

---

# **Konfigurator für OSM-Datenaufbereitungs-Prozesse**

*Release 1.0.0*

**Felix Weik, Jan-Phillip Hansen, Karl Bernhard, Pascal Dawideit, S**

**Jan 12, 2023**



# USER GUIDE

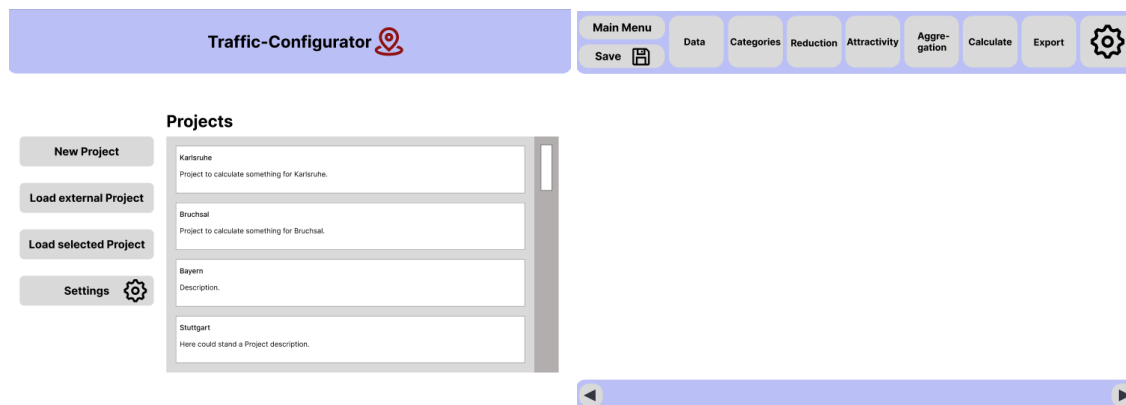
<b>1</b>	<b>KonfiguratorFuerOSMDaten</b>	<b>1</b>
1.1	Installation . . . . .	1
1.2	Usage . . . . .	2
1.3	License . . . . .	2
1.4	Contribution . . . . .	2
<b>2</b>	<b>User Manual</b>	<b>3</b>
2.1	Installation . . . . .	3
2.2	The KonfiguratorFuerOSMDaten Manual . . . . .	4
<b>3</b>	<b>Class Descriptions</b>	<b>5</b>
3.1	src . . . . .	5
<b>4</b>	<b>Indices and tables</b>	<b>103</b>
	<b>Python Module Index</b>	<b>105</b>
	<b>Index</b>	<b>107</b>



## KONFIGURATORFUEROSMDATEN

license MIT

Whether it's biking to college or driving to the supermarket, traffic affects us all. This product generates from geodata of the free project 'OpenStreetMap' (OSM) a numerical ranking of the attractiveness of geographic locations. A main focus is the configurability of the generation of this score. Traffic planners can easily generate the right data for their traffic forecasting models.



This project was developed in the context of the lecture "Praxis der Softwareentwicklung(PSE)" at the University KIT in 2022, the waterfall model with feedback was used to develop the project. For each of this phase you can find the documentation in the corresponding subfolder, in the [github](#), most of the documentation is in german.

For a more specific description of the code, check the `README.md` in the subfolder `pythoncode`.

### 1.1 Installation

We recommend installing KonfiguratorFuerOSMDaten using one of the available built distributions, for example using `pip` or `conda`:

```
$ here will stand sth. in the future
```

## 1.2 Usage

To use the application after installing it, simply run the program. For an description about how the application works check out the folder `Pflichtenheft`.

## 1.3 License

KonfiguratorFuerOSMDaten is licensed under the MIT [License](#).

## 1.4 Contribution

To contribute to the project:

1. Fork it (<https://github.com/LuposX/KonfiguratorFuerOSMDaten/fork>)
2. Create your feature branch (`git checkout -b feature/fooBar`)
3. Commit your changes (`git commit -am 'Add some fooBar'`)
4. Push to the branch (`git push origin feature/fooBar`)
5. Create a new Pull Request

## USER MANUAL

## 2.1 Installation

### 2.1.1 Installation for local development

To install KonfiguratorFuerOSMDaten for local development, clone the package from Github:

```
$ git clone https://github.com/LuposX/KonfiguratorFuerOSMDaten.git
$ cd KonfiguratorFuerOSMDaten
```

For development, use of a virtual environment is strongly recommended. For example using `venv`:

```
$ python3 -m PSE .
$ pip install -r requirements.txt
(PSE) $
```

Or using `conda`:

```
$ conda env create --file enviroment2.yml
$ conda activate PSE
(PSE) $
```

### 2.1.2 Testing KonfiguratorFuerOSMDaten

To test that the installed libraries are working correctly the tests in the `test` folder can be used:

```
$ cd pythoncode
$ cd tests
$ cd libraryTests
```

Each file is a test for one functionality of a library that is needed in our project, to test `.py` files:

```
$ python script_name.py
```

To test `.ipynb` files:

```
$ jupyter-lab
```

and run the files in `jupyter-lab`. If the libraries are correctly installed you shouldn't get any errors, when running a file.

## 2.2 The KonfiguratorFuerOSMDaten Manual

**Author**

Simon

**Version**

1.0

**Date**

Jan 12, 2023

**Copyright**

This work is licensed under a *MIT* license.

**Abstract**

This document explains how to use the KonfiguratorFuerOSMDaten Application.

### 2.2.1 Introduction

---

**Note:** Here will stand sth. in the future

---

For an description about how the application works check out the folder *Pflichtenheft*.



## CLASS DESCRIPTIONS

### 3.1 src

#### 3.1.1 src package

##### Subpackages

##### src.osm\_configurator package

##### Subpackages

##### src.osm\_configurator.control package

##### Submodules

##### src.osm\_configurator.control.aggregation\_controller module

**class** `AggregationController`(*model*)

Bases: `object`

The `AggregationController` is responsible for consistently forwarding requests to the model, regarding the aggregation-calculations and the aggregation methods of the currently selected project.

**`__init__`**(*model*)

Creates a new instance of the `AggregationController`, with an association to the model.

##### Parameters

**model** (`application_interface.IApplication`) – The interface which is used to communicate with the model.

**`get_aggregation_methods`**()

Returns a list of all aggregation methods that are available. This function returns all available aggregation methods, not just the ones that are active in the current project.

##### Returns

The list of the available aggregation methods

##### Return type

`list[aggregation_method_enum.AggregationMethod]`

### **is\_aggregation\_method\_active(*method*)**

Checks, whether an aggregation method is active in the currently selected project.

#### **Parameters**

**method** (`aggregation_method_enum.AggregationMethod`) – The aggregation method that is checked for.

#### **Returns**

True, if there is currently a project selected and the given aggregation method is active in it; False otherwise.

#### **Return type**

`bool`

### **set\_aggregation\_method\_active(*method*, *active*)**

Activates or deactivates an aggregation method (of the currently selected project). Activates the given method, if `active=True` and deactivates it otherwise.

#### **Parameters**

- **method** (`aggregation_method_enum.AggregationMethod`) – The aggregation method we want to deactivate/activate
- **active** (`bool`) – True, if we want to activate the given method; False, if we want to deactivate it.

#### **Returns**

True, if a project is currently selected and the aggregation method was (de-)activated successfully; False, otherwise.

#### **Return type**

`bool`

## **src.osm\_configurator.control.application\_controller module**

### **class ApplicationController**

Bases: `object`

The application controller is responsible for creating the model, the view and the control. It is the start of the application and boots everything up.

#### **main()**

Starts the application. This class method's only job is, to give control to an instance of the ApplicationController.

#### **\_\_init\_\_()**

Creates a new Application. It creates the view, the model and the control. It is responsible for starting everything up and to switch to the normal workflow of the application.

**src.osm\_configurator.control.calculation\_controller module****class CalculationController**(*model*)Bases: `object`

The CalculationController is responsible for forwarding requests to the model, regarding calculations. It may be used to gather information and to control the calculation-process of the currently selected project.

**\_\_init\_\_**(*model*)

Creates a new instance of the CalculationController, with an association to the model.

**Parameters**

**model** (`application_interface.IApplication`) – The interface which is used to communicate with the model.

**start\_calculations**(*starting\_phase*)

Starts the calculations in the given calculation phase in the currently selected project. The calculation process is split in different calculation phases. This function starts the calculation in a given phase.

**Parameters**

**starting\_phase** (`calculation_phase_enum.CalculationPhase`) – The phase, in which the calculation should start

**Returns**

The status of the calculation: `RUNNING`, if the calculation was started successfully. For details on the meaning of this return value, see `CalculationState`

**Return type**

`calculation_state_enum.CalculationState`

**get\_calculation\_state**()

Gives the current calculation state of the selected project.

**Returns**

Returns the current state of the calculation. For details see documentation of `CalculationState`.

**Return type**

`calculation_state_enum.CalculationState`

**get\_current\_calculation\_phase**()

Returns the calculation phase of the currently selected project.

**Returns**

The phase, that is currently running. `NONE`, if no phase is currently running.

**Return type**

`calculation_phase_enum.CalculationPhase`

**get\_current\_calculation\_process**()

Returns an approximation of the progress of the calculations in the currently selected project. The progress is given as a number between 0 and 1, where 0 indicates that the calculation has not started yet and 1 indicates, that the calculations are done.

**Returns**

The value of the approximation.

**Return type**

`float`

**cancel\_calculations()**

Cancels the calculations of the currently selected project. The calculation phase that is currently running will be stopped.

**Returns**

True, if the calculation was canceled successfully; False, otherwise.

**Return type**

`bool`

**src.osm\_configurator.control.category\_controller module****class CategoryController(model)**

Bases: `object`

The CategoryController is responsible for consistently forwarding requests to the model, regarding changes to the categories of the current project.

**\_\_init\_\_(model)**

Creates a new instance of the CategoryController, with an association to the model.

**Parameters**

**model** (`application_interface.IApplication`) – The interface which is used to communicate with the model.

**check\_conflicts\_in\_category\_configuration(path)**

Checks for a given file, if it is a valid category-file and checks, whether there are naming conflicts with the categories of the currently selected project.

**Parameters**

**path** (`pathlib.Path`) – The path to the category-file

**Returns**

True, if there is currently a project selected and there are no naming conflicts; False, otherwise.

**Return type**

`bool`

**import\_category\_configuration(path)**

Imports the given categories into the currently selected project. Adds the given categories to the category list of the project.

**Parameters**

**path** (`pathlib.Path`) – The path to the category file

**Returns**

True, if the categories were added successfully; False, if there is no project loaded, the category file is corrupted or the category file does not exist.

**Return type**

`bool`

**get\_list\_of\_categories()**

Returns the list of all categories, that are currently in the currently selected project.

**Returns**

A list of the categories of the project in no particular order.

**Return type**

`list[category.Category]`

**create\_category()**

Creates a new category in the currently selected project. A new category is added to the list of categories of the project. The category has empty properties, except for an arbitrary name. If the creation fails, none will be returned and there won't be a category added.

**Returns**

The newly created category, none if there was an error

**Return type**

*category.Category*

**delete\_category(category)**

Deletes the given category. Removes the given category from the list of categories of the currently selected project

**Parameters**

**category** (*category.Category*) – The category, to be deleted

**Returns**

True, if the category was deleted successfully; False, otherwise

**Return type**

*bool*

**get\_list\_of\_key\_recommendations(current\_input)**

Returns a list of recommended keys, based on the input that is already entered by the user.

**Parameters**

**current\_input** (*str*) – The input, that is currently written by the user.

**Returns**

A list of key recommendations, based on the current\_input.

**Return type**

*list[str]*

**get\_attractivities\_of\_category(category)**

Returns the attractivity attributes that are defined for the given category.

**Parameters**

**category** (*category.Category*) – The category, whose attractivities are of interest.

**Returns**

The list of attractivity attributes of the given category

**Return type**

*list[attractivity\_attribute.AttractivityAttribute]*

**src.osm\_configurator.control.control module****class Control**

Bases: *IControl*

This class provides a consistent interface for access to the control-package. It is a facade, to make access easy. The control manages the access to the module. That's why this interface should give access to all the features provided by the model.

This implementation of the interface IControl forwards all requests to other classes of this package. For details see the documentation of the corresponding functions.

### `__init__()`

Creates a new instance of Control, with a association to the model.

#### Parameters

**model** (`application_interface.IApplication`) – The interface which is used to communicate with the model.

### `get_list_of_passive_projects()`

Returns the list of (passive) projects, which are in the default project folder of the application.

#### Returns

The list of passive projects in the default project folder.

#### Return type

`list[passive_project.PassiveProject]`

### `load_project(path)`

Loads a project All relevant data of a project are verified and loaded in memory. All coming project-referring calls will be directed to the given project.

#### Parameters

**path** (`pathlib.Path`) – The path to the project folder of the project, to be loaded.

#### Returns

True, if the project was loaded successfully; False if an error occurred, while trying to load the project. An error happens, if the path is not pointing to a valid project folder or if the project has corrupted files.

#### Return type

`bool`

### `create_project(name, description, destination)`

Creates a new project with the given attributes and loads it. The model creates a new project folder at the given destination, all relevant files are generated and the project is loaded into memory.

#### Parameters

- **name** (`str`) – The name of the to-be-created project, may not contain any line-breaks.
- **description** (`str`) – The description of the to-be-created project. May contain line-breaks.
- **destination** (`pathlib.Path`) – The path to the location, where the project-folder of the project should be created.

#### Returns

True, if the project was created successfully; False if an error occurred. An error occurs, if the name of the project is not valid, if the destination-path is not valid or if the destination-location is already occupied.

#### Return type

`bool`

### `delete_passive_project(project)`

Deletes a project out of the default project folder.

#### Parameters

**project** (`passive_project.PassiveProject`) – The project, that is going to be deleted.

#### Returns

True, if the (passive) project has been deleted successfully; False otherwise: The project does not exist or the application has not the right permissions to delete the project.

**Return type**`bool`**save\_project()**

Saves the project. The currently selected project is stored on the disk. All progress made since the last saving are saved.

**Returns**

True, if the project was saved successfully; False if an error occurred, while attempting to save the project or when there is no project selected.

**Return type**`bool`**set\_current\_config\_phase(*config\_phase*)**

Stores the current configuration phase in the model.

