Name Id

advance Programming coursework report

Table of Contents

[Section 1 2](#_Toc5543239)

[Section 2 2](#_Toc5543240)

[Section 3 5](#_Toc5543241)

[3.1 Bugs - List plus brief description 5](#_Toc5543242)

[3.2 Weaknesses - List plus brief description 5](#_Toc5543243)

[Section 4 5](#_Toc5543244)

[Level 1 5](#_Toc5543245)

[Level 2 6](#_Toc5543246)

[a. UML class diagram for the inheritance hierarchy 6](#_Toc5543247)

[b. UML class diagram for the interfaces 6](#_Toc5543248)

[c. Brief Reflection 7](#_Toc5543249)

[Level 3 7](#_Toc5543250)

[Level 4 8](#_Toc5543251)

[Pattern 1 8](#_Toc5543252)

[Pattern 2 9](#_Toc5543253)

[Pattern 3 10](#_Toc5543254)

[Level 5 10](#_Toc5543255)

[Level 6 11](#_Toc5543256)

[a. Dials indicators move smoothly 11](#_Toc5543257)

[b. Simple Game 11](#_Toc5543258)

[Section 5 12](#_Toc5543259)

[Section 6 18](#_Toc5543260)

[GitHub Commits 19](#_Toc5543261)

[Section 7 24](#_Toc5543262)

[Section 8 24](#_Toc5543263)

[References 24](#_Toc5543264)

# Section 1

The application that we designed is a dashboard for a small air craft. We managed to go through all the levels, and, according to our own view, we succeeded in implementing all the requirements and made a functional software which delivers what is expected to do. We did not approach it from the first level to the last one in the given order, Level 5 and Level 2 were attempted in the end. We made use of two libraries since SteelSeries depends on trident, thus both, Grunwald (2012) and Gruchnikov (2019) can be found in the references for our report.

|  |  |
| --- | --- |
| Which type of dashboard application did you implement? | Train small air craft |
| 1.1 Circle the parts of the coursework you have fully completed and are fully working | 1a 1b 2a 2bi 2bii 3  4a 4bi 4bii 5a 5b 6a 6b |

# Section 2

As it can be noticed in our UML Class Diagram, Dashboard class is in the centre since it interacts with most of the classes, there are few aggregation relationships and plenty of simple associations with other classes that provide different functionality.

There are 2 diagrams for this section, as it felt right to add one with class names only (Figure 1) so that the relationships between classes are easier to read, and the second one is a more in-detail diagram (Figure 2) which contains attributes and methods.

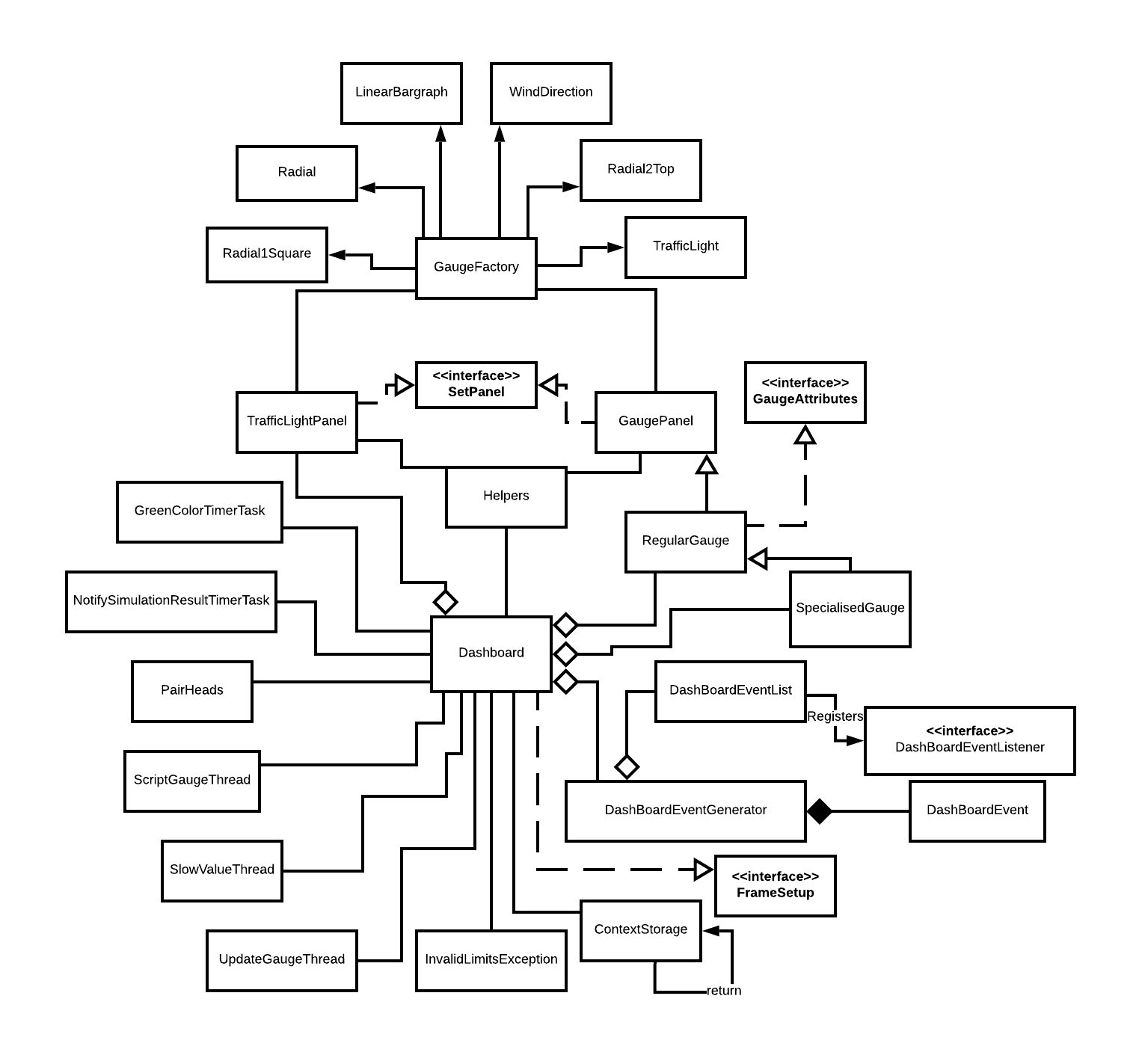


Figure Simple UML Class Diagram

The classes that came from the DashboardDemoProject which we were given as a start were not included in the detailed diagram, neither the classes for dials, as they were not created by us and we decided not to add them, additionally there was a lot of content there so probably there will be too much detail. It is only Figure 1 that contains every class we make use of in our program.

