]

```
void com.backend.StudentData.setYearsDone (
    String yearsDone )
```

Changes the value of the number of years a student has participated in FBLA, updating the database if necessary.

## Parameters

<i>yearsDone</i>	The new number of years done
------------------	------------------------------

Definition at line 420 of file StudentData.java.

The documentation for this class was generated from the following file:

- StudentData.java



# Index

- achievedGoal
  - com.backend.StudentData, [72](#)
- achievedGoalPieChart
  - com.frontend.Charts, [7](#)
- addStudentHours
  - com.backend.MySQLMethods, [41](#)
- addUserPass
  - com.backend.MySQLMethods, [41](#)
- barGraphCommunityServiceCategoryAchieved
  - com.frontend.Charts, [8](#)
- barGraphCommunityServiceCategoryGoals
  - com.frontend.Charts, [8](#)
- beforeEnter
  - com.frontend.Login.LoginView, [36](#)
- clone
  - com.backend.Date, [15](#)
- com.backend.Date, [12](#)
  - clone, [15](#)
  - compare, [15](#)
  - compareTo, [15](#)
  - Date, [14](#)
  - equals, [16](#)
  - equalsNoDay, [16](#)
  - fakeDate, [17](#)
  - getDay, [17](#)
  - getLocalDate, [17](#)
  - getMonth, [17](#), [18](#)
  - getMonthString, [19](#)
  - getYear, [19](#)
  - optionToDate, [19](#)
  - setDate, [20](#)
  - setDay, [20](#)
  - setMonth, [21](#)
  - setYear, [21](#)
  - toString, [21](#)
  - toStringRegular, [21](#)
- com.backend.Event, [22](#)
  - compare, [25](#)
  - compareTo, [25](#)
  - Event, [24](#), [25](#)
  - getDate, [26](#)
  - getDay, [26](#)
  - getEventName, [26](#)
  - getHours, [27](#)
  - getLocalDate, [27](#)
  - getMonth, [27](#)
  - getMonthsWithYear, [27](#)
  - getTotalHours, [28](#)
  - getYear, [28](#)
  - monthsInRange, [28](#)
  - setDate, [29](#), [30](#)
  - setDay, [30](#)
  - setEventName, [30](#)
  - setHours, [31](#)
  - setMonth, [31](#)
  - setYear, [33](#)
  - toString, [33](#)
  - updateEvent, [33](#)
- com.backend.MySQLMethods, [38](#)
  - addStudentHours, [41](#)
  - addUserPass, [41](#)
  - createDatabase, [42](#)
  - createStudent, [42](#)
  - createStudentTable, [42](#)
  - delete, [43](#)
  - getHoursAll, [43](#)
  - getHoursMonth, [44](#)
  - getHoursWeek, [44](#)
  - getHoursYear, [45](#)
  - getStudentData, [45](#)
  - getStudents, [46](#)
  - makeName, [46](#)
  - numberOfRows, [47](#)
  - round, [47](#)
  - selectPass, [48](#)
  - selectStudentEventsAsEvent, [48](#), [49](#)
  - selectStudentEventsInRange, [49](#)
  - selectTrackerAsStudent, [49](#)
  - selectTrackerDouble, [50](#)
  - selectTrackerInt, [50](#)
  - selectTrackerShort, [52](#)
  - selectTrackerString, [52](#)
  - updateEvent, [53](#)
  - updateFirstName, [53](#)
  - updateLastName, [54](#)
  - updateStudentID, [54](#)
  - updateToCurrentDate, [54](#)
  - updateTracker, [55](#)
- com.backend.Percent, [56](#)
  - doubleValue, [58](#)
  - floatValue, [58](#)
  - getPercent, [58](#)
  - intValue, [58](#)
  - longValue, [59](#)
  - Percent, [57](#)
  - round, [59](#)
  - toString, [59](#)