**Parameters**

**config\_phase** (`config_phase_enum.ConfigPhase`) – The new configuration phase.

**Returns**

True, if setting the configuration phase was successful; False, otherwise.

**Return type**`bool`**get\_current\_config\_phase()**

Returns the configuration phase, that is currently stored in the model.

**Returns**

The configuration phase, that is currently stored in the model.

**Return type**`config_phase_enum.ConfigPhase`**is\_project\_loaded()**

Checks, whether any project is currently loaded/selected.

**Returns**

True, if a project is currently selected; False, otherwise.

**Return type**`bool`**set\_osm\_data\_reference(*path*)**

Sets the reference to the osm-data for the selected project. The reference contains the osm-data used in the calculations of the project. This method does not check if the given data is valid.

**Parameters**

**path** (`pathlib.Path`) – The reference to the osm-data

**Returns**

True, if the new reference was set successfully; False, if an error occurred while setting the reference.

**Return type**`bool`**get\_osm\_data\_reference()**

Returns the path to the osm-data, that is used in the currently selected project.

**Returns**

The path to the osm-data of the currently selected project.

**Return type**

`pathlib.Path`

**get\_cut\_out\_mode()**

Gets the method of how the geofilter shall cut out on the OSM-Data.

**Returns**

The cut-out-mode of the currently selected project.

**Return type**

`cut_out_mode_enum.CutOutMode`

**set\_cut\_out\_mode(mode)**

Sets the method of how the geofilter shall cut out on the OSM-Data.

**Parameters**

**mode** (`cut_out_mode_enum.CutOutMode`) – The mode, to be set

**Returns**

True, if the CutOutMode was set successfully; False, if an error occurred or no project is currently selected.

**Return type**

`bool`

**set\_cut\_out\_reference(path)**

Sets the reference to the cut-out file of the currently selected project. This file is later used to calculate the geofilter.

**Parameters**

**path** (`pathlib.Path`) – The path to the file containing the cut-out-geometries

**Returns**

True, if the reference was set successfully; False, if an error occurred. An error occurs, if no project is currently selected or if the given path is not valid or occupied.

**Return type**

`bool`

**get\_cut\_out\_reference()**

Gets the reference to the cut-out file of the currently selected project.

**Returns**

The current reference to the cut-out file.

**Return type**

`pathlib.Path`

**check\_conflicts\_in\_category\_configuration(path)**

Checks for a given file, if it is a valid category-file and checks, whether there are naming conflicts with the categories of the currently selected project.

**Parameters**

**path** (`pathlib.Path`) – The path to the category-file

**Returns**

True, if there is currently a project selected and there are no naming conflicts; False, otherwise.



**Return type**`bool`**import\_category\_configuration(*path*)**

Imports the given categories into the currently selected project. Adds the given categories to the category list of the project.

**Parameters**

**path** (`pathlib.Path`) – The path to the category file

**Returns**

True, if the categories were added successfully; False, if there is no project loaded, the category file is corrupted or the category file does not exist.

**Return type**`bool`**get\_list\_of\_categories()**

Returns the list of all categories, that are currently in the currently selected project.

**Returns**

A list of the categories of the project in no particular order.

**Return type**`list[category.Category]`**create\_category()**

Creates a new category in the currently selected project. A new category is added to the list of categories of the project. The category has empty properties, except for an arbitrary name. If the creation fails, none will be returned and there won't be a category added.

**Returns**

The newly created category, none if there was an error

**Return type**`category.Category`**delete\_category(*category*)**

Deletes the given category. Removes the given category from the list of categories of the currently selected project

**Parameters**

**category** (`category.Category`) – The category, to be deleted

**Returns**

True, if the category was deleted successfully; False, otherwise

**Return type**`bool`**get\_list\_of\_key\_recommendations(*current\_input*)**

Returns a list of recommended keys, based on the input that is already entered by the user.

**Parameters**

**current\_input** (`str`) – The input, that is currently written by the user.

**Returns**

A list of key recommendations, based on the `current_input`.

**Return type**`list[str]`

**get\_attractivities\_of\_category**(*category*)

Returns the attractivity attributes that are defined for the given category.

**Parameters**

**category** (*category.Category*) – The category, whose attractivities are of interest.

**Returns**

The list of attractivity attributes of the given category

**Return type**

*list[[attractivity\\_attribute.AttractivityAttribute](#)]*

**get\_aggregation\_methods**()

Returns a list of all aggregation methods that are available. This function returns all available aggregation methods, not just the ones that are active in the current project.

**Returns**

The list of the available aggregation methods

**Return type**

*list[[aggregation\\_method\\_enum.AggregationMethod](#)]*

**is\_aggregation\_method\_active**(*method*)

Checks, whether an aggregation method is active in the currently selected project.

**Parameters**

**method** (*aggregation\_method\_enum.AggregationMethod*) – The aggregation method that is checked for.

**Returns**

True, if there is currently a project selected and the given aggregation method is active in it;  
False otherwise.

**Return type**

*bool*

**set\_aggregation\_method\_active**(*method, active*)

Activates or deactivates an aggregation method (of the currently selected project). Activates the given method, if active=True and deactivates it otherwise.

**Parameters**

- **method** (*aggregation\_method\_enum.AggregationMethod*) – The aggregation method we want to deactivate/activate
- **active** (*bool*) – True, if we want to activate the given method; False, if we want to deactivate it.

**Returns**

True, if a project is currently selected and the aggregation method was (de-)activated successfully; False, otherwise.

**Return type**

*bool*

**start\_calculations**(*starting\_phase*)

Starts the calculations in the given calculation phase in the currently selected project. The calculation process is split in different calculation phases. This function starts the calculation in a given phase.

**Parameters**

**starting\_phase** (*calculation\_phase\_enum.CalculationPhase*) – The phase, in which the calculation should start

**Returns**

The status of the calculation: `RUNNING`, if the calculation was started successfully. For details on the meaning of this return value, see `CalculationState`

**Return type**

*calculation\_state\_enum.CalculationState*

**get\_calculation\_state()**

Gives the current calculation state of the selected project.

**Returns**

Returns the current state of the calculation. For details see documentation of `CalculationState`.

**Return type**

*calculation\_state\_enum.CalculationState*

**get\_current\_calculation\_phase()**

Returns the calculation phase of the currently selected project.

**Returns**

The phase, that is currently running. `NONE`, if no phase is currently running.

**Return type**

*calculation\_phase\_enum.CalculationPhase*

**get\_current\_calculation\_process()**

Returns an approximation of the progress of the calculations in the currently selected project. The progress is given as a number between 0 and 1, where 0 indicates that the calculation has not started yet and 1 indicates, that the calculations are done.

**Returns**

The value of the approximation.

**Return type**

`float`

**cancel\_calculations()**

Cancels the calculations of the currently selected project. The calculation phase that is currently running will be stopped.

**Returns**

True, if the calculation was canceled successfully; False, otherwise.

**Return type**

`bool`

**export\_project(path)**

Exports the currently selected project. Before it will be exported, the project will be saved. The folders and files of the project are copied to the given destination.

**Parameters**

**path** (*pathlib.Path*) – The place in storage, where the project should be exported to.

**Returns**

True, if the export was successful; False, if an error occurred: The path was not valid or occupied, there was not enough space in storage or there was no project selected.

**Return type**

`bool`

**export\_calculations(*path*)**

Exports the result of the calculations of the currently selected project. The folders and files regarding the results of the calculations are copied to the given destination.

**Parameters**

**path** (*pathlib.Path*) – The place in storage, where the results should be exported to.

**Returns**

True, if the export was successful; False, if an error occurred: The path was not valid or occupied, there was not enough space in storage, the calculations have not produced results yet or there was no project selected.

**Return type**

bool

**export\_configurations(*path*)**

Exports the category file of the currently selected project. A list of categories in the current project is stored at the given destination.

**Parameters**

**path** (*pathlib.Path*) – The place in storage, where the categories should be stored at.

**Returns**

True, if the export was successful; False, if an error occurred: The path was not valid or occupied, there was not enough space in storage or there was no project selected.

**Return type**

bool

**get\_project\_name()**

Gets the name of the currently selected project.

**Returns**

The name of the project

**Return type**

str

**set\_project\_name(*name*)**

Sets the name of the currently selected project

**Parameters**

**name** (*str*) – The new name of the project, may not contain line breaks.

**Returns**

True, if the name was changed successfully; False, if an error occurred: The name is not valid or no project was selected.

**Return type**

bool

**get\_project\_description()**

Gets the description of the currently selected project.

**Returns**

The description of the project

**Return type**

str

**set\_project\_description**(*description*)

Sets the description of the currently selected project.

**Parameters**

**description** (*str*) – The new description of the project, may contain line breaks.

**Returns**

True, if the description was changed successfully; False, otherwise.

**Return type**

`bool`

**get\_project\_default\_folder**()

Gets the project default folder. The project default folder is the folder, where projects are stored by default.

**Returns**

The path to the project default folder

**Return type**

`pathlib.Path`

**set\_project\_default\_folder**(*default\_folder*)

Sets the project default folder. The project default folder is the folder, where projects are stored by default. Projects of an old default folder will not be copied over.

**Parameters**

**default\_folder** (`pathlib.Path`) – The path to the new project default folder

**Returns**

True, if the default folder was set successfully; False if an error occurred: The path is not valid or occupied.

**Return type**

`bool`

**generate\_cut\_out\_map**()

Generates a map of the data of the currently selected project. Using the cut-out file of the project, this function creates a map as a html-file of the project. The path to the html-file is returned.

**Returns**

The path to the file, where the map is stored.

**Return type**

`pathlib.Path`

**get\_calculation\_visualization**()

Generates a graphic that visualizes the results of the calculations of the currently selected project.

**Returns**

The resulting visualization as axes of the matplotlib library.

**Return type**

`matplotlib.Axes`

**src.osm\_configurator.control.control\_interface module****class IControl**Bases: [ABC](#)

This class provides a consistent interface for access to the control-package. It is a facade, to make access easy. The control manages the access to the module. That's why this interface should give access to all the features provided by the model.

**abstract get\_list\_of\_passive\_projects()**

Returns the list of (passive) projects, which are in the default project folder of the application.

**Returns**

The list of passive projects in the default project folder.

**Return type**

[list](#)[\[](#)[passive\\_project.PassiveProject](#)[\]](#)

**abstract load\_project(path)**

Loads a project All relevant data of a project are verified and loaded in memory. All coming project-referring calls will be directed to the given project.

**Parameters**

**path** ([pathlib.Path](#)) – The path to the project folder of the project, to be loaded.

**Returns**

True, if the project was loaded successfully; False if an error occurred, while trying to load the project. An error happens, if the path is not pointing to a valid project folder or if the project has corrupted files.

**Return type**

[bool](#)

**abstract create\_project(name, description, destination)**

Creates a new project with the given attributes and loads it. The model creates a new project folder at the given destination, all relevant files are generated and the project is loaded into memory.

**Parameters**

- **name** ([str](#)) – The name of the to-be-created project, may not contain any line-breaks.
- **description** ([str](#)) – The description of the to-be-created project. May contain line-breaks.
- **destination** ([pathlib.Path](#)) – The path to the location, where the project-folder of the project should be created.

**Returns**

True, if the project was created successfully; False if an error occurred. An error occurs, if the name of the project is not valid, if the destination-path is not valid or if the destination-location is already occupied.

**Return type**

[bool](#)

**abstract delete\_passive\_project(project)**

Deletes a project out of the default project folder.

**Parameters**

**project** ([passive\\_project.PassiveProject](#)) – The project, that is going to be deleted.

**Returns**

True, if the (passive) project has been deleted successfully; False otherwise: The project does not exist or the application has not the right permissions to delete the project.

**Return type**

`bool`

**abstract save\_project()**

Saves the project. The currently selected project is stored on the disk. All progress made since the last saving are saved.

**Returns**

True, if the project was saved successfully; False if an error occurred, while attempting to save the project or when there is no project selected.

**Return type**

`bool`

**abstract set\_current\_config\_phase(*config\_phase*)**

Stores the current configuration phase in the model.

**Parameters**

**config\_phase** (`config_phase_enum.ConfigPhase`) – The new configuration phase.

**Returns**

True, if setting the configuration phase was successful; False, otherwise.

**Return type**

`bool`

**abstract get\_current\_config\_phase()**

Returns the configuration phase, that is currently stored in the model.

**Returns**

The configuration phase, that is currently stored in the model.

**Return type**

`config_phase_enum.ConfigPhase`

**abstract is\_project\_loaded()**

Checks, whether any project is currently loaded/selected.

**Returns**

True, if a project is currently selected; False, otherwise.

**Return type**

`bool`

**abstract set\_osm\_data\_reference(*path*)**

Sets the reference to the osm-data for the selected project. The reference contains the osm-data used in the calculations of the project. This method does not check if the given data is valid.

**Parameters**

**path** (`pathlib.Path`) – The reference to the osm-data

**Returns**

True, if the new reference was set successfully; False, if an error occurred while setting the reference.

**Return type**

`bool`

**abstract get\_osm\_data\_reference()**

Returns the path to the osm-data, that is used in the currently selected project.

**Returns**

The path to the osm-data of the currently selected project.

**Return type**

`pathlib.Path`

**abstract download\_osm\_data(path)**

Downloads osm-data The osm-data to be downloaded are defined by a geojson-file. The data is downloaded and the reference to the correct osm-files is stored.

**Parameters**

**path** (`pathlib.Path`) – The path to the geojson-file.

**Returns**

True on success, False otherwise

**Return type**

`bool`

**abstract get\_cut\_out\_mode()**

Gets the method of how the geofilter shall cut out on the OSM-Data.

**Returns**

The cut-out-mode of the currently selected project.

**Return type**

`cut_out_mode_enum.CutOutMode`

**abstract set\_cut\_out\_mode(mode)**

Sets the method of how the geofilter shall cut out on the OSM-Data.

**Parameters**

**mode** (`cut_out_mode_enum.CutOutMode`) – The mode, to be set

**Returns**

True, if the CutOutMode was set successfully; False, if an error occurred or no project is currently selected.

**Return type**

`bool`

**abstract set\_cut\_out\_reference(path)**

Sets the reference to the cut-out file of the currently selected project. This file is later used to calculate the geofilter.