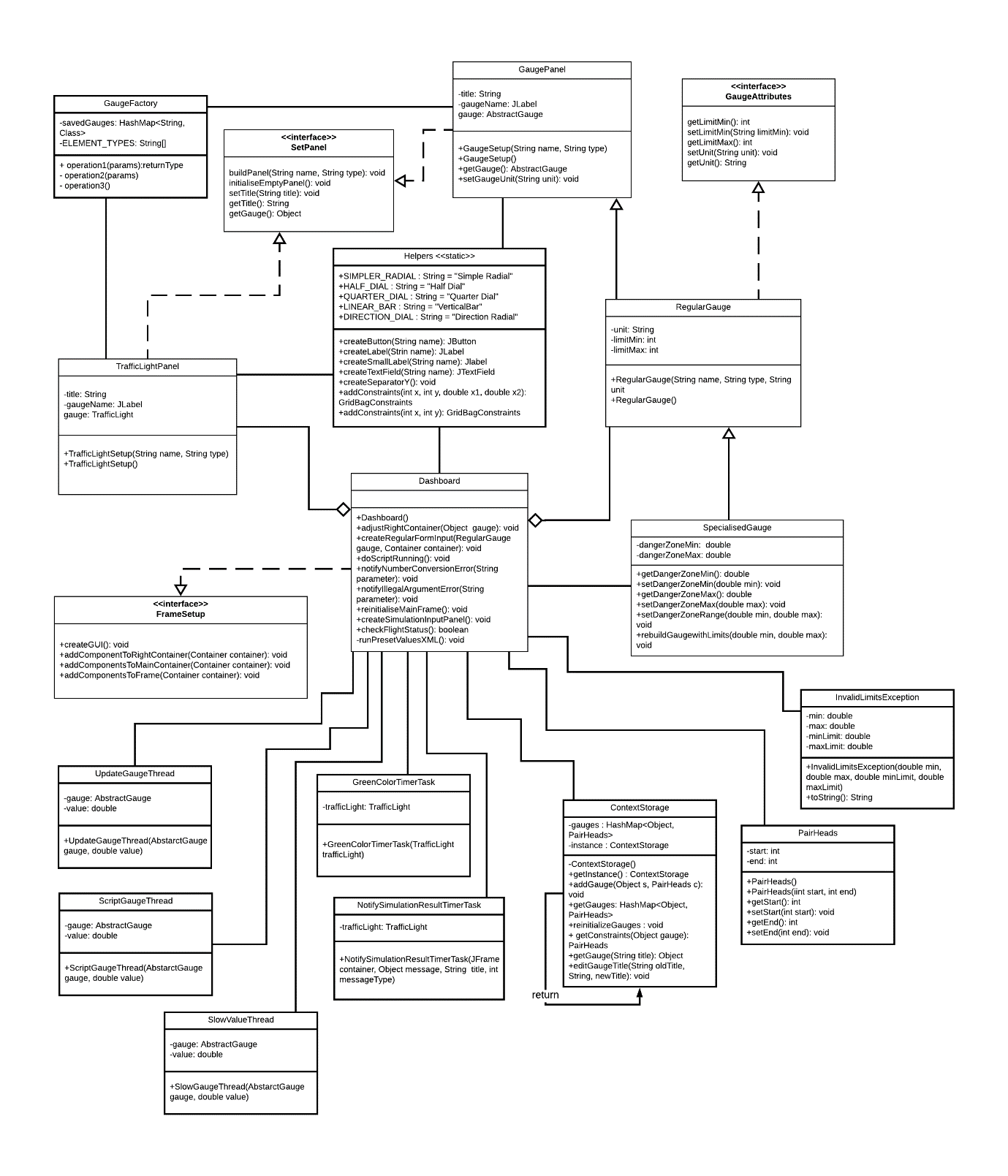


Figure Detailed UML Class Diagram

# Section 3

## 3.1 Bugs - List plus brief description

* **Second click on the same dial -** The graphical user interface has a strange behaviour once a user clicks the second time on the same gauge, no matter of the time difference between the two clicks, thr panel that allows the user to edit different parameters of the dial disappears and only one text field remains, the one to change the value where the dial currently points at.
* **Maximum and minimum for a gauge –** If user wants to change the range of the selected dial, the value display will not be shown as the one inserted, on the screen only shows an rounded value to tenths, this is how the library for gauges works, thus the actual range will be sometimes more or less than the actual value desired by the user;

## 3.2 Weaknesses - List plus brief description

* **No complete check for certain inputs** - For the simulation game, when user inputs more than the maximum value, or less than minimum, there is no check to verify if the input exceeds the given range and technically user can add any value. Also, values are stored as double, and although it requires a lot of digits for overflow, it could still happen as there is no check that verifies the value used as input.

# Section 4

## Level 1

The code that we were provided with seemed to be simple and easy to use as well, however the solution we provided has more methods that simplify the creation of some of the GUI components, including containers with multiple text fields, labels and buttons. Additionally, it makes use of SteelSeries libraries which comes with various types of gauges, thus no need for writing the code to create a dial manually. Moreover, threading improves the overall performance of the application, which prevents the GUI from freezing when executing tasks that require slightly more than one second.