- com.backend.Student, 64
  - getCreateNewStudent, 66
  - getFirstName, 66
  - getFullName, 66
  - getLastName, 67
  - getStudent, 67
  - getStudentData, 67
  - getStudentID, 67
  - setFirstName, 68
  - setLastName, 68
  - setStudent, 68
  - setStudentID, 69
  - Student, 65, 66
  - toString, 69
- com.backend.StudentData, 69
  - achievedGoal, 72
  - getActiveCategory, 73
  - getActiveGoal, 73
  - getAverageCategory, 74
  - getAverageGoal, 74
  - getCommunityServiceCategory, 74
  - getCommunityServiceCategoryInt, 75
  - getCommunityServiceHours, 75
  - getCurrentCommunityServiceCategory, 75
  - getCurrentCommunityServiceCategoryInt, 75
  - getEmail, 76
  - getGrade, 76
  - getGradeInt, 76
  - getLastEdited, 77
  - getYearsDone, 77
  - isActive, 77
  - removeActive, 77
  - removeInactive, 78
  - setCommunityServiceCategory, 78
  - setCommunityServiceHours, 78
  - setCommunityServiceHoursFromSelect, 79
  - setCommunityServiceHoursTemp, 79
  - setEmail, 81
  - setFirstName, 81
  - setGrade, 81, 82
  - setLastEdited, 82
  - setLastName, 82
  - setStudentID, 82, 83
  - setYearsDone, 83
  - StudentData, 72
- com.frontend.Add.AddHours.AddHours, 5
- com.frontend.Add.CreateStudent.CreateStudent, 12
- com.frontend.Application, 5
  - main, 6
- com.frontend.Charts, 6
  - achievedGoalPieChart, 7
  - barGraphCommunityServiceCategoryAchieved, 8
  - barGraphCommunityServiceCategoryGoals, 8
  - contributionTreeMapChart, 8, 9
  - currentDivisionChart, 9
  - goalDivisionChart, 10
  - monthLineGraph, 10
  - solidGauge, 11
- com.frontend.Documentation.Documentation, 22
- com.frontend.GetStudentInformation.GetStudentInformation, 34
  - deleteButton, 34
  - expandButton, 35
- com.frontend.Home, 35
- com.frontend.Login.LoginView, 36
  - beforeEnter, 36
- com.frontend.MainView, 37
  - getTabs, 38
- com.frontend.Reports.Reports, 60
  - groupReport, 61
  - individualReport, 61
  - makeBulletList, 61
  - nextSection, 62
  - setText, 62
- com.frontend.Security.ConfigureUIServiceInitListener, 11
- com.frontend.Security.SecurityConfiguration, 63
  - userDetailsService, 63
- com.frontend.Security.SecurityUtils, 64
- compare
  - com.backend.Date, 15
  - com.backend.Event, 25
- compareTo
  - com.backend.Date, 15
  - com.backend.Event, 25
- contributionTreeMapChart
  - com.frontend.Charts, 8, 9
- createDatabase
  - com.backend.MySQLMethods, 42
- createStudent
  - com.backend.MySQLMethods, 42
- createStudentTable
  - com.backend.MySQLMethods, 42
- currentDivisionChart
  - com.frontend.Charts, 9
- Date
  - com.backend.Date, 14
- delete
  - com.backend.MySQLMethods, 43
- deleteButton
  - com.frontend.GetStudentInformation.GetStudentInformation, 34
- doubleValue
  - com.backend.Percent, 58
- equals
  - com.backend.Date, 16
- equalsNoDay
  - com.backend.Date, 16
- Event
  - com.backend.Event, 24, 25
- expandButton
  - com.frontend.GetStudentInformation.GetStudentInformation, 35
- fakeDate



- com.backend.Date, 17
- floatValue
  - com.backend.Percent, 58
- getActiveCategory
  - com.backend.StudentData, 73
- getActiveGoal
  - com.backend.StudentData, 73
- getAverageCategory
  - com.backend.StudentData, 74
- getAverageGoal
  - com.backend.StudentData, 74
- getCommunityServiceCategory
  - com.backend.StudentData, 74
- getCommunityServiceCategoryInt
  - com.backend.StudentData, 75
- getCommunityServiceHours
  - com.backend.StudentData, 75
- getCreateNewStudent
  - com.backend.Student, 66
- getCurrentCommunityServiceCategory
  - com.backend.StudentData, 75
- getCurrentCommunityServiceCategoryInt
  - com.backend.StudentData, 75
- getDate
  - com.backend.Event, 26
- getDay
  - com.backend.Date, 17
  - com.backend.Event, 26
- getEmail
  - com.backend.StudentData, 76
- getEventName
  - com.backend.Event, 26
- getFirstName
  - com.backend.Student, 66
- getFullName
  - com.backend.Student, 66
- getGrade
  - com.backend.StudentData, 76
- getGradeInt
  - com.backend.StudentData, 76
- getHours
  - com.backend.Event, 27
- getHoursAll
  - com.backend.MySQLMethods, 43
- getHoursMonth
  - com.backend.MySQLMethods, 44
- getHoursWeek
  - com.backend.MySQLMethods, 44
- getHoursYear
  - com.backend.MySQLMethods, 45
- getLastEdited
  - com.backend.StudentData, 77
- getLastName
  - com.backend.Student, 67
- getLocalDate
  - com.backend.Date, 17
  - com.backend.Event, 27
- getMonth
  - com.backend.Date, 17, 18
  - com.backend.Event, 27
- getMonthString
  - com.backend.Date, 19
- getMonthsWithYear
  - com.backend.Event, 27
- getPercent
  - com.backend.Percent, 58
- getStudent
  - com.backend.Student, 67
- getStudentData
  - com.backend.MySQLMethods, 45
  - com.backend.Student, 67
- getStudentID
  - com.backend.Student, 67
- getStudents
  - com.backend.MySQLMethods, 46
- getTabs
  - com.frontend.MainView, 38
- getTotalHours
  - com.backend.Event, 28
- getYear
  - com.backend.Date, 19
  - com.backend.Event, 28
- getYearsDone
  - com.backend.StudentData, 77
- goalDivisionChart
  - com.frontend.Charts, 10
- groupReport
  - com.frontend.Reports.Reports, 61
- individualReport
  - com.frontend.Reports.Reports, 61
- intValue
  - com.backend.Percent, 58
- isActive
  - com.backend.StudentData, 77
- longValue
  - com.backend.Percent, 59
- main
  - com.frontend.Application, 6
- makeBulletList
  - com.frontend.Reports.Reports, 61
- makeName
  - com.backend.MySQLMethods, 46
- monthLineGraph
  - com.frontend.Charts, 10
- monthsInRange
  - com.backend.Event, 28
- nextSection
  - com.frontend.Reports.Reports, 62
- numberOfRows
  - com.backend.MySQLMethods, 47
- optionToDate
  - com.backend.Date, 19