**Parameters**

**path** (`pathlib.Path`) – The path to the file containing the cut-out-geometries

**Returns**

True, if the reference was set successfully; False, if an error occurred. An error occurs, if no project is currently selected or if the given path is not valid or occupied.

**Return type**

`bool`

**abstract get\_cut\_out\_reference()**

Gets the reference to the cut-out file of the currently selected project.



**Returns**

The current reference to the cut-out file.

**Return type**

`pathlib.Path`

**abstract check\_conflicts\_in\_category\_configuration(*path*)**

Checks for a given file, if it is a valid category-file and checks, whether there are naming conflicts with the categories of the currently selected project.

**Parameters**

**path** (`pathlib.Path`) – The path to the category-file

**Returns**

True, if there is currently a project selected and there are no naming conflicts; False, otherwise.

**Return type**

`bool`

**abstract import\_category\_configuration(*path*)**

Imports the given categories into the currently selected project. Adds the given categories to the category list of the project.

**Parameters**

**path** (`pathlib.Path`) – The path to the category file

**Returns**

True, if the categories were added successfully; False, if there is no project loaded, the category file is corrupted or the category file does not exist.

**Return type**

`bool`

**abstract get\_list\_of\_categories()**

Returns the list of all categories, that are currently in the currently selected project.

**Returns**

A list of the categories of the project in no particular order.

**Return type**

`list[category.Category]`

**abstract create\_category()**

Creates a new category in the currently selected project. A new category is added to the list of categories of the project. The category has empty properties, except for an arbitrary name. If the creation fails, none will be returned and there won't be a category added.

**Returns**

The newly created category, none if there was an error

**Return type**

`category.Category`

**abstract delete\_category(*category*)**

Deletes the given category. Removes the given category from the list of categories of the currently selected project

**Parameters**

**category** (`category.Category`) – The category, to be deleted

**Returns**

True, if the category was deleted successfully; False, otherwise

**Return type**

`bool`

**abstract get\_list\_of\_key\_recommendations**(*current\_input*)

Returns a list of recommended keys, based on the input that is already entered by the user.

**Parameters**

**current\_input** (*str*) – The input, that is currently written by the user.

**Returns**

A list of key recommendations, based on the *current\_input*.

**Return type**

`list[str]`

**abstract get\_attractivities\_of\_category**(*category*)

Returns the attractivity attributes that are defined for the given category.

**Parameters**

**category** (*category.Category*) – The category, whose attractivities are of interest.

**Returns**

The list of attractivity attributes of the given category

**Return type**

`list[attractivity_attribute.AttractivityAttribute]`

**abstract get\_aggregation\_methods**()

Returns a list of all aggregation methods that are available. This function returns all available aggregation methods, not just the ones that are active in the current project.

**Returns**

The list of the available aggregation methods

**Return type**

`list[aggregation_method_enum.AggregationMethod]`

**abstract is\_aggregation\_method\_active**(*method*)

Checks, whether an aggregation method is active in the currently selected project.

**Parameters**

**method** (*aggregation\_method\_enum.AggregationMethod*) – The aggregation method that is checked for.

**Returns**

True, if there is currently a project selected and the given aggregation method is active in it;  
False otherwise.

**Return type**

`bool`

**abstract set\_aggregation\_method\_active**(*method*, *active*)

Activates or deactivates an aggregation method (of the currently selected project). Activates the given method, if *active*=True and deactivates it otherwise.

**Parameters**

- **method** (*aggregation\_method\_enum.AggregationMethod*) – The aggregation method we want to deactivate/activate
- **active** (*bool*) – True, if we want to activate the given method; False, if we want to deactivate it.

**Returns**

True, if a project is currently selected and the aggregation method was (de-)activated successfully; False, otherwise.

**Return type**

`bool`

**abstract start\_calculations**(*starting\_phase*)

Starts the calculations in the given calculation phase in the currently selected project. The calculation process is split in different calculation phases. This function starts the calculation in a given phase.

**Parameters**

**starting\_phase** (`calculation_phase_enum.CalculationPhase`) – The phase, in which the calculation should start

**Returns**

The status of the calculation: `RUNNING`, if the calculation was started successfully. For details on the meaning of this return value, see `CalculationState`

**Return type**

`calculation_state_enum.CalculationState`

**abstract get\_calculation\_state**()

Gives the current calculation state of the selected project.

**Returns**

Returns the current state of the calculation. For details see documentation of `CalculationState`.

**Return type**

`calculation_state_enum.CalculationState`

**abstract get\_current\_calculation\_phase**()

Returns the calculation phase of the currently selected project.

**Returns**

The phase, that is currently running. `NONE`, if no phase is currently running.

**Return type**

`calculation_phase_enum.CalculationPhase`

**abstract get\_current\_calculation\_process**()

Returns an approximation of the progress of the calculations in the currently selected project. The progress is given as a number between 0 and 1, where 0 indicates that the calculation has not started yet and 1 indicates, that the calculations are done.

**Returns**

The value of the approximation.

**Return type**

`float`

**abstract cancel\_calculations**()

Cancels the calculations of the currently selected project. The calculation phase that is currently running will be stopped.

**Returns**

True, if the calculation was canceled successfully; False, otherwise.

**Return type**

`bool`

**abstract export\_project**(*path*)

Exports the currently selected project. Before it will be exported, the project will be saved. The folders and files of the project are copied to the given destination.

**Parameters**

**path** (*pathlib.Path*) – The place in storage, where the project should be exported to.

**Returns**

True, if the export was successful; False, if an error occurred: The path was not valid or occupied, there was not enough space in storage or there was no project selected.

**Return type**

bool

**abstract export\_calculations**(*path*)

Exports the result of the calculations of the currently selected project. The folders and files regarding the results of the calculations are copied to the given destination.

**Parameters**

**path** (*pathlib.Path*) – The place in storage, where the results should be exported to.

**Returns**

True, if the export was successful; False, if an error occurred: The path was not valid or occupied, there was not enough space in storage, the calculations have not produced results yet or there was no project selected.

**Return type**

bool

**abstract export\_configurations**(*path*)

Exports the category file of the currently selected project. A list of categories in the current project is stored at the given destination.

**Parameters**

**path** (*pathlib.Path*) – The place in storage, where the categories should be stored at.

**Returns**

True, if the export was successful; False, if an error occurred: The path was not valid or occupied, there was not enough space in storage or there was no project selected.

**Return type**

bool

**abstract export\_cut\_out\_map**(*path*)

Exports the map generated by the cut-out configuration.

**Parameters**

**path** (*pathlib.Path*) – The place in storage, where the cut-out-map should be stored at.

**Returns**

True, if the export was successful; False, if an error occurred: The path was not valid or occupied, there was not enough space in storage, the application wasn't able to create the map or there was no project selected.

**Return type**

bool

**abstract get\_project\_name**()

Gets the name of the currently selected project.

**Returns**

The name of the project

**Return type**

`str`

**abstract set\_project\_name**(*name*)

Sets the name of the currently selected project

**Parameters**

**name** (`str`) – The new name of the project, may not contain line breaks.

**Returns**

True, if the name was changed successfully; False, if an error occurred: The name is not valid or no project was selected.

**Return type**

`bool`

**abstract get\_project\_description**()

Gets the description of the currently selected project.

**Returns**

The description of the project

**Return type**

`str`

**abstract set\_project\_description**(*description*)

Sets the description of the currently selected project.

**Parameters**

**description** (`str`) – The new description of the project, may contain line breaks.

**Returns**

True, if the description was changed successfully; False, otherwise.

**Return type**

`bool`

**abstract get\_project\_default\_folder**()

Gets the project default folder. The project default folder is the folder, where projects are stored by default.

**Returns**

The path to the project default folder

**Return type**

`pathlib.Path`

**abstract set\_project\_default\_folder**(*default\_folder*)

Sets the project default folder. The project default folder is the folder, where projects are stored by default. Projects of an old default folder will not be copied over.

**Parameters**

**default\_folder** (`pathlib.Path`) – The path to the new project default folder

**Returns**

True, if the default folder was set successfully; False if an error occurred: The path is not valid or occupied.

**Return type**

`bool`

**abstract generate\_cut\_out\_map()**

Generates a map of the data of the currently selected project. Using the cut-out file of the project, this function creates a map as a html-file of the project. The path to the html-file is returned.

**Returns**

The path to the file, where the map is stored.

**Return type**

`pathlib.Path`

**abstract get\_calculation\_visualization()**

Generates a graphic that visualizes the results of the calculations of the currently selected project.

**Returns**

The resulting visualization as axes of the matplotlib library.

**Return type**

`matplotlib.Axes`

**src.osm\_configurator.control.cut\_out\_controller module**

**class CutOutController(*model*)**

Bases: `object`

The CutOutController is responsible for consistently forwarding requests to the model, concerning the cut-out filter of the currently selected project.

**\_\_init\_\_(*model*)**

Creates a new instance of the CutOutController, with an association to the model.

**Parameters**

**model** (`application_interface.IApplication`) – The interface which is used to communicate with the model.

**get\_cut\_out\_mode()**

Gets the method of how the geofilter shall cut out on the OSM-Data.

**Returns**

The cut-out-mode of the currently selected project.

**Return type**

`cut_out_mode_enum.CutOutMode`

**set\_cut\_out\_mode(*mode*)**

Sets the method of how the geofilter shall cut out on the OSM-Data.

**Parameters**

**mode** (`cut_out_mode_enum.CutOutMode`) – The mode, to be set

**Returns**

True, if the CutOutMode was set successfully; False, if an error occurred or no project is currently selected.

**Return type**

`bool`

**set\_cut\_out\_reference(*path*)**

Sets the reference to the cut-out file of the currently selected project. This file is later used to calculate the geofilter.

**Parameters**

**path** (`pathlib.Path`) – The path to the file containing the cut-out-geometries

**Returns**

True, if the reference was set successfully; False, if an error occurred. An error occurs, if no project is currently selected or if the given path is not valid or occupied.

**Return type**

`bool`

**get\_cut\_out\_reference()**

” Gets the reference to the cut-out file of the currently selected project.

**Returns**

The current reference to the cut-out file.

**Return type**

`pathlib.Path`

**src.osm\_configurator.control.data\_visualization\_controller module****class DataVisualizationController(model)**

Bases: `object`

The DataVisualizationController is responsible for forwarding requests to the model, regarding the visualization of data from the model.

**\_\_init\_\_(model)**

Creates a new instance of the DataVisualizationController, with an association to the model.

**Parameters**

**model** (`application_interface.IApplication`) – The interface which is used to communicate with the model.

**generate\_cut\_out\_map()**

Generates a map of the data of the currently selected project. Using the cut-out file of the project, this function creates a map as a html-file of the project. The path to the html-file is returned.

**Returns**

The path to the file, where the map is stored.

**Return type**

`pathlib.Path`

**get\_calculation\_visualization()**

Generates a graphic that visualizes the results of the calculations of the currently selected project.

**Returns**

The resulting visualization as axes of the matplotlib library.

**Return type**

`matplotlib.Axes`

## src.osm\_configurator.control.export\_controller module

**class** `ExportController`(*model*)

Bases: `object`

The ExportController forwards requests to the model, regarding the export of information as files, in the currently selected project.

**\_\_init\_\_**(*model*)

Creates a new instance of the ExportController, with an association to the model.

**Parameters**

**model** (`application_interface.IApplication`) – The interface which is used to communicate with the model.

**export\_project**(*path*)

Exports the currently selected project. Before it will be exported, the project will be saved. The folders and files of the project are copied to the given destination.

**Parameters**

**path** (`pathlib.Path`) – The place in storage, where the project should be exported to.

**Returns**

True, if the export was successful; False, if an error occurred: The path was not valid or occupied, there was not enough space in storage or there was no project selected.

**Return type**

`bool`

**export\_calculations**(*path*)

Exports the result of the calculations of the currently selected project. The folders and files regarding the results of the calculations are copied to the given destination.

**Parameters**

**path** (`pathlib.Path`) – The place in storage, where the results should be exported to.

**Returns**

True, if the export was successful; False, if an error occurred: The path was not valid or occupied, there was not enough space in storage, the calculations have not produced results yet or there was no project selected.

**Return type**

`bool`

**export\_configurations**(*path*)

Exports the category file of the currently selected project. A list of categories in the current project is stored at the given destination.

**Parameters**

**path** (`pathlib.Path`) – The place in storage, where the categories should be stored at.

**Returns**

True, if the export was successful; False, if an error occurred: The path was not valid or occupied, there was not enough space in storage or there was no project selected.

**Return type**

`bool`

**export\_cut\_out\_map**(*path*)

Exports the map generated by the cut-out configuration.



**Parameters**

**path** (*pathlib.Path*) – The place in storage, where the cut-out-map should be stored at.

**Returns**

True, if the export was successful; False, if an error occurred: The path was not valid or occupied, there was not enough space in storage, the application wasn't able to create the map or there was no project selected.

**Return type**

bool

**src.osm\_configurator.control.osm\_data\_controller module**

**class** OSMDDataController(*model*)

Bases: *object*

The OSMDDataController is responsible for consistently forwarding requests regarding the OSM-data of the currently selected project.

**\_\_init\_\_**(*model*)

Creates a new instance of the OSMDDataController, with an association to the model.

**Parameters**

**model** (*application\_interface.IApplication*) – The interface which is used to communicate with the model.

**set\_osm\_data\_reference**(*path*)

Sets the reference to the osm-data for the selected project. The reference contains the osm-data used in the calculations of the project. This method does not check if the given data is valid.

**Parameters**

**path** (*pathlib.Path*) – The reference to the osm-data

**Returns**

True, if the new reference was set successfully; False, if an error occurred while setting the reference.

**Return type**

bool

**get\_osm\_data\_reference**()

Returns the path to the osm-data, that is used in the currently selected project.

**Returns**

The path to the osm-data of the currently selected project.

**Return type**

*pathlib.Path*

**download\_osm\_data**(*path*)

Downloads osm-data The osm-data to be downloaded are defined by a geojson-file. The data is downloaded and the reference to the correct osm-files is stored.

**Parameters**

**path** (*pathlib.Path*) – The path to the geojson-file.

**Returns**

True on success, False otherwise

**Return type**`bool`**src.osm\_configurator.control.project\_controller module****class ProjectController(*model*)**Bases: `object`

The ProjectController is responsible for consistently forwarding requests regarding the project management to the model. It is responsible for managing, saving, loading, deleting and creating projects.