## Level 2

### UML class diagram for the inheritance hierarchy

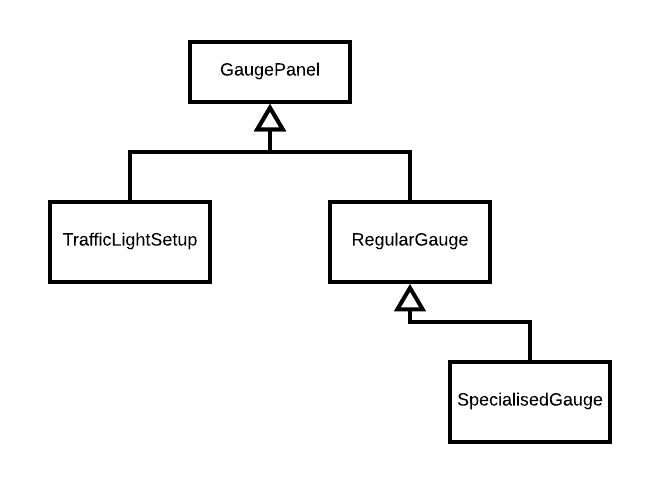


Figure UML Inheritance Hierarchy

### UML class diagram for the interfaces

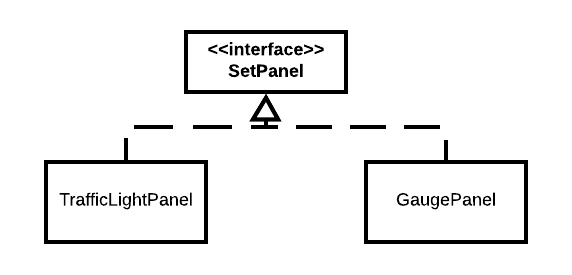


Figure UML SetPanel interface

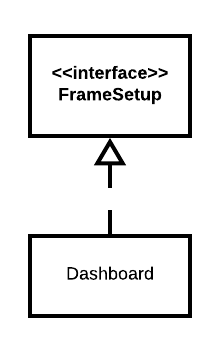


Figure UML FrameSetup interface

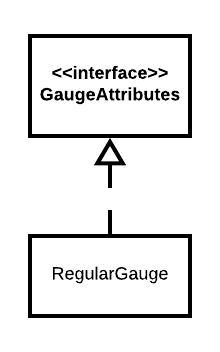


Figure UML GaugeAttributes interface

### Brief Reflection

The object-oriented features implemented for this level make the code cleaner and easier to maintain because we are making use of inheritance and interfaces which allows a programmer to add new fields or methods to a super class and make all the other classes automatically have the same properties if the right visibility is set, same applies for interfaces, where no matter of the parameters of the different classes, functions will return what they are meant to, depending on the class that implements it.

## Level 3

For Junit testing the following classes where tested:

* GaugePanel – 12 test cases;
* RegularGauge – 8 test cases;

## Level 4

### Pattern 1

Name: Singleton

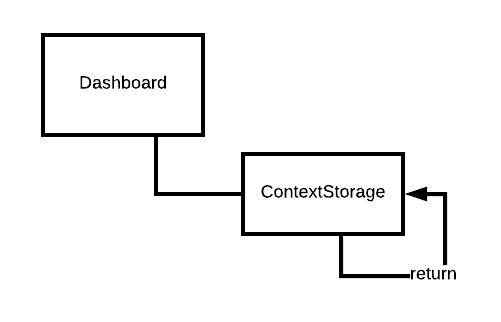


Figure ContextStorage Singleton

Implementing the Singleton pattern adds limitations to the number of instances of the class that can exist. Additionally, since it has the same values throughout the app life-cycle, accessing it from different classes should return the same result, thus storing information could be considered, such as user settings. This can be easily shown in the code of the application where, instead of having a global variable, when the instance of class is needed it’s locally initialised, thus it always has the latest updated values.

### Pattern 2

Name: Factory

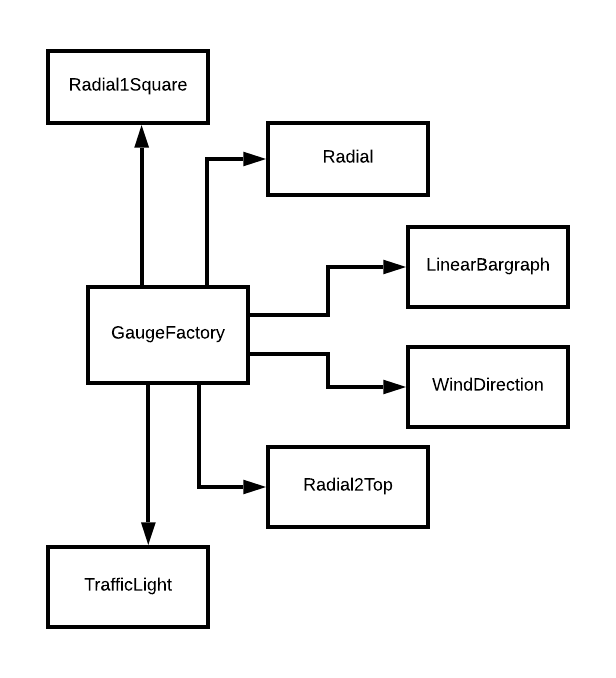


Figure Factory Pattern

In general, the factory pattern is used to create instances of objects when there is uncertainty regarding the type of the object that should be instantiated, so instead of using a specific type, through casting the desired results can be achieved. In future, if the programmer decides to allow the user to dynamically instantiate objects at running time, this feature will prove very useful, and, in case more variety is to be added in matter of the dials that can be generated it will require adding only two lines of code for each new component. All the components that GaugeFactory returns are part of the SteelSeries library (Grunwald, 2012).

### Pattern 3

Name: Servant

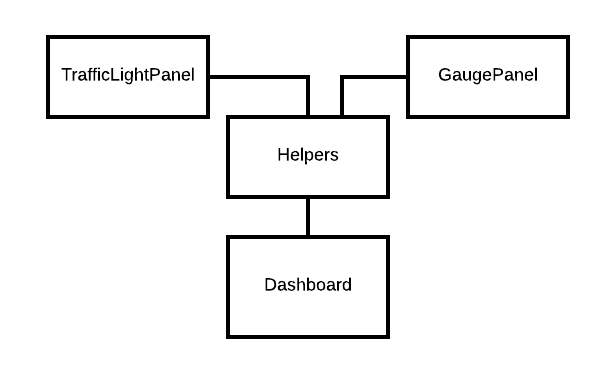


Figure UML Helpers class

The Servant Pattern can provide a programmer with a lot of functionality if same functionality is needed in classes that have nothing in common, also constants can be kept inside this class. The servant does not need to know anything about the class that is using its functions, only the serviced class needs to know exactly what it needs from the Servant and asked for it. It is easy to maintain, and a programmer should not find it difficult to add extra functionality.

## Level 5

Class name: Converter

The created JavaBean is meant to serve as a converter for anything, user can set a scale so that every time calculus is required, the solution is based on the scale that was set.

There are four properties in total (4 getters, 4 setters), which allows changing of the units displayed next to the input fields, the name of the converter and the scale. The icon of our bean is also displayed in the pallets.