- Percent
  - com.backend.Percent, [57](#)
- removeActive
  - com.backend.StudentData, [77](#)
- removeInactive
  - com.backend.StudentData, [78](#)
- round
  - com.backend.MySQLMethods, [47](#)
  - com.backend.Percent, [59](#)
- selectPass
  - com.backend.MySQLMethods, [48](#)
- selectStudentEventsAsEvent
  - com.backend.MySQLMethods, [48](#), [49](#)
- selectStudentEventsInRange
  - com.backend.MySQLMethods, [49](#)
- selectTrackerAsStudent
  - com.backend.MySQLMethods, [49](#)
- selectTrackerDouble
  - com.backend.MySQLMethods, [50](#)
- selectTrackerInt
  - com.backend.MySQLMethods, [50](#)
- selectTrackerShort
  - com.backend.MySQLMethods, [52](#)
- selectTrackerString
  - com.backend.MySQLMethods, [52](#)
- setCommunityServiceCategory
  - com.backend.StudentData, [78](#)
- setCommunityServiceHours
  - com.backend.StudentData, [78](#)
- setCommunityServiceHoursFromSelect
  - com.backend.StudentData, [79](#)
- setCommunityServiceHoursTemp
  - com.backend.StudentData, [79](#)
- setDate
  - com.backend.Date, [20](#)
  - com.backend.Event, [29](#), [30](#)
- setDay
  - com.backend.Date, [20](#)
  - com.backend.Event, [30](#)
- setEmail
  - com.backend.StudentData, [81](#)
- setEventName
  - com.backend.Event, [30](#)
- setFirstName
  - com.backend.Student, [68](#)
  - com.backend.StudentData, [81](#)
- setGrade
  - com.backend.StudentData, [81](#), [82](#)
- setHours
  - com.backend.Event, [31](#)
- setLastEdited
  - com.backend.StudentData, [82](#)
- setLastName
  - com.backend.Student, [68](#)
  - com.backend.StudentData, [82](#)
- setMonth
  - com.backend.Date, [21](#)
- com.backend.Event, [31](#)
- setStudent
  - com.backend.Student, [68](#)
- setStudentID
  - com.backend.Student, [69](#)
  - com.backend.StudentData, [82](#), [83](#)
- setText
  - com.frontend.Reports.Reports, [62](#)
- setYear
  - com.backend.Date, [21](#)
  - com.backend.Event, [33](#)
- setYearsDone
  - com.backend.StudentData, [83](#)
- solidGauge
  - com.frontend.Charts, [11](#)
- Student
  - com.backend.Student, [65](#), [66](#)
- StudentData
  - com.backend.StudentData, [72](#)
- toString
  - com.backend.Date, [21](#)
  - com.backend.Event, [33](#)
  - com.backend.Percent, [59](#)
  - com.backend.Student, [69](#)
- toStringRegular
  - com.backend.Date, [21](#)
- updateEvent
  - com.backend.Event, [33](#)
  - com.backend.MySQLMethods, [53](#)
- updateFirstName
  - com.backend.MySQLMethods, [53](#)
- updateLastName
  - com.backend.MySQLMethods, [54](#)
- updateStudentID
  - com.backend.MySQLMethods, [54](#)
- updateToDate
  - com.backend.MySQLMethods, [54](#)
- updateTracker
  - com.backend.MySQLMethods, [55](#)
- userDetailsService
  - com.frontend.Security.SecurityConfiguration, [63](#)