**`__init__`(*model*)**

Creates a new instance of the ProjectController, with an association to the model.

**Parameters**

**model** (`application_interface.IApplication`) – The interface which is used to communicate with the model.

**`get_list_of_passive_projects`()**

Returns the list of (passive) projects, which are in the default project folder of the application.

**Returns**

The list of passive projects in the default project folder.

**Return type**`list[passive_project.PassiveProject]`**`load_project`(*path*)**

Loads a project All relevant data of a project are verified and loaded in memory. All coming project-referring calls will be directed to the given project.

**Parameters**

**path** (`pathlib.Path`) – The path to the project folder of the project, to be loaded.

**Returns**

True, if the project was loaded successfully; False if an error occurred, while trying to load the project. An error happens, if the path is not pointing to a valid project folder or if the project has corrupted files.

**Return type**`bool`**`create_project`(*name, description, destination*)**

Creates a new project with the given attributes and loads it. The model creates a new project folder at the given destination, all relevant files are generated and the project is loaded into memory.

**Parameters**

- **name** (`str`) – The name of the to-be-created project, may not contain any line-breaks.
- **description** (`str`) – The description of the to-be-created project. May contain line-breaks.
- **destination** (`pathlib.Path`) – The path to the location, where the project-folder of the project should be created.

**Returns**

True, if the project was created successfully; False if an error occurred. An error occurs, if the name of the project is not valid, if the destination-path is not valid or if the destination-location is already occupied.

**Return type**`bool`**delete\_passive\_project**(*project*)

Deletes a project out of the default project folder.

**Parameters**

**project** (`passive_project.PassiveProject`) – The project, that is going to be deleted.

**Returns**

True, if the (passive) project has been deleted successfully; False otherwise: The project does not exist or the application has not the right permissions to delete the project.

**Return type**`bool`**save\_project**()

Saves the project. The currently selected project is stored on the disk. All progress made since the last saving are saved.

**Returns**

True, if the project was saved successfully; False if an error occurred, while attempting to save the project or when there is no project selected.

**Return type**`bool`**set\_current\_config\_phase**(*config\_phase*)

Stores the current configuration phase in the model.

**Parameters**

**config\_phase** (`config_phase_enum.ConfigPhase`) – The new configuration phase.

**Returns**

True, if setting the configuration phase was successful; False, otherwise.

**Return type**`bool`**get\_current\_config\_phase**()

Returns the configuration phase, that is currently stored in the model.

**Returns**

The configuration phase, that is currently stored in the model.

**Return type**`config_phase_enum.ConfigPhase`**is\_project\_loaded**()

Checks, whether any project is currently loaded/selected.

**Returns**

True, if a project is currently selected; False, otherwise.

**Return type**`bool`

**src.osm\_configurator.control.settings\_controller module****class SettingsController**(*model*)Bases: `object`

The SettingsController is responsible for forwarding requests to the model, regarding the settings of the application and the currently selected project.

**\_\_init\_\_**(*model*)

Creates a new instance of the SettingsController, with an association to the model.

**Parameters**

**model** (`application_interface.IApplication`) – The interface which is used to communicate with the model.

**get\_project\_name**()

Gets the name of the currently selected project.

**Returns**

The name of the project

**Return type**

`str`

**set\_project\_name**(*name*)

Sets the name of the currently selected project

**Parameters**

**name** (`str`) – The new name of the project, may not contain line breaks.

**Returns**

True, if the name was changed successfully; False, if an error occurred: The name is not valid or no project was selected.

**Return type**

`bool`

**get\_project\_description**()

Gets the description of the currently selected project.

**Returns**

The description of the project

**Return type**

`str`

**set\_project\_description**(*description*)

Sets the description of the currently selected project.

**Parameters**

**description** (`str`) – The new description of the project, may contain line breaks.

**Returns**

True, if the description was changed successfully; False, otherwise.

**Return type**

`bool`

**get\_project\_default\_folder**()

Gets the project default folder. The project default folder is the folder, where projects are stored by default.

**Returns**

The path to the project default folder

**Return type**

`pathlib.Path`

**set\_project\_default\_folder**(*default\_folder*)

Sets the project default folder. The project default folder is the folder, where projects are stored by default. Projects of an old default folder will not be copied over.

**Parameters**

**default\_folder** (`pathlib.Path`) – The path to the new project default folder

**Returns**

True, if the default folder was set successfully; False if an error occurred: The path is not valid or occupied.

**Return type**

`bool`

**Module contents**

**src.osm\_configurator.model package**

**Subpackages**

**src.osm\_configurator.model.application package**

**Submodules**

**src.osm\_configurator.model.application.application module**

**class Application**

Bases: *IApplication*

The IApplication job, is to provide the functionality the application needs.

**\_\_init\_\_()**

Creates a new instance of the application\_interface.Application.

**get\_passive\_project\_list()**

Returns the list of all passive project in the current project default folder.

**Returns**

The list of the passive projects.

**Return type**

`list[passive_project.PassiveProject]`

**get\_key\_recommendation**(*input*)

Creates recommendations based on user input

**Parameters**

**input** (`str`) – The input from which to generate suggestions.

**Returns**

Returns a list of strings containing the recommendations depending on the input.

**Return type**

`list[str]`

**create\_project**(*name, description, destination*)

This method creates a new project with a name, a description and saves it at a given destination.

**Parameters**

- **name** (*str*) – The name of the new project.
- **description** (*str*) – The description of the new project.
- **destination** (*pathlib.Path*) – The path, where the new project should be saved.

**Returns**

True if create\_project completed successfully, otherwise false.

**Return type**

`bool`

**load\_project**(*path*)

This method loads an existing project. This project can be internal or external ones. The path is pointing towards the folder, where the project is saved.

**Parameters**

**path** (*pathlib.Path*) – The path of the project, to be loaded.

**Returns**

True if loading the project is working, otherwise false.

**Return type**

`bool`

**start\_calculation**(*calculation\_phase*)

This method is to start the calculation (after the configuration is finished).

**Parameters**

**calculation\_phase** (*calculation\_phase\_enum.CalculationPhase*) – The calculation phase, where the calculation shall start.

**Returns**

The calculation state where the calculation started. Can be an error state, so signify an error, that prevents the start of the calculation.

**Return type**

*CalculationState*

**get\_osm\_data**()

Gives back the path pointing towards the OSM data file.

**Returns**

The path pointing towards the OSM data.

**Return type**

*pathlib.Path*

**set\_osm\_data**(*osm\_data*)

Edits the path pointing towards the OSM data file.

**Parameters**

**osm\_data** (*pathlib.Path*) – The new path towards the osm data file.

**Returns**

True if changing the path works, otherwise false.

**Return type**

`bool`

**get\_all\_aggregation\_methods()**

Gives back a List of all possible aggregation methods.

**Returns**

A list containing all aggregation methods.

**Return type**

`list[aggregation_method_enum.AggregationMethod]`

**is\_aggregation\_method\_active(method)**

Checks, if a given aggregation method is active.

**Parameters**

**method** (`aggregation_method_enum.AggregationMethod`) – The method, which is to be checked.

**Returns**

True if the aggregation method is active, otherwise false.

**Return type**

`bool`

**set\_aggregation\_method\_active(method, active)**

Changes the aggregation method from active to inactive and vice versa. If an already active aggregation method should be activated, it stays active. The same applies to inactive aggregation methods, which should be deactivated.

**Parameters**

- **method** (`aggregation_method_enum.AggregationMethod`) – The method, which state should be changed.
- **active** (`bool`) – This is the new state of the aggregation method.

**Returns**

True if changing the state works, otherwise false.

**Return type**

`bool`

**get\_cut\_out\_mode()**

Gives back the used cut-out mode.

**Returns**

The used cut-out mode.

**Return type**

`cut_out_mode_enum.CutOutMode`

**set\_cut\_out\_mode(new\_cut\_out\_mode)**

Changes the cut-out mode used during the reduction phase in the calculation.

**Parameters**

**new\_cut\_out\_mode** (`cut_out_mode_enum.CutOutMode`) – The new cut-out mode for the calculation.

**Returns**

True if changing the cut-out mode works, otherwise false.

**Return type**

`bool`

**get\_cut\_out\_path()**

Gives back the path pointing towards the cut-out file.

**Returns**

The path pointing towards the cut-out.

**Return type**

`pathlib.Path`

**set\_cut\_out\_path(path)**

Changes the path pointing towards the cut-out file.

**Parameters**

**path** (`pathlib.Path`) – The new path, towards a cut-out file.

**Returns**

True if changing the cut-out path works, otherwise false.

**Return type**

`bool`

**get\_category(index)**

Gets a category based on the index.

**Parameters**

**index** (`int`) – Index in the categories-list, that will be returned.

**Returns**

The Category we wanted, NONE if the index is out of bounds of the list.

**Return type**

`category.Category`

**get\_categories()**

Getter for all the Categories.

**Returns**

List of the chosen categories.

**Return type**

`list[category.Category]`

**create\_category()**

Creates a new category, that will be empty.

**Returns**

The newly created category.

**Return type**

`category.Category`

**remove\_category(category)**

Removes the given category from the categories list, if element is inside the List.

**Parameters**

**category** (`category.Category`) – Category that will be removed.



**Returns**

True, if the element was removed correctly, else false.

**Return type**

`bool`

**override\_categories**(*new\_category\_list*)

Overwrites the list of categories with the given list, if both lists are not identical.

**Parameters**

**new\_category\_list** (*list*[`category.Category`]) – List of categories, that will overwrite the already existing list.

**Returns**

True, if the replacement was successful, else false.

**Return type**

`bool`

**merge\_categories**(*category\_input\_list*)

Merges the existing category list with the given list if both lists are not identical. If two categories conflict in their name, the newer category will be used.

**Parameters**

**category\_input\_list** (*list*[`category.Category`]) – New list of categories that will be merged into the existing list.

**Returns**

True, if the merging was successful, else False.

**Return type**

`bool`

**create\_map**(*cut\_out*)

This method to create a map from to given cut-out.

**Parameters**

**cut\_out** (`cut_out_configuration.CutOutConfiguration`) – The cut-out configuration from which the map should be created.

**Returns**

True if creating the map works, otherwise false.

**Return type**

`bool`

**create\_boxplot**(*data*)

This method is to visualize the data by creating a boxplot. It is used to visualize the calculated end result via a boxplot.

**Parameters**

**data** (`matplotlib.axes.Axes`) – A plot of the data which we want to visualize.

**Returns**

True if creating the boxplot works, otherwise false.

**Return type**

`bool`

**get\_location**()

Getter for the location of the active project on the disk.

**Returns**

The location of the active project

**Return type**

`pathlib.Path`

**set\_name(*new\_name*)**

This method changes the name of the project.

**Parameters**

**new\_name** (*str*) – The new name of the project

**Returns**

true if change was successful, false else

**Return type**

`bool`

**get\_name()**

This method returns the name of the project.

**Returns**

name of the project

**Return type**

`str`

**set\_description(*new\_description*)**

This method changes the description of the project.

**Parameters**

**new\_description** (*str*) – The new description of the project

**Returns**

true if change successful, false else

**Return type**

`bool`

**get\_description()**

This method returns the description of the project.

**Returns**

The description of the project

**Return type**

`str`

**export\_project(*path*)**

Exports the whole project to the given path.

**Parameters**

**path** (*pathlib.Path*) – The path where the project shall be exported to

**Returns**

true, if export was successful, otherwise false.

**Return type**

`bool`

**export\_configuration(*path*)**

Exports the configuration to the given path. More specific, exports all the categories and their configurations, to the given path.

**Parameters**

**path** (*pathlib.Path*) – The path where the configurations shall be exported to

**Returns**

true, if export was successful, otherwise false.

**Return type**

bool

**export\_calculation(path)**

Exports the results of the calculation to the given path. Whereby the calculation are a folder with all the different results from each calculation step in it.

**Parameters**

**path** (*pathlib.Path*) – The path where the results of the calculation shall be exported to

**Returns**

true, if export was successful, otherwise false.

**Return type**

bool

**export\_map(path)**

Exports an HTML-Data with the map in it, to the given path.

**Parameters**

**path** (*pathlib.Path*) – The path, where the map shall be exported to

**Returns**

true, if export was successful, otherwise false.

**Return type**

bool

**src.osm\_configurator.model.application.application\_interface module**

**class IApplication**

Bases: ABC

The IApplication job, is to provide the functionality the application needs.

**abstract get\_passive\_project\_list()**

Returns the list of all passive project in the current project default folder.

**Returns**

The list of the passive projects.

**Return type**

list[*passive\_project.PassiveProject*]

**abstract get\_key\_recommendation(input)**

Creates recommendations based on user input

**Parameters**

**input** (*str*) – The input from which to generate suggestions.

**Returns**

Returns a list of strings containing the recommendations depending on the input.

**Return type**

list[str]

**abstract create\_project**(*name, description, destination*)

This method creates a new project with a name, a description and saves it at a given destination.

**Parameters**

- **name** (*str*) – The name of the new project.
- **description** (*str*) – The description of the new project.
- **destination** (*pathlib.Path*) – The path, where the new project should be saved.

**Returns**

True if create\_project completed successfully, otherwise false.

**Return type**

*bool*

**abstract load\_project**(*path*)

This method loads an existing project. This project can be internal or external ones. The path is pointing towards the folder, where the project is saved.

**Parameters**

**path** (*pathlib.Path*) – The path of the project, to be loaded.

**Returns**

True if loading the project is working, otherwise false.

**Return type**

*bool*

**abstract start\_calculation**(*calculation\_phase*)

This method is to start the calculation (after the configuration is finished).

**Parameters**

**calculation\_phase** (*calculation\_phase\_enum.CalculationPhase*) – The calculation phase, where the calculation shall start.

**Returns**

The calculation state where the calculation started. Can be an error state, so signify an error, that prevents the start of the calculation.

**Return type**

*CalculationState*

**abstract get\_osm\_data**()

Gives back the path pointing towards the OSM data file.

**Returns**

The path pointing towards the OSM data.

**Return type**

*pathlib.Path*

**abstract set\_osm\_data**(*osm\_data*)

Edits the path pointing towards the OSM data file.