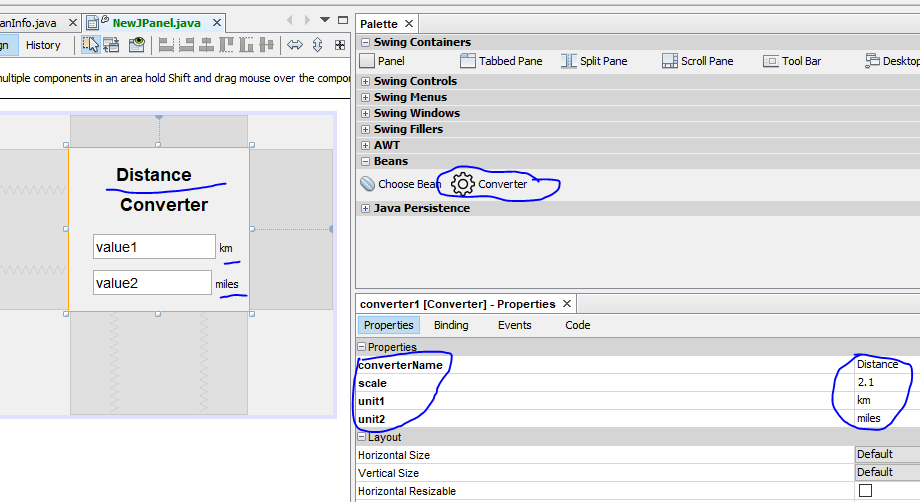


Figure Converter JavaBean

## Level 6

### Dials indicators move smoothly

In order to make the indicators of the dials move smoothly threads were required. Classes that extend the default Thread API class or implement Runnable interface were created so that the desired results can be achieved. Also, depending on the use of the Thread, thread-safety was added either on the instance of the object or on a given class. For example, class UpdateGaugeThread is meant to allow the user to set the value for a gauge, and thread safety is set for the entire class, so that no other dial will have its value updated until the one currently running finished the processing.

Weaknesses:

* User cannot update the values of other dials until the current one does not finish;
* A thread is sent to sleep in a for loop which might result in an InterruptedException and no proper feedback to the user of such error was implemented;

### Simple Game

The game is meant to simulate a take-off and a landing of an airplane, although is relatively short. If the user selects the button “Play simulation”, a panel appears, and the user can fill in with values. The simulation is implemented by reading from a file predefined values, so it always runs the same way. The movements on the dials are smooth and based on given input an algorithm is used to calculate whether the flight will be successful, or the plane will crash.

Weaknesses:

* Since the simulation runs from a script file, it is always the same effects taking place on dials;
* There is no limitation for the number that can be inputted in a text field, thus if user decides to add a very large number, calculation errors may occur;

# Section 5

Running the application will output the following screen:

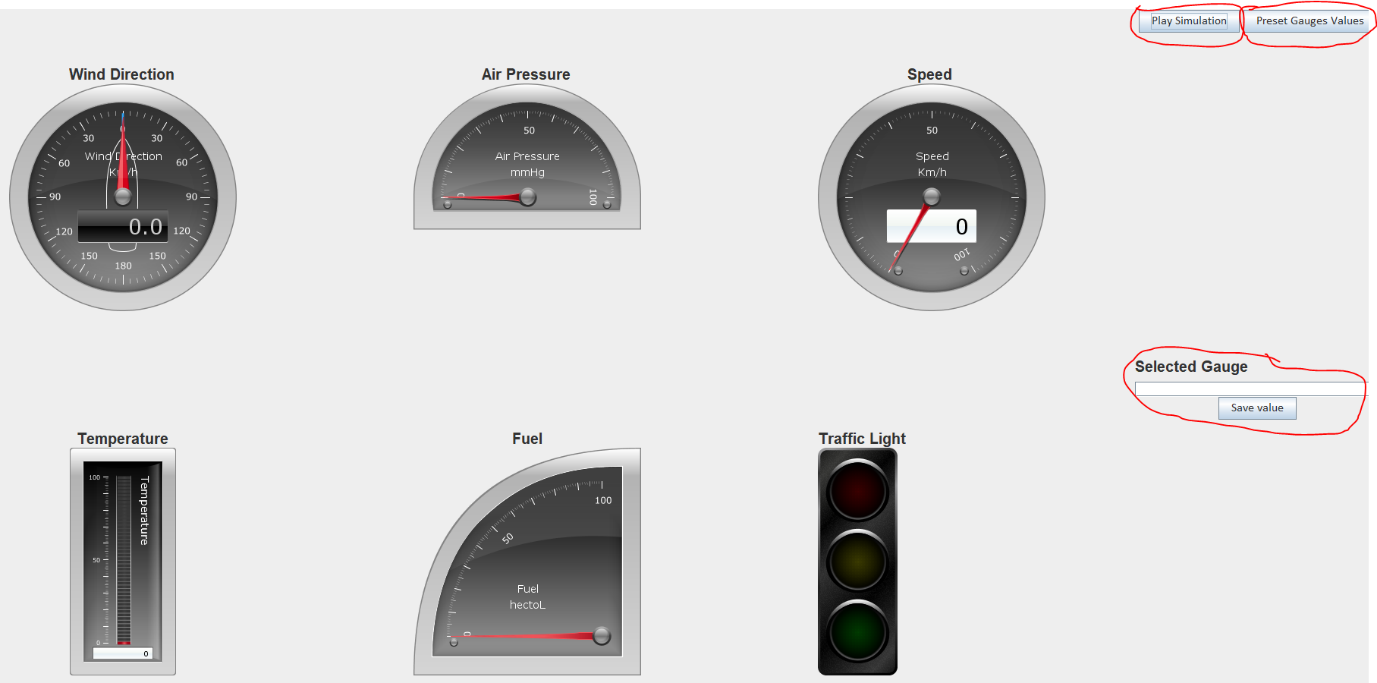


Figure First run of the application

The user should be able to see four dials, one vertical bar and traffic lights. Additionally, there are three buttons on the right-hand side of the screen, each will have a different functionality. If the user attempts to enter a value and clicks the button “Save value” without previously selecting one of the six dashboard elements the following error message will show:

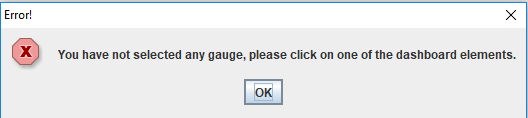


Figure No gauge selected

Selecting the button “Preset gauges values” will make the values of all the dashboard elements change based on one of the xml scripts written for this purpose, the change of the values is smooth and all of them will have their indicators move at the same time. The result should be the following:

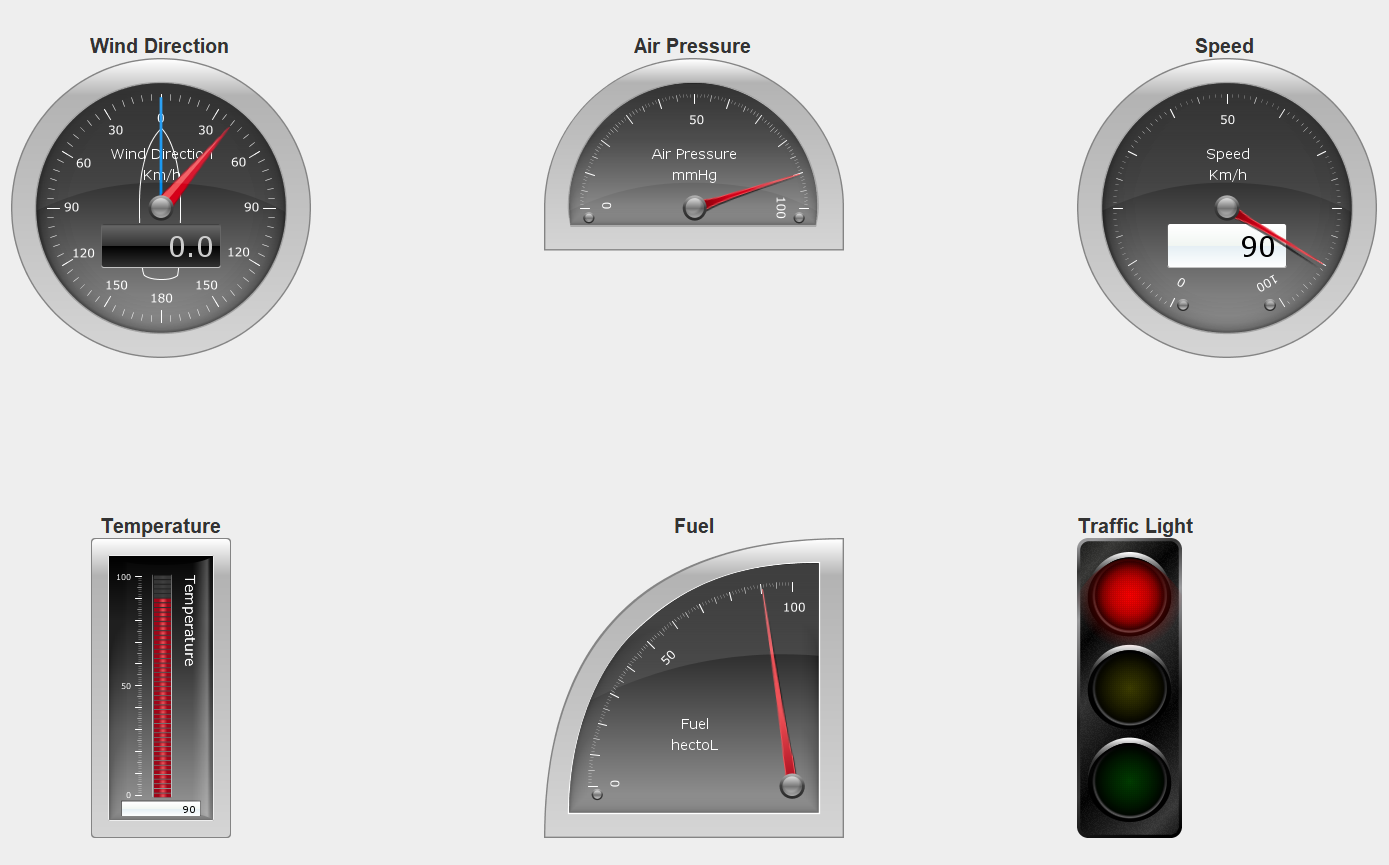


Figure Preset Gauges Values

The other button left, “Play Simulation”, changes the right-hand side panel as it can be noticed in the next image, some input fields are added:

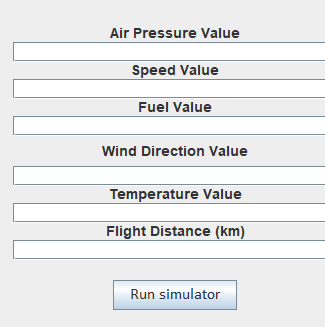


Figure Simulation Input Panel

Technically, users can input any kind of values in the fields, the indicators of the gauges only point to the right value if it is a valid one, although the digital display could show values outside of the range of the gauge itself. Once the user hits the “Run simulator” button, if errors occur there should be a pop-up message dialog box showing the error:

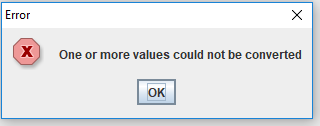


Figure Conversion error

The message is not very specific, but the user should double-check the input fields. If the user inputs values which make the simulation start, and changes should be seen on the screen:

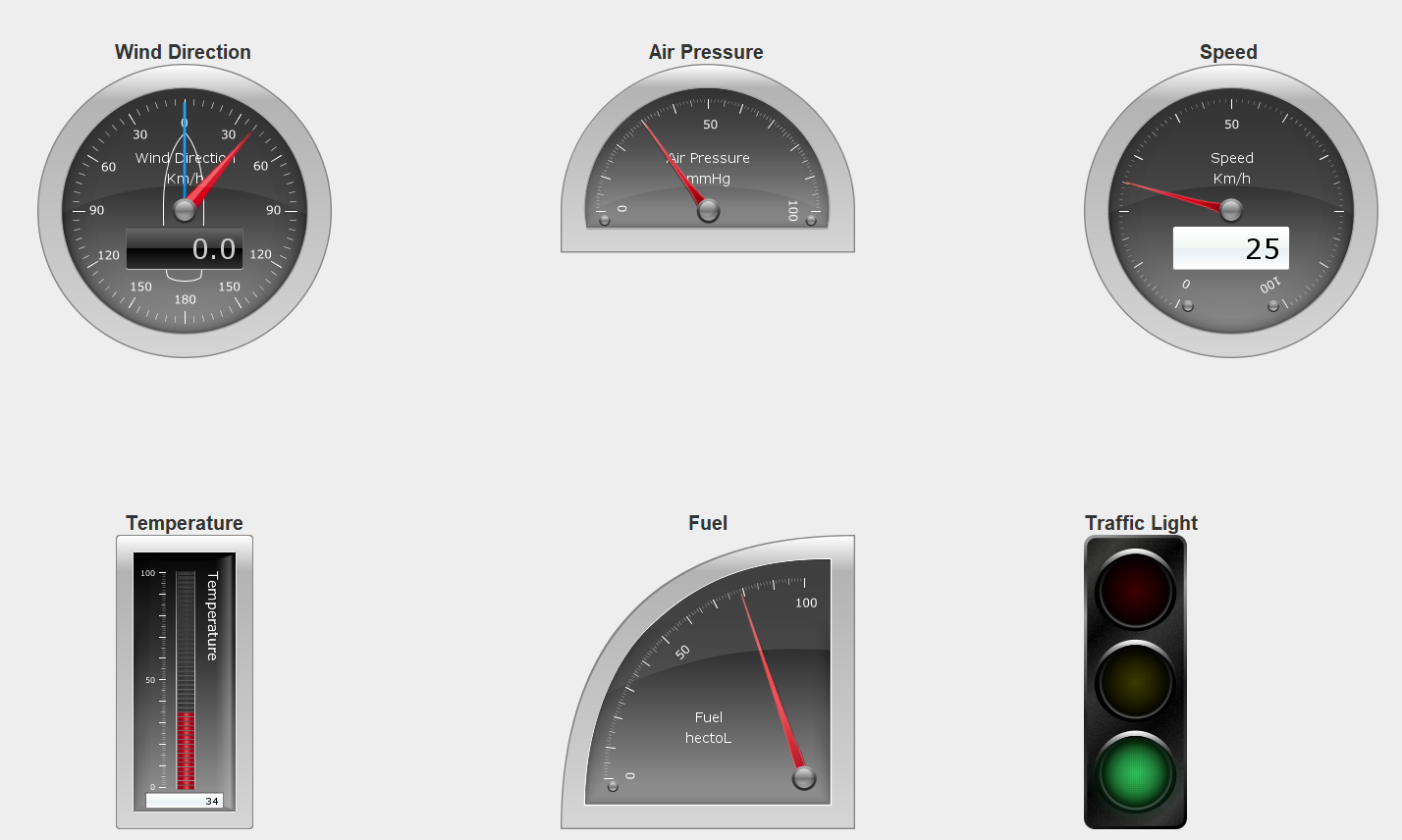


Figure Running Simulation

Depending on the values inputted by the user one of the following messages will be shown after the simulation finishes:

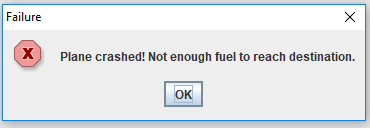


Figure Plane crashed

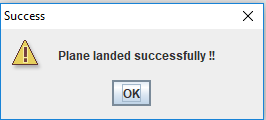


Figure Plane landed successfully

If the user clicks on any of the dashboard elements, depending on the selected component one of the following is shown in the panel on the right:

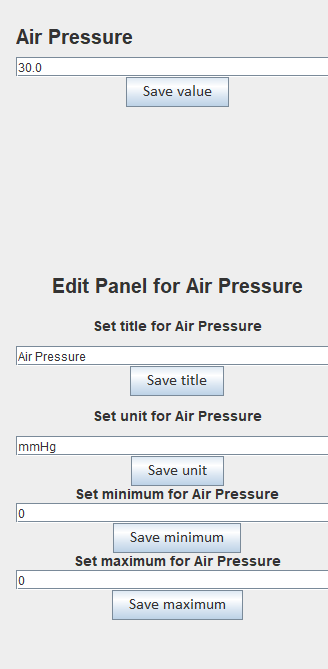


Figure An instance of RegularGauge selected

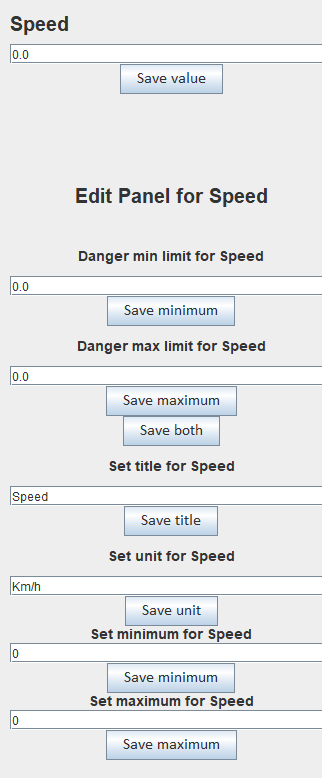
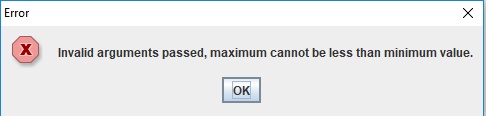


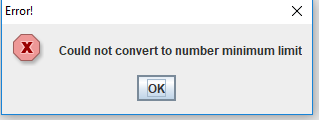
Figure Instance of SpecialisedGauge selected



Figure Instance of TrafficLightSetup selected

This time, for each of the input fields the message boxes are more precise with the information they supply to the user, as they let the user know which text field has an incorrect parameter:





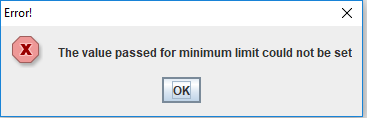


Figure Error boxes

One of the weird behaviours our app is experiencing is that when a user clicks the second time on the same dial and makes the right panel only display the input field for the value of the dial, even though it displays the title for the correct gauge, there is a message dialog box that informs the user the value that was added could not be set, however there is no issue with the input itself. A picture with the error is attached below.

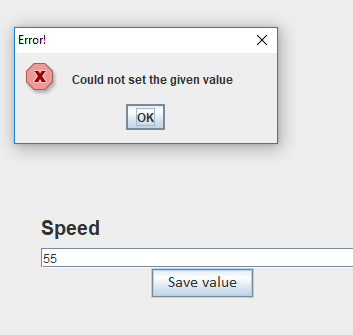


Figure Bug when user clicks second time

# Section 6

Throughout the period we worked on the project for this coursework we used GitHub which is a web-service for version control that uses Git. We attached a screenshot of our repository together with a part of the commits we did, although it is only one name mainly appearing there it is because we mainly pair programmed, thus only one of us will be at the computer at a time, thus we mainly used Madalin’s account when pushing changes since our strategy involved him at the computer and Gabriel assisting as we discovered we were more efficient that way.