**Parameters**

**osm\_data** (*pathlib.Path*) – The new path towards the osm data file.

**Returns**

True if changing the path works, otherwise false.

**Return type**

`bool`

**abstract get\_all\_aggregation\_methods()**

Gives back a List of all possible aggregation methods.

**Returns**

A list containing all aggregation methods.

**Return type**

`list[aggregation_method_enum.AggregationMethod]`

**abstract is\_aggregation\_method\_active(method)**

Checks, if a given aggregation method is active.

**Parameters**

**method** (`aggregation_method_enum.AggregationMethod`) – The method, which is to be checked.

**Returns**

True if the aggregation method is active, otherwise false.

**Return type**

`bool`

**abstract set\_aggregation\_method\_active(method, active)**

Changes the aggregation method from active to inactive and vice versa. If an already active aggregation method should be activated, it stays active. The same applies to inactive aggregation methods, which should be deactivated.

**Parameters**

- **method** (`aggregation_method_enum.AggregationMethod`) – The method, which state should be changed.
- **active** (`bool`) – This is the new state of the aggregation method.

**Returns**

True if changing the state works, otherwise false.

**Return type**

`bool`

**abstract get\_cut\_out\_mode()**

Gives back the used cut-out mode.

**Returns**

The used cut-out mode.

**Return type**

`cut_out_mode_enum.CutOutMode`

**abstract set\_cut\_out\_mode(new\_cut\_out\_mode)**

Changes the cut-out mode used during the reduction phase in the calculation.

**Parameters**

**new\_cut\_out\_mode** (`cut_out_mode_enum.CutOutMode`) – The new cut-out mode for the calculation.

**Returns**

True if changing the cut-out mode works, otherwise false.

**Return type**

`bool`

**abstract get\_cut\_out\_path()**

Gives back the path pointing towards the cut-out file.

**Returns**

The path pointing towards the cut-out.

**Return type**

`pathlib.Path`

**abstract set\_cut\_out\_path(path)**

Changes the path pointing towards the cut-out file.

**Parameters**

**path** (`pathlib.Path`) – The new path, towards a cut-out file.

**Returns**

True if changing the cut-out path works, otherwise false.

**Return type**

`bool`

**abstract get\_category(index)**

Gets a category based on the index.

**Parameters**

**index** (`int`) – Index in the categories-list, that will be returned.

**Returns**

The Category we wanted, NONE if the index is out of bounds of the list.

**Return type**

`category.Category`

**abstract get\_categories()**

Getter for all the Categories.

**Returns**

List of the chosen categories.

**Return type**

`list[category.Category]`

**abstract create\_category()**

Creates a new category, that will be empty.

**Returns**

The newly created category.

**Return type**

`category.Category`

**abstract remove\_category(category)**

Removes the given category from the categories list, if element is inside the List.

**Parameters**

**category** (`category.Category`) – Category that will be removed.

**Returns**

True, if the element was removed correctly, else false.

**Return type**`bool`**abstract override\_categories**(*new\_category\_list*)

Overwrites the list of categories with the given list, if both lists are not identical.

**Parameters**

**new\_category\_list** (*list*[`category.Category`]) – List of categories, that will overwrite the already existing list.

**Returns**

True, if the replacement was successful, else false.

**Return type**`bool`**abstract merge\_categories**(*category\_input\_list*)

Merges the existing category list with the given list if both lists are not identical. If two categories conflict in their name, the newer category will be used.

**Parameters**

**category\_input\_list** (*list*[`category.Category`]) – New list of categories that will be merged into the existing list.

**Returns**

True, if the merging was successful, else False.

**Return type**`bool`**abstract create\_map**(*cut\_out*)

This method to create a map from to given cut-out.

**Parameters**

**cut\_out** (`cut_out_configuration.CutOutConfiguration`) – The cut-out configuration from which the map should be created.

**Returns**

True if creating the map works, otherwise false.

**Return type**`bool`**abstract create\_boxplot**(*data*)

This method is to visualize the data by creating a boxplot. It is used to visualize the calculated end result via a boxplot.

**Parameters**

**data** (`matplotlib.axes.Axes`) – A plot of the data which we want to visualize.

**Returns**

True if creating the boxplot works, otherwise false.

**Return type**`bool`**abstract get\_location**()

Getter for the location of the active project on the disk.

**Returns**

The location of the active project

**Return type**

`pathlib.Path`

**get\_default\_location()**

Gives back the path pointing towards the project.

**Returns**

Returns the path of the default location.

**Return type**

`pathlib.Path`

**set\_default\_location(*new\_location*)**

Sets the default path pointing towards the project to a new Location.

**Parameters**

**new\_location** (`pathlib.Path`) – The new Location, where the user wants to save new projects.

**abstract set\_name(*new\_name*)**

This method changes the name of the project.

**Parameters**

**new\_name** (`str`) – The new name of the project

**Returns**

true if change was successful, false else

**Return type**

`bool`

**abstract get\_name()**

This method returns the name of the project.

**Returns**

name of the project

**Return type**

`str`

**abstract set\_description(*new\_description*)**

This method changes the description of the project.

**Parameters**

**new\_description** (`str`) – The new description of the project

**Returns**

true if change successful, false else

**Return type**

`bool`

**abstract get\_description()**

This method returns the description of the project.

**Returns**

The description of the project

**Return type**

`str`



**abstract export\_project(*path*)**

Exports the whole project to the given path.

**Parameters**

**path** (*pathlib.Path*) – The path where the project shall be exported to

**Returns**

true, if export was successful, otherwise false.

**Return type**

bool

**abstract export\_configuration(*path*)**

Exports the configuration to the given path. More specific, exports all the categories and their configurations, to the given path.

**Parameters**

**path** (*pathlib.Path*) – The path where the configurations shall be exported to

**Returns**

true, if export was successful, otherwise false.

**Return type**

bool

**abstract export\_calculation(*path*)**

Exports the results of the calculation to the given path. Whereby the calculation are a folder with all the different results from each calculation step in it.

**Parameters**

**path** (*pathlib.Path*) – The path where the results of the calculation shall be exported to

**Returns**

true, if export was successful, otherwise false.

**Return type**

bool

**abstract export\_map(*path*)**

Exports an HTML-Data with the map in it, to the given path.

**Parameters**

**path** (*pathlib.Path*) – The path, where the map shall be exported to

**Returns**

true, if export was successful, otherwise false.

**Return type**

bool

**src.osm\_configurator.model.application.application\_settings module****class ApplicationSettings**

Bases: *object*

This class job is to manage the settings apart from the project settings. In those settings the default-location to save projects can be changed.

**\_\_init\_\_()**

Creates a new instance of the ApplicationSettings.

### **get\_default\_location()**

Gives back the path pointing towards the project.

#### **Returns**

Returns the path of the default location.

#### **Return type**

`pathlib.Path`

### **set\_default\_location(*new\_location*)**

Sets the default path pointing towards the project to a new Location.

#### **Parameters**

**new\_location** (`pathlib.Path`) – The new Location, where the user wants to save new projects.

## **src.osm\_configurator.model.application.passive\_project module**

### **class PassiveProject(*project\_folder\_path*)**

Bases: `object`

This class job is to manage the passive projects. Those are the all projects shown in the Main Menu. Therefore, the class holds the name, description, last edit date and path of the projects.

#### **\_\_init\_\_(*project\_folder\_path*)**

Creates a new instance of the PassiveProject.

#### **Parameters**

**project\_folder\_path** (`Path`) – The path to the project you want to make a PassiveProject on.

### **get\_name()**

Gives back the name of the passive project.

#### **Returns**

The name of the passive project.

#### **Return type**

`str`

### **get\_description()**

Gives back the description of the passive project.

#### **Returns**

The description of the passive project.

#### **Return type**

`str`

### **get\_edit\_date()**

Gives back the last edit date of the passive project.

#### **Returns**

The last edit date of the passive project.

#### **Return type**

`str`

**get\_project\_folder\_path()**

Gives back the path pointing towards the passive project.

**Returns**

The path pointing towards the passive project.

**Return type**

`pathlib.Path`

**src.osm\_configurator.model.application.recommender\_system module****class RecommenderSystem**

Bases: `object`

This class job is to provide recommendations to different classes.

**\_\_init\_\_()**

Creates a new instance of the RecommenderSystem.

**recommend(input)**

Creates recommendations based on user input

**Parameters**

**input** (`str`) – The input from which to generate suggestions.

**Returns**

Returns a list of strings containing the recommendations depending on the input.

**Return type**

`list<str>`

**Module contents****src.osm\_configurator.model.parser package****Submodules****src.osm\_configurator.model.parser.calculation\_parser module****class CalculationParser**

Bases: `CalculationParserInterface`

The CalculationParser job is to parse the calculation files, that are created from the calculation process. It ensures that all data required for a calculation step is there.

Examples: The TagFilterPhase needs the files that got previously calculated in GeoDataPhase.

**\_\_init\_\_()**

Creates a new instance of the calculation\_parser\_interface.CalculationParser.

**check\_validity\_of\_calculation\_step(project\_path, starting\_point)**

Checks whether the passed starting\_point is valid. A starting\_point is valid when all calculation files required for the next step are present in the project.

**Parameters**

- **project\_path** (*pathlib.Path*) – The path pointing towards the project, that needs to be validated.
- **starting\_point** (*calculation\_phase\_enum.CalculationPhase*) – The step we want to calculate next.

**Returns**

True when starting\_point is valid, otherwise false.

**Return type**

bool

**src.osm\_configurator.model.parser.calculation\_parser\_interface module**

**class CalculationParserInterface**

Bases: *ABC*

The CalculationParser job is to parse the calculation files, that are created from the calculation process. It ensures that all data required for a calculation step is there.

Examples: The TagFilterPhase needs the files that got previously calculated in GeoDataPhase.

**abstract check\_validity\_of\_calculation\_step**(*project\_path, starting\_point*)

Checks whether the passed starting\_point is valid. A starting\_point is valid when all calculation files required for the next step are present in the project.

**Parameters**

- **project\_path** (*pathlib.Path*) – The path pointing towards the project, that needs to be validated.
- **starting\_point** (*calculation\_phase\_enum.CalculationPhase*) – The step we want to calculate next.

**Returns**

True when starting\_point is valid, otherwise false.

**Return type**

bool

**src.osm\_configurator.model.parser.category\_parser module**

**class CategoryParser**

Bases: *CategoryParserInterface*

The CategoryParser job, is to parse the category file that are created when creating a project and make an internal representation out of it. In the category file there are the different categories from the project defined, for more information about this look at the documentation of Category.

**\_\_init\_\_()**

Creates a new instance of the CategoryParser.

**parse\_category\_file**(*path*)

Creates an internal representation of the category file it got as an input. What the Category includes, check this: Category.

**Parameters**

**path** (*pathlib.Path*) – The path to the category file.

**Returns**

A List of categories, that describe each category from the category file.

**Return type**

`list[category.Category]`

**src.osm\_configurator.model.parser.category\_parser\_interface module****class CategoryParserInterface**

Bases: `ABC`

The CategoryParser job, is to parse the category file that are created when creating a project and make an internal representation out of it. In the category file there are the different categories from the project defined, for more information about this look at the documentation of `Category`.

**abstract parse\_category\_file(path)**

Creates an internal representation of the category file it got as an input. What the Category includes, check this: `Category`.

**Parameters**

**path** (`pathlib.Path`) – The path to the category file.

**Returns**

A List of categories, that describe each category from the category file.

**Return type**

`list[category.Category]`

**src.osm\_configurator.model.parser.cutOut\_parser module****class CutOutParser**

Bases: `CutOutParserInterface`

This Class parses cut\_out files to an internal representation of the cut\_out\_file.

**parse\_cutout\_file(path)**

This method takes in the path to a cut\_out file and parses to an internal representation of TrafficCells.

A cut\_out file is a `.geojson` file that consists of multiple TrafficCells. Each TrafficCell has a name and a polygon, which is the bounding box of the Traffic Cell.

**Parameters**

**path** (`pathlib.Path`) – The path pointing towards cut\_out file we want to parse.

**Returns**

Our cut\_out file transformed into a list of TrafficCells.

**Return type**

`list[traffic_cell.TrafficCell]`

## Examples

To see an example for a cut\_out file check out the file *data/partOfKarlsruhe.geojson*.

### src.osm\_configurator.model.parser.cutOut\_parser\_interface module

#### class CutOutParserInterface

Bases: [ABC](#)

This Class parses cut\_out files to an internal representation of the cut\_out\_file.

##### abstract parse\_cutout\_file(path)

This method takes in the path to a cut\_out file and parses to an internal representation of TrafficCells.

A cut\_out file is a *.geojson* file that consists of multiple TrafficCells. Each TrafficCell has a name and a polygon, which is the bounding box of the Traffic Cell.

##### Parameters

**path** ([pathlib.Path](#)) – The path pointing towards cut\_out file we want to parse.

##### Returns

Our cut\_out file transformed into a list of TrafficCells.

##### Return type

[list\[traffic\\_cell.TrafficCell\]](#)

## Examples

To see an example for a cut\_out file check out the file *data/partOfKarlsruhe.geojson*.

### src.osm\_configurator.model.parser.osm\_data\_parser module

#### class CategoryParser

Bases: [OSMDataParserInterface](#)

The OSMDataParser job is to parse the OSMData into a human-readable format. This human-readable format is a GeoDataFrame from GeoPandas.

##### \_\_init\_\_()

Creates a new instance of the CategoryParser.

##### parse\_osm\_data\_file(path)

It gets a path pointing towards an OSM data in protocol buffer Binary Format(pbf) and transforms it into an GeoDataFrame. Each row in the GeoDataFrame is a single data entry, which is an osm element from the read osm data. Each column in the GeoDataFrame is a feature of the osm element from the osm\_data, such as the location of the osm element, whereby a feature is a tag or something otherwise that describes the osm-element e.g. location.

##### Parameters

**path** ([pathlib.Path](#)) – The path pointing towards the OSM data we want to parse in the “.pbF” format.

##### Returns

The parsed OSM data as a GeoDataFrame.

##### Return type

GeoDataFrame

**src.osm\_configurator.model.parser.osm\_data\_parser\_interface module****class OSMDataParserInterface**Bases: [ABC](#)

The OSMDataParser job is to parse the OSMData into a human-readable format. This human-readable format is a GeoDataFrame from GeoPandas.