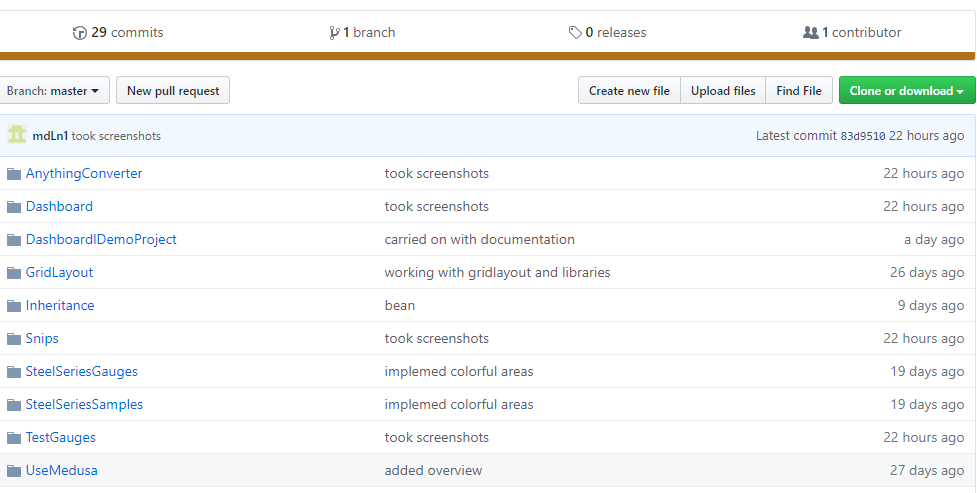


Figure GitHub Private Repository

### GitHub Commits

Commits on Apr 5, 2019

1. [**took screenshots**](https://github.com/mdLn1/PlaneDashboard/commit/83d9510113ef2d44ec0fbbb62510ea4c48b78358)

[@mdLn1](https://github.com/mdLn1)

[**mdLn1**](https://github.com/mdLn1/PlaneDashboard/commits?author=mdLn1) committed 22 hours ago

[**83d9510**](https://github.com/mdLn1/PlaneDashboard/commit/83d9510113ef2d44ec0fbbb62510ea4c48b78358)

1. [**carried on with documentation**](https://github.com/mdLn1/PlaneDashboard/commit/3e5ce6aaa6b0ee374fb208dae19e7c7b4a65fa84)

[@mdLn1](https://github.com/mdLn1)

[**mdLn1**](https://github.com/mdLn1/PlaneDashboard/commits?author=mdLn1) committed a day ago

[**3e5ce6a**](https://github.com/mdLn1/PlaneDashboard/commit/3e5ce6aaa6b0ee374fb208dae19e7c7b4a65fa84)

1. [**worked on documentation**](https://github.com/mdLn1/PlaneDashboard/commit/bb40d2059f33148921c5a069c39b0458a272204b)

[@mdLn1](https://github.com/mdLn1)

[**mdLn1**](https://github.com/mdLn1/PlaneDashboard/commits?author=mdLn1) committed 2 days ago

[**bb40d20**](https://github.com/mdLn1/PlaneDashboard/commit/bb40d2059f33148921c5a069c39b0458a272204b)

Commits on Apr 4, 2019

1. [**finished with comments**](https://github.com/mdLn1/PlaneDashboard/commit/85ee27108c948fc598d19bb15978ae3b6cc89757)

[@mdLn1](https://github.com/mdLn1)

[**mdLn1**](https://github.com/mdLn1/PlaneDashboard/commits?author=mdLn1) committed 2 days ago

[**85ee271**](https://github.com/mdLn1/PlaneDashboard/commit/85ee27108c948fc598d19bb15978ae3b6cc89757)

Commits on Apr 3, 2019

1. [**game added**](https://github.com/mdLn1/PlaneDashboard/commit/4d83b03ef21a92fe3ff753d3e9f3fbc9e402685a)

[@mdLn1](https://github.com/mdLn1)

[**mdLn1**](https://github.com/mdLn1/PlaneDashboard/commits?author=mdLn1) committed 3 days ago

[**4d83b03**](https://github.com/mdLn1/PlaneDashboard/commit/4d83b03ef21a92fe3ff753d3e9f3fbc9e402685a)

Commits on Apr 2, 2019

1. [**run from script**](https://github.com/mdLn1/PlaneDashboard/commit/96e55cdaa2a220d9f1a8eebf3d88658395fda2b0)

[@mdLn1](https://github.com/mdLn1)

[**mdLn1**](https://github.com/mdLn1/PlaneDashboard/commits?author=mdLn1) committed 4 days ago

[**96e55cd**](https://github.com/mdLn1/PlaneDashboard/commit/96e55cdaa2a220d9f1a8eebf3d88658395fda2b0)

Commits on Mar 29, 2019

1. [**removed some irrelevant code and finished bean**](https://github.com/mdLn1/PlaneDashboard/commit/0d03774e2c3e4220c7868ba2e0e4794d4e2d1475)

[@mdLn1](https://github.com/mdLn1)

[**mdLn1**](https://github.com/mdLn1/PlaneDashboard/commits?author=mdLn1) committed 8 days ago

[**0d03774**](https://github.com/mdLn1/PlaneDashboard/commit/0d03774e2c3e4220c7868ba2e0e4794d4e2d1475)

Commits on Mar 28, 2019

1. [**bean**](https://github.com/mdLn1/PlaneDashboard/commit/9f9f6960ef9c159f3f426c138a76f0a85bff4ec5)

[@mdLn1](https://github.com/mdLn1)

[**mdLn1**](https://github.com/mdLn1/PlaneDashboard/commits?author=mdLn1) committed 9 days ago

[**9f9f696**](https://github.com/mdLn1/PlaneDashboard/commit/9f9f6960ef9c159f3f426c138a76f0a85bff4ec5)

1. [**little changes**](https://github.com/mdLn1/PlaneDashboard/commit/e550898ac24d3af2b9f75cea0a99d48f79dc0b76)

[@mdLn1](https://github.com/mdLn1)

[**mdLn1**](https://github.com/mdLn1/PlaneDashboard/commits?author=mdLn1) committed 9 days ago

[**e550898**](https://github.com/mdLn1/PlaneDashboard/commit/e550898ac24d3af2b9f75cea0a99d48f79dc0b76)

Commits on Mar 22, 2019

1. [**started working on simulation**](https://github.com/mdLn1/PlaneDashboard/commit/6f89ef4f830f23a91a0cc68c376f4b6d40dea06e)

[@mdLn1](https://github.com/mdLn1)

[**mdLn1**](https://github.com/mdLn1/PlaneDashboard/commits?author=mdLn1) committed 16 days ago

[**6f89ef4**](https://github.com/mdLn1/PlaneDashboard/commit/6f89ef4f830f23a91a0cc68c376f4b6d40dea06e)

Commits on Mar 21, 2019

1. [**added latest version**](https://github.com/mdLn1/PlaneDashboard/commit/9848c1599d5eb23898ff5e7cc0949fd91cf7e95a)

[@mdLn1](https://github.com/mdLn1)

[**mdLn1**](https://github.com/mdLn1/PlaneDashboard/commits?author=mdLn1) committed 16 days ago

Verified

[**9848c15**](https://github.com/mdLn1/PlaneDashboard/commit/9848c1599d5eb23898ff5e7cc0949fd91cf7e95a)