**abstract parse\_osm\_data\_file(path)**

It gets a path pointing towards an OSM data in protocol buffer Binary Format(pbf) and transforms it into an GeoDataFrame. Each row in the GeoDataFrame is a single data entry, which is an osm element from the read osm data. Each column in the GeoDataFrame is a feature of the osm element from the osm\_data, such as the location of the osm element, whereby a feature is a tag or something otherwise that describes the osm-element e.g. location.

**Parameters**

**path** ([pathlib.Path](#)) – The path pointing towards the OSM data we want to parse in the “.pbf” format.

**Returns**

The parsed OSM data as a GeoDataFrame.

**Return type**

GeoDataFrame

**Module contents****src.osm\_configurator.model.project package****Subpackages****src.osm\_configurator.model.project.calculation package****Submodules****src.osm\_configurator.model.project.calculation.aggregation\_method\_enum module****class AggregationMethod(value)**Bases: [Enum](#)

This enum describes all the available aggregation methods that are possible to use. Whereby an aggregation methods is a method, that takes in data and an attractivity attribute. Finally, it outputs and calculates a function on these parameters. The first argument points towards the function, while the second argument is the name of the method.

**SUM** = (<function \_sum>, 'sum')

Calculates the sum of the attractivity attribute over all osm elements from the data.

**AVERAGE** = (<function \_average>, 'average')

Calculates the average of the attractivity attribute over all osm elements from the data.

**MEAN** = (<function \_mean>, 'mean')

Calculates the mean of the attractivity attribute over all osm elements from the data.

**UPPER\_QUARTILE** = (<function `_upper_quartile`>, 'upper quartile')

Calculates the upper\_quartile of the attractivity attribute over all osm elements from the data.

**LOWER\_QUARTILE** = (<function `_lower_quartile`>, 'lower quartile')

Calculates the lower quartile of the attractivity attribute over all osm elements from the data.

**MAXIMUM** = (<function `_maximum`>, 'maximum')

Calculates the maximum of the attractivity attribute over all osm elements from the data.

**MINIMUM** = (<function `_minimum`>, 'minimum')

Calculates the minimum of the attractivity attribute over all osm elements from the data.

**calculate\_aggregation**(*data*, *attractivity\_name*)

Executes the aggregation method of the called enum type.

**Parameters**

- **data** (*geopandas.GeoDataFrame*) – The data on which we want to execute the function on, should be a GeoDataFrame containing osm elements.
- **attractivity\_name** (*str*) – This is the name of the attractivity through which we want to call the function, the *attractivity\_name* should be the name of a column in the data GeoDataFrame.

**Returns**

The aggregated value of the attractivity values from all osm elements.

**Return type**

*float*

**get\_name**()

Getter for the name of the enum type.

**Returns**

Name of the enum type

**Return type**

*str*

## **src.osm\_configurator.model.project.calculation.aggregation\_phase module**

### **class AggregationPhase**

Bases: *ICalculationPhase*

This calculation phase is responsible for aggregating the attractivity attributes in the given traffic cells. For details see the method `calculate()`.

**calculate**(*configuration\_manager*)

Aggregates the attractivity attributes in the given traffic cells. The calculation phase reads the data of the previous calculation phase. Now for every traffic cell all selected aggregation methods are performed for all attractivity attributes. For details on the different aggregation methods, see `AggregationMethod`. After the calculations are done, the results are stored on the hard-drive.

**Parameters**

**configuration\_manager** (*configuration\_manager.ConfigurationManager*) – The object containing all the configuration needed for execution.

**Returns**

The state of the calculation, after this phase finished its execution or failed trying so.



**Return type***calculation\_state\_enum.CalculationState***src.osm\_configurator.model.project.calculation.attractivity\_phase module****class AttractivityPhase**Bases: *ICalculationPhase*

This calculation phase is responsible for calculating the attractivity attributes of the OSM-elements. For details see the method `calculate()`.

**calculate**(*configuration\_manager*)

Calculates the attractivity attributes of the osm-elements The calculation phase reads the data of the previous calculation phase. Now it calculates the attractivity attributes of every OSM-element. The attractivity attributes that are calculated for an osm-element are dependent on the category, the element belongs to. The value of an attractivity attribute is computed as a linear function with the previously computed attributes. The factors of this linear function are given in the configuration of the category. After the calculations are done, the results are stored on the hard-drive.

**Parameters**

**configuration\_manager** (*configuration\_manager.ConfigurationManager*) – The object containing all the configuration needed for execution.

**Returns**

The state of the calculation, after this phase finished its execution or failed trying so.

**Return type***calculation\_state\_enum.CalculationState***src.osm\_configurator.model.project.calculation.building\_on\_edge\_manager module****class BuildingOnEdgeManager**(*file\_paths, border*)Bases: *object*

This class handles the edge-case when buildings are on the edge of specified bounding-box. It is mainly used to remove building that are on this edge.

**\_\_init\_\_**(*file\_paths, border*)

Creates a new instance of the “BuildingOnEdgeManager”.

**Parameters**

- **file\_paths** (*List[pathlib.Path]*) – A list of path each pointing towards an osm-data file through which we want to remove the buildings on the edge.
- **border** (*List[shapely.Polygon]*) – A list of polygon. Each polygon belongs to one entry in the file\_path list and specifies the border of said file.

**remove\_buildings\_on\_edge**()

Reads in the data from the file path it got, and removes all buildings from the data that are on the edge. This means building which are on the edge, half in half out.

**Returns**

True when successful, otherwise false.

**Return type**  
(bool)

### src.osm\_configurator.model.project.calculation.calculation\_manager module

**class** CalculationManager(configuration\_manager)

Bases: `object`

The CalculationManager manages the calculation of the Project. The Calculation are distributed on the calculation phases which are also managed by the CalculationController.

**\_\_init\_\_**(configuration\_manager)

Gets called when we first create an object of this class. It saves all information it needs for managing the calculations.

**Parameters**

**configuration\_manager** (`configuration_manager.ConfigurationManager`) – Saves all information required to configure the calculation.

**cancel\_calculation**()

This method will cancel an ongoing calculation. A calculation consists of an CalculationPhase, that will be interrupted.

**Returns**

True if it is successful and false if something goes wrong, or no calculation is going on.

**Return type**

`bool`

**start\_calculation**(starting\_point)

Starts the calculation. Distributes the calculations to the calculation phases.

**Parameters**

**starting\_point** (`calculation_phase_enum.CalculationPhase`) – The starting phase of the calculation. The calculations start from this phase.

**Returns**

The state of the calculation, after trying to start the calculations.

**Return type**

`calculation_phase_enum.CalculationState`

### src.osm\_configurator.model.project.calculation.calculation\_phase\_enum module

**class** CalculationPhase(value)

Bases: `Enum`

This enum provides a list of phases the calculation can be in. These phases will be worked through in order like a pipeline. An enum consists of two values a name which will be displayed and an order in which the phases will be calculated, also used to know in which order to display the phases. If you want to know more about the calculation phases refer to the “Pflichtenheft”, or the documentation of the individual phases in *project.calculation*

Each enum consists of three variables: (name, folder\_name, order). The name of the phase is used in the GUI, to display the correct name of the phase. The folder\_name is used to save the results of each phase in it. The order is used to differentiate in which order the phases get called.

```
NONE = ('None', 'none', 0)
```

```
GEO_DATA_PHASE = ('Data Input and Geo-Filter', 'geo_data_phase_results', 1)
```

```
TAG_FILTER_PHASE = ('Tag-filter', 'tag_filter_phase', 2)
```

```
REDUCTION_PHASE = ('Reduction', 'reduction_phase_results', 3)
```

```
ATTRACTIVITY_PHASE = ('Attractivity', 'attractivity_phase_results', 4)
```

```
AGGREGATION_PHASE = ('Aggregation', 'aggregation_phase_result', 5)
```

```
get_name()
```

Getter for the name of the enum type.

**Returns**

Name of the Phase.

**Return type**

(str)

```
get_folder_name_for_results()
```

Getter for the folder name of the enum type.

**Returns**

The folder name of the enum.

**Return type**

(str)

```
get_order()
```

Getter for the order of the enum type.

**Returns**

order of the enum.

**Return type**

(int)

### src.osm\_configurator.model.project.calculation.calculation\_phase\_interface module

```
class ICalculationPhase
```

Bases: [ABC](#)

This class represents a calculation phase. A calculation phase is a single step in the big process of computing the final results. Calculation phases are executed after each other. A calculation phase consists of the following 3 steps:

1. Load needed results of previously computed calculation phases.
2. Execute the computations of this calculation phase.
3. Store the results of this computation phase so the following execution phases can read it.

```
abstract calculate(configuration_manager)
```

Performs the calculations of the calculation phase. This consists of the following steps:

1. Load needed results of previously computed calculation phases.
2. Execute the computations of this calculation phase.

3. Store the results of this computation phase so the following execution phases can read it.

**Parameters**

**configuration\_manager** (`configuration_manager.ConfigurationManager`) – The ConfigurationManager where the information about the configuration of the configuration is stored.

**Returns**

The state of the calculation after this phase finished its execution or failed trying so.

**Return type**

*calculation\_state\_enum.CalculationState*

**src.osm\_configurator.model.project.calculation.calculation\_state\_enum module**

**class CalculationState**(*value*)

Bases: `Enum`

This enum provides a list of states the calculations can be in. The states can be positive, indicating the calculation is working correctly. But they can be negative as well, indicating an error in the calculation. Every state is defined by a unique description of the state.

**NOT\_STARTED\_YET** = ('Not started yet', 'The calculation was not started yet.')

**RUNNING** = ('Running', 'The calculations are currently running.')

**ENDED\_SUCCESSFULLY** = ('Done', 'The calculations ended successfully.')

**ERROR\_INVALID\_OSM\_DATA** = ('Invalid OSM Data', 'Error: The osm data are not valid.')

**ERROR\_INVALID\_CUT\_OUT\_DATA** = ('Invalid Cut Out Data', 'Error: The cut out data are not valid.')

**ERROR\_INVALID\_CATEGORIES** = ('Invalid Categories', 'Error: The category configuration is not valid.')

**ERROR\_INVALID\_PREVIOUS\_CALCULATIONS** = ('Invalid calculation phase', "Error: This calculation phase can not be calculated, because a previous calculation has invalid results or wasn't run.")

**get\_name()**

Gives back the name of a calculation state.

**Returns**

The name of the calculation state.

**Return type**

`str`

**get\_description()**

Gives back the description of a calculation state.

**Returns**

The description, that describes the state in natural language.

**Return type**

`str`

**src.osm\_configurator.model.project.calculation.geo\_data\_phase module****class GeoDataPhase**Bases: *ICalculationPhase*

This Phase is responsible for three things: 1. Converting the osm\_data file into the right format. 2. Splitting the osm\_data file into smaller pieces. 3. If selected, removing building which are on the border of the traffic cell. For details see the method calculate().

**calculate**(*configuration\_manager*)

This method does three things: 1. It splits the big input osm\_data file into multiple smaller one. There are three main reason to do that - Organisational: Each file contains the osm elements of one previously defined traffic cell. This is more organized. - Parallelization: Splitting the file into multiple smaller files allows, for better parallelization, since every thread/process can work with one file. - RAM usage: RAM capacity is limited. We can't load one big file into the memory at once, so we need to split up the file. 2. After that it converts the osm data files into files with the ".pbf" osm data file format, which is done since the library we use internally uses ".pbf" formats. 3. If the option "buildings on edge are in" didn't get selected. It removes all buildings which are on the edge/border.

**Parameters**

**configuration\_manager** (*configuration\_manager.ConfigurationManager*) – The object containing all the configuration needed for an execution.

**Returns**

The state of the calculation after this phase finished its execution or failed trying so.

**Return type**

*calculation\_state\_enum.CalculationState*

**src.osm\_configurator.model.project.calculation.osm\_file\_converter module****class OSMFileConverter**(*file\_path*)Bases: *object*

This class handles osm file conversion, it is used to transform an osm data file from one format to the others. For more on the different file format check "calculation.OSMFileFormat" out.

**\_\_init\_\_**(*file\_path*)

Creates a new instance of "OSMFileConverter".

**Parameters**

**file\_path** (*pathlib.Path*) – The path pointing towards the file which format we want to transform into another format.

**convert\_file**(*data\_format*)

Transforms an osm file format into another osm file format. Allowed format: ".pbf", ".osm.bz2", ".osm".

**Parameters**

**data\_format** (*osm\_file\_format\_enum.OSMFileFormat*) – In which osm file format we want to transform our file into.

**Returns**

True if successful, otherwise false.

**Return type**

(bool)

## src.osm\_configurator.model.project.calculation.osm\_file\_format\_enum module

**class** `OSMFileFormat`(*value*)

Bases: `Enum`

This enum describes the different osm file formats we use for file\_conversion. For more information on the osm file formats check this out: [https://wiki.openstreetmap.org/wiki/OSM\\_file\\_formats](https://wiki.openstreetmap.org/wiki/OSM_file_formats).

**PBF** = `'.pbf'`

**BZ2** = `'.osm.bz2'`

**OSM** = `'.osm'`

**get\_file\_extension()**

Getter for the file extension of an enum type.

**Returns**

The File extension the osm file format uses.

**Return type**

(str)

## src.osm\_configurator.model.project.calculation.reduction\_phase module

**class** `ReductionPhase`

Bases: `ICalculationPhase`

This calculation phase is responsible for reducing bigger OSM-elements on single coordinates and for generating the values of the attributes for alle OSM-elements. For details see the method calculate().

**calculate**(*configuration\_manager*)

Reduces OSM-elements on single points and calculates their attributes. The calculation phase reads the data of the previous calculation phase. OSM-elements that are not just a single node, must be reduced on one coordinate. For that the centre of the given shape is calculated and set as the new coordinate. This calculation phase does also calculate the attributes of every OSM-element. There is no generic form for calculation attributes, every attribute has an individual calculation. If a method of calculation is not possible or if the user turned it off, the value of the attributes is defined by the default value list of the category. The value is given by the highest priority entry of the default value list, that matches the osm-element. After the calculations are done, the results are stored on the hard-drive.