Commits on Mar 20, 2019

1. [**started working on simulation**](https://github.com/mdLn1/PlaneDashboard/commit/1b7505e97d3e7ffec5ba5abe488d4ac0c4727f10)

[@mdLn1](https://github.com/mdLn1)

[**mdLn1**](https://github.com/mdLn1/PlaneDashboard/commits?author=mdLn1) committed 17 days ago

[**1b7505e**](https://github.com/mdLn1/PlaneDashboard/commit/1b7505e97d3e7ffec5ba5abe488d4ac0c4727f10)

Commits on Mar 19, 2019

1. [**fully functional gauges**](https://github.com/mdLn1/PlaneDashboard/commit/5f9d08be653bab335d001257c8bd06ec2e619ded)

[@mdLn1](https://github.com/mdLn1)

[**mdLn1**](https://github.com/mdLn1/PlaneDashboard/commits?author=mdLn1) committed 18 days ago

[**5f9d08b**](https://github.com/mdLn1/PlaneDashboard/commit/5f9d08be653bab335d001257c8bd06ec2e619ded)

Commits on Mar 18, 2019

1. [**implemed colorful areas**](https://github.com/mdLn1/PlaneDashboard/commit/88192ee9fcb3c06ba341b857f1ee60e338a268d9)

[@mdLn1](https://github.com/mdLn1)

[**mdLn1**](https://github.com/mdLn1/PlaneDashboard/commits?author=mdLn1) committed 19 days ago

[**88192ee**](https://github.com/mdLn1/PlaneDashboard/commit/88192ee9fcb3c06ba341b857f1ee60e338a268d9)

1. [**refresh interface and listeners**](https://github.com/mdLn1/PlaneDashboard/commit/a3268d66de44d457b2d74726e8c0fe005c97570f)

[@mdLn1](https://github.com/mdLn1)

[**mdLn1**](https://github.com/mdLn1/PlaneDashboard/commits?author=mdLn1) committed 19 days ago

[**a3268d6**](https://github.com/mdLn1/PlaneDashboard/commit/a3268d66de44d457b2d74726e8c0fe005c97570f)

Commits on Mar 17, 2019

1. [**singleton**](https://github.com/mdLn1/PlaneDashboard/commit/c665d7b1869ccbd104d0aef320cb46ebe4d83ffe)

[@mdLn1](https://github.com/mdLn1)

[**mdLn1**](https://github.com/mdLn1/PlaneDashboard/commits?author=mdLn1) committed 20 days ago

[**c665d7b**](https://github.com/mdLn1/PlaneDashboard/commit/c665d7b1869ccbd104d0aef320cb46ebe4d83ffe)

Commits on Mar 16, 2019

1. [**implemented few design features**](https://github.com/mdLn1/PlaneDashboard/commit/c705d350d95da1c7ced5f4462fda2e5202d1aa59)

[@mdLn1](https://github.com/mdLn1)

[**mdLn1**](https://github.com/mdLn1/PlaneDashboard/commits?author=mdLn1) committed 21 days ago

[**c705d35**](https://github.com/mdLn1/PlaneDashboard/commit/c705d350d95da1c7ced5f4462fda2e5202d1aa59)

Commits on Mar 12, 2019

1. [**factory patern**](https://github.com/mdLn1/PlaneDashboard/commit/8aceaea21f2ef6e0a9371d34388463a9b04f478c)

[@mdLn1](https://github.com/mdLn1)

[**mdLn1**](https://github.com/mdLn1/PlaneDashboard/commits?author=mdLn1) committed 25 days ago

[**8aceaea**](https://github.com/mdLn1/PlaneDashboard/commit/8aceaea21f2ef6e0a9371d34388463a9b04f478c)

1. [**worked on inheritance and interfaces**](https://github.com/mdLn1/PlaneDashboard/commit/3013955f40e5ed95b7d0ec55ee23a17c7a4da693)

[@mdLn1](https://github.com/mdLn1)

[**mdLn1**](https://github.com/mdLn1/PlaneDashboard/commits?author=mdLn1) committed 25 days ago

[**3013955**](https://github.com/mdLn1/PlaneDashboard/commit/3013955f40e5ed95b7d0ec55ee23a17c7a4da693)

Commits on Mar 11, 2019

1. [**built sample with gauges for coursework**](https://github.com/mdLn1/PlaneDashboard/commit/36ed3f64a05ad7e84886b6281d0e07c9a6c2dcfc)

[@mdLn1](https://github.com/mdLn1)

[**mdLn1**](https://github.com/mdLn1/PlaneDashboard/commits?author=mdLn1) committed 26 days ago

[**36ed3f6**](https://github.com/mdLn1/PlaneDashboard/commit/36ed3f64a05ad7e84886b6281d0e07c9a6c2dcfc)

1. [**working with gridlayout and libraries**](https://github.com/mdLn1/PlaneDashboard/commit/c2170004e8a40ee890383709c1df0e8a6c48c308)

[@mdLn1](https://github.com/mdLn1)

[**mdLn1**](https://github.com/mdLn1/PlaneDashboard/commits?author=mdLn1) committed 26 days ago

[**c217000**](https://github.com/mdLn1/PlaneDashboard/commit/c2170004e8a40ee890383709c1df0e8a6c48c308)

Commits on Mar 10, 2019

1. [**added overview**](https://github.com/mdLn1/PlaneDashboard/commit/d2db1f6e638b474e224e01a05dad134a28525db2)

[@mdLn1](https://github.com/mdLn1)

[**mdLn1**](https://github.com/mdLn1/PlaneDashboard/commits?author=mdLn1) committed 27 days ago

[**d2db1f6**](https://github.com/mdLn1/PlaneDashboard/commit/d2db1f6e638b474e224e01a05dad134a28525db2)

1. [**first commit**](https://github.com/mdLn1/PlaneDashboard/commit/5ec6f06d8fe0e65446a23e2f808552fa6b233151)

[@mdLn1](https://github.com/mdLn1)

[**mdLn1**](https://github.com/mdLn1/PlaneDashboard/commits?author=mdLn1) committed 27 days ago

# Section 7

# Section 8

# References

1. Grunwald G. (2012), SteelSeries-Swing [WWW] Github.COM. <https://github.com/HanSolo/SteelSeries-Swing/blob/master/src/main/java/eu/hansolo/steelseries/gauges/AbstractGauge.java> Accessed on: [10/03/2019]
2. Gruchnikov K. (2019), radiance/trident [WWW] Github.COM. <https://github.com/kirill-grouchnikov/radiance/tree/master/docs/trident> Accessed on: [10/03/2019]