**Parameters**

**configuration\_manager** (`configuration_manager.ConfigurationManager`) – The object containing all the configuration required for an execution.

**Returns**

The state of the calculation after this phase finished its execution or failed trying so.

**Return type**

`calculation_state_enum.CalculationState`

**src.osm\_configurator.model.project.calculation.split\_up\_files module****class SplitUpFile**(*file\_path*)Bases: *object*

This class is responsible to split up osm-data files, into multiple smaller osm-data files. This is useful since an osm-data file loaded into the ram can be bigger than the capacity of the RAM.

**\_\_init\_\_**(*file\_path*)

Creates a new instance of “SplitUpFile”.

**Parameters****file\_path** (*pathlib.Path*) – The path pointing towards the osm\_data file we want to split.**split\_up\_files**(*coordinates*)

This method splits up the file into multiple smaller ones based on the coordinates it receives.

**Parameters****coordinates** (*List[shapely.Polygon]*) – A list of polygon. Each polygon ist the bounding box of one traffic cell we want to split up.**Returns**

True if successful, otherwise false.

**Return type**

(bool)

**src.osm\_configurator.model.project.calculation.tag\_filter\_phase module****class TagFilterPhase**Bases: *ICalculationPhase*

This calculation phase is responsible for sorting OSM-elements into their corresponding categories. For details see the method calculate().

**calculate**(*configuration\_manager*)

Sorts OSM-elements into their corresponding categories. Firstly this method reads in the OSM-files of the previously executed calculation phase. Every category has defined a tag filter in the configuration phase. The OSM-Elements are now sorted into the categories, depending on whether they do pass or do not pass the corresponding tag filters. A tag filter is defined by a black- and a whitelist. Each list is a collection of constraints of the tags of the osm-elements. An osm-element passes a tag filter, if all constraints of the whitelist are satisfied and no entry of the blacklist is satisfied. After execution the results shall be stored again on the hard-drive.

**Parameters****configuration\_manager** (*configuration\_manager.ConfigurationManager*) – The object containing all the configuration needed for an execution.**Returns**

The state of the calculation after this phase finished its execution or failed trying so.

**Return type***calculation\_state\_enum.CalculationState*

## src.osm\_configurator.model.project.calculation.traffic\_cell module

### class TrafficCell

Bases: `object`

A TrafficCell is an object which describes an area on the earth, it has a name and a bounding box which entail the described area from the name.

### Examples

The “Karlsruher Schloss” is a trafficCell with the name=“Karlsruher Schloss” and its bounding box being a polygon of coordinates(latitude, longitude) which entail the area of the “Karlsruher Schloss”.

#### `__init__()`

Creates a new instance of the TrafficCell.

#### `get_name()`

Getter for the name of the TrafficCell.

##### Returns

The name of the TrafficCell.

##### Return type

`str`

#### `get_bounding_box()`

Getter for the bounding box of the TrafficCell.

##### Returns

A list of coordinates that describe the bounding box of the TrafficCell.

##### Return type

`shapely.Polygon`

## Module contents

## src.osm\_configurator.model.project.configuration package

### Submodules

## src.osm\_configurator.model.project.configuration.aggregation\_configuration module

### class AggregationConfiguration

Bases: `object`

This class manages the different aggregation methods stored in an enum. Therefore, it activates and deactivates those methods. Activating means that this aggregation methods should be used during the aggregation phase in the calculation. Deactivating means the opposite.

#### `__init__()`

Creates a new instance of the AggregationConfiguration.



### **get\_all\_aggregation\_methods()**

Gives back a List of all possible aggregation methods.

#### **Returns**

A list containing all aggregation methods.

#### **Return type**

`list[aggregation_method_enum.AggregationMethod]`

### **is\_aggregation\_method\_active(method)**

Checks, if a given aggregation method is active.

#### **Parameters**

**method** (`aggregation_method_enum.AggregationMethod`) – The method, which is to be checked.

#### **Returns**

True if the aggregation method is active, otherwise false.

#### **Return type**

`bool`

### **set\_aggregation\_method\_active(method, active)**

Changes the aggregation method from active to inactive and vice versa. If an already active aggregation method should be activated, it stays active. The same applies to inactive aggregation methods, which should be deactivated.

#### **Parameters**

- **method** (`aggregation_method_enum.AggregationMethod`) – The method, which state should be changed.
- **active** (`bool`) – This is the new state of the aggregation method.

#### **Returns**

True if changing the state works, otherwise false.

#### **Return type**

`bool`

## **src.osm\_configurator.model.project.configuration.attractivity\_attribute module**

### **class AttractivityAttribute(attractivity\_attribute\_name, attractivity\_attribute\_list, base\_attractivity)**

Bases: `object`

AttractivityAttribute models a single Attractivity Attributes to its factors. Each AttractivityAttribute consists of the following elements: - A name, which describes the AttractivityAttribute - A List of attributes factor pairs, which describe the attractivity attribute - A base factor

#### **\_\_init\_\_(attractivity\_attribute\_name, attractivity\_attribute\_list, base\_attractivity)**

Creates a new instance of a “AttractivityAttribute” class.

#### **Parameters**

- **attractivity\_attribute\_name** (`str`) – The name of the Attractivity Attributes
- **attractivity\_attribute\_list** (`List[(attribute_enum.Attribute, float)]`) – A list of attributes each having its own factor.
- **base\_attractivity** (`float`) – The base attractivity value.

## Examples

An example for `attractivity_attribute_list`: [(AREA, 1.0), (NUMER\_OF\_FLOOR, 2.0), (GROUND\_AREA, 6.9)]

### `get_attractivity_attribute_name()`

Getter for attractivity attribute name.

#### Returns

The attractivity attribute name.

#### Return type

`str`

### `set_attractivity_attribute_name(name)`

Setter for the attractivity attribute name.

#### Parameters

**name** (`str`) – name of the attractivity attribute.

#### Returns

true, if the name was successfully set, false otherwise

#### Return type

`bool`

### `get_attractivity_attribute_list()`

Getter for the list of attributes and factors.

#### Returns

The list of attribute factor pairs.

#### Return type

`list[(attribute_enum.Attribute, float)]`

### `set_attractivity_attribute_list(attractivity_attribute_list)`

Setter for the list of attributes and factors.

#### Parameters

**attractivity\_attribute\_list** (`list[(attribute_enum.Attribute, float)]`) – A list of attribute factor pairs we want to set as the new list of attributes and factors.

### `get_base_factor()`

Getter for the base factor.

#### Returns

the base factor

#### Return type

`float`

### `set_base_factor(new_base_factor)`

Setter for the base factor.

#### Parameters

**new\_base\_factor** (`float`) – New value for the base factor

#### Returns

true if the base factor was successfully set, false else

**Return type**`bool`**src.osm\_configurator.model.project.configuration.attribute\_enum module****class Attribute**(*value*)Bases: `Enum`

This enum provides a list of Attributes, the DefaultValueEntry and AttractivityAttributes can use. If you are interested how exactly these Attributes get used checkout AttractivityPhase.

**PROPERTY\_AREA** = 'Property Area'

The area of the property of the osm-element

**NUMER\_OF\_FLOOR** = 'Number of Floors'

the number of floors the osm element has

**FIRST\_FLOOR\_AREA** = 'Floor Area'

the area that the first floor has

**get\_name()**

Getter for the name of the enum type.

**Returns**

Name of the Phase.

**Return type**`(str)`**src.osm\_configurator.model.project.configuration.calculation\_method\_of\_area\_enum module****class CalculationMethodOfArea**(*value*)Bases: `Enum`

Enum Provides Calculation Method of the Area.

**CALCULATE\_SITE\_AREA** = 'Calculate Site Area'**CALCULATE\_BUILDING\_AREA** = 'Calculate Building Area'**abstract get\_calculation\_method()**

Getter for the name of the Calculation Method.

**Returns**

The name of the Calculation Method.

**Return type**`str`

**src.osm\_configurator.model.project.configuration.category module****class Category**Bases: `object`

Represents a category. A category is a collection of configurations for the calculation process. A category defines which OSM-elements are contained by it with a white- and a blacklist. All configurations of the category do only affect does OSM-elements.

**\_\_init\_\_()**

Creates a new instance of a “Category” class.

**is\_active()**

Checks if value “active” is set.

**Returns**

True if active, false if inactive.

**Return type**`bool`**activate()**

Sets the active-value to True.

**Returns**

True, if value was set correctly, False if value was already True.

**Return type**`bool`**deactivate()**

Sets the active-value to False.

**Returns**

True, if value was set correctly, False if value was already False.

**Return type**`bool`**get\_whitelist()**

Getter for the whitelist of the category.

**Returns**

List containing all whitelist values of the class.

**Return type**`list[str]`**set\_whitelist(*new\_whitelist*)**

Changes the old whitelist to a new one.

**Parameters**

**new\_whitelist** (`list[str]`) – value for the new whitelist.

**Returns**

True, if the whitelist was overwritten successfully, else False.

**Return type**`bool`

**get\_blacklist()**

Getter for the blacklist of the category.

**Returns**

list[str]: list containing all blacklist attributes of the class.

**set\_blacklist(new\_blacklist)**

Overwrites the old Blacklist with a new value.

**Parameters**

**new\_blacklist** (*list[str]*) – new value for the blacklist.

**Returns**

True, if the blacklist was overwritten successfully, else False.

**Return type**

bool

**get\_category\_name()**

Getter for the category name.

**Returns**

name of the category.

**Return type**

str

**set\_category\_name(new\_category\_name)**

Overwrites the old category\_name.

**Parameters**

**new\_category\_name** (*str*) – new value for the category\_name.

**Returns**

True, if the overwriting process concluded successfully, else False.

**Return type**

bool

**get\_calculate\_area()**

This says if the area of the category should be calculated or not.

**Returns**

true if it should get calculated, false if not.

**Return type**

bool

**get\_calculation\_method\_of\_area()**

Getter for the calculated area method.

**Returns**

The method with which we calculate the area.

**Return type**

*calculation\_method\_of\_area\_enum.CalculationMethodOfArea*

**set\_calculation\_method\_of\_area(new\_calculate\_area)**

Overwrites current calculate\_area with the given value.

**Parameters**

**new\_calculate\_area** (*bool*) – new value that will overwrite the existing value.

### **get\_calculate\_floor\_area()**

This says if the floor area should be calculated or not.

#### **Returns**

true if the floor are should be calculated, otherwise false.

#### **Return type**

bool

### **set\_calculate\_floor\_area(*new\_calculate\_floor\_area*)**

Overwrites the existing instance of calculate\_floor\_area.

#### **Parameters**

**new\_calculate\_floor\_area** (bool) – new value for calculate\_floor\_are.

#### **Returns**

True, if the overwriting process was successful, else false.

#### **Return type**

bool

### **get\_strictly\_use\_default\_values()**

This says if in the calculation we should strictly use the default values.

#### **Returns**

value of strictly\_use\_default\_values.

#### **Return type**

bool

### **set\_strictly\_use\_default\_values(*new\_strictly\_use\_default\_values*)**

Overwrites the already existing value of strictly\_use\_default\_values.

#### **Parameters**

**new\_strictly\_use\_default\_values** (bool) – new value for strictly\_use\_default\_values.

#### **Returns**

True if the overwriting process was successful, else False.

### **get\_attractivity\_attributes()**

Getter for the AttractivityAttributes of the category.

#### **Returns**

List of all used attractivity attributes

#### **Return type**

list[attractivity\_attribute.AttractivityAttribute]

### **add\_attractivity\_attribute(*new\_attractivity\_attribute*)**

Adds a new attractivity attribute to the list.

#### **Parameters**

**new\_attractivity\_attribute** (attractivity\_attribute.  
AttractivityAttribute) – new attractivityAttribute that will be added.

#### **Returns**

True, if the attribute was added successfully, else False.

#### **Return type**

bool

**remove\_attractivity\_attribute**(*attractivity\_attribute*)

Removes an already existing attribute from the list.

**Parameters**

**attractivity\_attribute** (*attractivity\_attribute.AttractivityAttribute*) – attractivity attribute that will be removed from the list.

**Returns**

True, if the element was removed, else False.

**get\_default\_value\_list**()

Getter for the default values of the category.

**Returns**

List of all used default values.

**Return type**

*list*[*DefaultValueEntry*]

**add\_default\_value\_entry**(*new\_default\_value\_entry*)

Adds a new value to the default\_value\_entry list.

**Parameters**

**new\_default\_value\_entry** (*DefaultValueEntry*) – element to add.

**Returns**

True, if element was added successfully, else False

**Return type**

*bool*

**remove\_default\_value\_entry**(*default\_value\_entry*)

Removes an already existing element from the default\_value\_entry list.

**Parameters**

**default\_value\_entry** (*default\_value\_entry.DefaultValueEntry*) – value that will be removed.

**Returns**

True, if the element was removed successfully, else False.

**Return type**

*bool*

**move\_default\_value\_entry\_up**(*default\_value\_entry*)

Moves an already existing default value from the list one element up.

**Parameters**

**default\_value\_entry** (*default\_value\_entry.DefaultValueEntry*) – element from the list, that will be incremented by one.

**Returns**

True, if the change was successful, else False.

**Return type**

*bool*

**move\_default\_value\_entry\_down**(*default\_value\_entry*)

Moves an already existing default value from list one element down.

**Parameters**

**default\_value\_entry** (`default_value_entry.DefaultValueEntry`) – element from the list, that will be decremented by one.

**Returns**

True, if the change was successful, else false.

**Return type**

`bool`

**src.osm\_configurator.model.project.configuration.category\_manager module**

**class** `CategoryManager`(*categories*)

Bases: `object`

Category Manager holds a list of categories and changes them according to the given needs.

**\_\_init\_\_**(*categories*)

Constructor of the class.

**Parameters**

**categories** (`Category`) – Starting list of categories.

**get\_category**(*index*)

Gets a category based on the index.

**Parameters**

**index** (`int`) – Index in the categories-list, that will be returned.

**Returns**

The Category we wanted.

**Return type**

`category.Category`

**get\_categories**()

Getter for all the Categories.

**Returns**

List of the chosen categories.

**Return type**

`list[Category]`

**create\_category**(*new\_category*)

Creates a new category, that will be empty.

**Returns**

The newly created category.

**Return type**

`category.Category`

**remove\_category**(*category*)

Removes the given category from the categories list, if element is inside the List.

**Parameters**

**category** (`Category`) – Category that will be removed.

**Returns**

True, if the element was removed correctly, else false.



**Return type**

`bool`

**override\_categories**(*new\_category\_list*)

Overwrites the list of categories with the given list, if both lists are not identical.

**Parameters**

**new\_category\_list** (*list*[*Categories*]) – List of categories, that will overwrite the already existing list.

**Returns**

True, if the replacement was successful, else False.

**Return type**

`bool`

**merge\_categories**(*category\_input\_list*)

Merges the existing category list with the given list if both lists are not identical.

**Parameters**

**category\_input\_list** (*list*[*Category*]) – New list of categories that will be merged into the existing list.

**Returns**

True, if the merging was successful, else False.

**Return type**

`bool`

**src.osm\_configurator.model.project.configuration.configuration\_manager module**

**class ConfigurationManager**(*active\_project\_path*)

Bases: `object`

This class job is to manage the configurations of the OSM data, aggregation, cut-out and categories. It also makes this information available to the calculation

**\_\_init\_\_**(*active\_project\_path*)

Creates a new instance of the ConfigurationManager.

**Parameters**

**active\_project\_path** (*pathLib.Path*) – The path pointing towards the project folder.

**get\_osm\_data**()

Gives back the path pointing towards the OSM data file.

**Returns**

The path pointing towards the OSM data.

**Return type**

`pathlib.Path`

**set\_osm\_data**(*osm\_data*)

Edits the path pointing towards the OSM data file.

**Parameters**

**osm\_data** (*pathlib.Path*) – The new path towards the osm data file.

**Returns**

True if changing the path works, otherwise false.

**Return type**

`bool`

**get\_all\_aggregation\_methods()**

Gives back a List of all possible aggregation methods.

**Returns**

A list containing all aggregation methods.

**Return type**

`list[aggregation_method_enum.AggregationMethod]`

**is\_aggregation\_method\_active(method)**

Checks, if a given aggregation method is active.

**Parameters**

**method** (`aggregation_method_enum.AggregationMethod`) – The method, which is to be checked.

**Returns**

True if the aggregation method is active, otherwise false.

**Return type**

`bool`

**set\_aggregation\_method\_active(method, active)**

Changes the aggregation method from active to inactive and vice versa. If an already active aggregation method should be activated, it stays active. The same applies to inactive aggregation methods, which should be deactivated.

**Parameters**

- **method** (`aggregation_method_enum.AggregationMethod`) – The method, which state should be changed.
- **active** (`bool`) – This is the new state of the aggregation method.

**Returns**

True if changing the state works, otherwise false.

**Return type**

`bool`

**get\_cut\_out\_mode()**

Gives back the used cut-out mode.

**Returns**

The used cut-out mode.

**Return type**

`cut_out_mode_enum.CutOutMode`

**set\_cut\_out\_mode(new\_cut\_out\_mode)**

Changes the cut-out mode used during the reduction phase in the calculation.

**Parameters**

**new\_cut\_out\_mode** (`cut_out_mode_enum.CutOutMode`) – The new cut-out mode for the calculation.

**Returns**

True if changing the cut-out mode works, otherwise false.

**Return type**`bool`**get\_cut\_out\_path()**

Gives back the path pointing towards the cut-out file.

**Returns**

The path pointing towards the cut-out.

**Return type**`pathlib.Path`**set\_cut\_out\_path(path)**

Changes the path pointing towards the cut-out file.

**Parameters**

**path** (`pathlib.Path`) – The new path.

**Returns**

True if changing the cut-out path works, otherwise false.

**Return type**`bool`**get\_category(index)**

Gets a category based on the index.

**Parameters**

**index** (`int`) – Index in the categories-list, that will be returned.

**Returns**

The Category we wanted.

**Return type**`category.Category`**get\_categories()**

Getter for all the Categories.

**Returns**

List of the chosen categories.

**Return type**`list[Category]`**create\_category()**

Creates a new category, that will be empty.

**Returns**

The newly created category.

**Return type**`category.Category`**remove\_category(category)**

Removes the given category from the categories list, if element is inside the List.

**Parameters**

**category** (`Category`) – Category that will be removed.

**Returns**

True, if the element was removed correctly, else false.

**Return type**

`bool`

**override\_categories**(*new\_category\_list*)

Overwrites the list of categories with the given list, if both lists are not identical.

**Parameters**

**new\_category\_list** (`list[Categories]`) – List of categories, that will overwrite the already existing list.

**Returns**

True, if the replacement was successful, else False.

**Return type**

`bool`

**merge\_categories**(*category\_input\_list*)

Merges the existing category list with the given list if both lists are not identical.

**Parameters**

**category\_input\_list** (`list[Category]`) – New list of categories that will be merged into the existing list.

**Returns**

True, if the merging was successful, else False.

**Return type**

`bool`

**get\_active\_project**()

This method gives back the active project.

**Returns**

The path pointing towards the current project folder.

**Return type**

`pathlib.Path`

**get\_is\_data\_downloaded**()

Gives back if data in DownloadData is downloaded.

**Returns**

True if data is downloaded, otherwise false.

**Return type**

`bool`

**src.osm\_configurator.model.project.configuration.cut\_out\_configuration module**

**class CutOutConfiguration**

Bases: `object`

This class job is to store the cut-out mode and the cut-out file path. Both are required during the reduction-phase in the calculation.

**\_\_init\_\_**()

Creates a new instance of the “CutOutConfiguration” class.

**get\_cut\_out\_mode()**

Gives back the used cut-out mode.

**Returns**

The used cut-out mode.

**Return type**

*cut\_out\_mode\_enum.CutOutMode*

**set\_cut\_out\_mode(new\_cut\_out\_mode)**

Changes the cut-out mode used during the reduction phase in the calculation.

**Parameters**

**new\_cut\_out\_mode** (*cut\_out\_mode\_enum.CutOutMode*) – The new cut-out mode for the calculation.

**Returns**

True if changing the cut-out mode works, otherwise false.

**Return type**

*bool*

**get\_cut\_out\_path()**

Gives back the path pointing towards the cut-out file.

**Returns**

The path pointing towards the cut-out.

**Return type**

*pathlib.Path*

**set\_cut\_out\_path(path)**

Changes the path pointing towards the cut-out file.

**Parameters**

**path** (*pathlib.Path*) – The new path.

**Returns**

True if changing the cut-out path works, otherwise false.

**Return type**

*bool*

**src.osm\_configurator.model.project.configuration.cut\_out\_mode\_enum module****class CutOutMode(value)**

Bases: *Enum*

The job of this enum is to store the different cut-out-modes used in the reduction during the calculation. We differentiate on, if we should include building which are on the edge/border, this mean partially inside the traffic cell, or not.

**BUILDINGS\_ON\_EDGE\_ACCEPTED** = 'Buildings on edge are accepted'

**BUILDINGS\_ON\_EDGE\_NOT\_ACCEPTED** = 'Building on the edge are not accepted'

**get\_name()**

Getter for the name of the enum type.

**Returns**  
the name of the enum

**Return type**  
`str`

## `src.osm_configurator.model.project.configuration.default_value_entry` module

**class** `DefaultValueEntry`(*tag, attribute\_default\_values*)

Bases: `object`

`DefaultValueEntry` stores a default value for every attribute. Default values can be set and read.

**`__init__`**(*tag, attribute\_default\_values*)

Constructor of the class, creates an empty `DefaultValueEntry` with 0 for all the factor values.

**`get_default_value_entry_tag`**()

Returns the tag associated with this default value entry :returns: The tag of this entry :rtype: `str`

**`set_tag`**(*new\_tag*)

Sets a new value for a given tag

**Parameters**

**`new_tag`** (`str`) – value for overwriting the current tag, must be a valid OSM-tag

**Returns**

true if the overwriting process was successful, else false

**Return type**  
`bool`

**`set_attribute_default`**(*attribute, value*)

Sets the default value of an attribute

**Parameters**

- **`attribute`** (`attribute_enum.Attribute`) – Attribute whose value will be overwritten
- **`value`** (`float`) – new default value

**Returns**

true, if overwriting process was successful, else false

**Return type**  
`bool`

**`get_attribute_default`**(*attribute*)

Gets the default value of a certain attribute

**Parameters**

**`attribute`** (`attribute_enum.Attribute`) – Attribute whose value is searched for

**Returns**

The default value of the attribute

**Return type**  
`float`

**src.osm\_configurator.model.project.configuration.download\_data module****class DownloadData**Bases: `object`

This class manages the download of OSM data depending on a list of coordinates.

**\_\_init\_\_()**

Creates a new instance of the DownloadData.

**download\_data(coordinates)**

Downloads the OSM data which the coordinates dictate.

**Parameters****coordinates** (`shapely.Polygon`) – The new area, which should be downloaded**Returns**

True when the download works, otherwise false.

**Return type**`bool`**src.osm\_configurator.model.project.configuration.osm\_data\_configuration module****class OSMDDataConfiguration**Bases: `object`

The job of the OSMDDataConfiguration is to store the path pointing towards the OSM data file.

**\_\_init\_\_()**

Creates a new instance of the “OSMDDataConfiguration” class.

**get\_osm\_data()**

Gives back the path pointing towards the OSM data file.

**Returns**

The path pointing towards the OSM data.

**Return type**`pathlib.Path`**set\_osm\_data(osm\_data)**

Edits the path pointing towards the OSM data file.

**Parameters****osm\_data** (`pathlib.Path`) – The new path towards the osm data file.**Returns**

True if changing the path works, otherwise false.

**Return type**`bool`**download\_data(coordinates)**

Downloads the OSM data which the coordinates dictate.

**Parameters****coordinates** (`shapely.Polygon`) – The new area, which should be downloaded

**Returns**

True when the download works, otherwise false.

**Return type**

bool

**Module contents**

**Submodules**

**src.osm\_configurator.model.project.active\_project module**

**class** `ActiveProject`(*project\_folder*, *is\_newly\_created*)

Bases: `object`

This class job is to manage the active project the user is working on. Whereby an active project, a project is that got selected by the user in the project selected screen or created.

**\_\_init\_\_**(*project\_folder*, *is\_newly\_created*)

Creates a new instance of the ActiveProject. In this process it creates the ConfigurationManager and also differentiate between the case that the project is new or loaded. In the case of an existing project it calls the ProjectLoader, otherwise it creates a new project.

**Parameters**

- **project\_folder** (*pathlib.Path*) – This is path pointing towards the folder, where the project is saved.
- **is\_newly\_created** (*bool*) – This argument is true if the project is newly created, otherwise false.

**create**(*name*, *description*)

This method creates a new project and adds a name and a description to it.

**Parameters**

- **name** (*str*) – The name of the project.
- **description** (*str*) – A description of the project.

**Returns**

True if creating the project works, otherwise false.

**Return type**

bool

**get\_last\_step**()

This method is there so that the user can continue working in the same phase in an existing project where he previously stopped.

**Returns**

The last phase the user was working on.

**Return type**

*config\_phase\_enum.ConfigPhase*

**start\_calculation**(*calculation\_phase*)

This method is to start the calculation (after the configuration is finished).



#### Parameters

**calculation\_phase** (`calculation_phase_enum.CalculationPhase`) – The calculation phase, where the calculation shall start.

#### Returns

The calculation state where the calculation started. Can be an error state, so signify an error, that prevented the start of the calculation.

#### Return type

`calculation_state_enum.CalculationState`

#### **get\_project\_path()**

This method is to give back the path pointing towards the project folder.

#### Returns

The path pointing towards the project folder.

#### Return type

`pathlib.Path`

#### **get\_osm\_data()**

Gives back the path pointing towards the OSM data file.

#### Returns

The path pointing towards the OSM data.

#### Return type

`pathlib.Path`

#### **set\_osm\_data(osm\_data)**

Edits the path pointing towards the OSM data file.

#### Parameters

**osm\_data** (`pathlib.Path`) – The new path towards the osm data file.

#### Returns

True if changing the path works, otherwise false.

#### Return type

`bool`

#### **get\_all\_aggregation\_methods()**

Gives back a List of all possible aggregation methods.

#### Returns

A list containing all aggregation methods.

#### Return type

`list[aggregation_method_enum.AggregationMethod]`

#### **is\_aggregation\_method\_active(method)**

Checks, if a given aggregation method is active.

#### Parameters

**method** (`aggregation_method_enum.AggregationMethod`) – The method, which is to be checked.

#### Returns

True if the aggregation method is active, otherwise false.

#### Return type

`bool`

**set\_aggregation\_method\_active**(*method*, *active*)

Changes the aggregation method from active to inactive and vice versa. If an already active aggregation method should be activated, it stays active. The same applies to inactive aggregation methods, which should be deactivated.

**Parameters**

- **method** (`aggregation_method_enum.AggregationMethod`) – The method, which state should be changed.
- **active** (`bool`) – This is the new state of the aggregation method.

**Returns**

True if changing the state works, otherwise false.

**Return type**

`bool`

**get\_cut\_out\_mode**()

Gives back the used cut-out mode.

**Returns**

The used cut-out mode.

**Return type**

`cut_out_mode_enum.CutOutMode`

**set\_cut\_out\_mode**(*new\_cut\_out\_mode*)

Changes the cut-out mode used during the reduction phase in the calculation.

**Parameters**

**new\_cut\_out\_mode** (`cut_out_mode_enum.CutOutMode`) – The new cut-out mode for the calculation.

**Returns**

True if changing the cut-out mode works, otherwise false.

**Return type**

`bool`

**get\_cut\_out\_path**()

Gives back the path pointing towards the cut-out file.

**Returns**

The path pointing towards the cut-out.

**Return type**

`pathlib.Path`

**set\_cut\_out\_path**(*path*)

Changes the path pointing towards the cut-out file.

**Parameters**

**path** (`pathlib.Path`) – The new path, towards a cut-out file.

**Returns**

True if changing the cut-out path works, otherwise false.

**Return type**

`bool`

**get\_category(*index*)**

Gets a category based on the index.

**Parameters**

**index** (*int*) – Index in the categories-list, that will be returned.

**Returns**

The Category we wanted, NONE if the index is out of bounds of the list.

**Return type**

*category.Category*

**get\_categories()**

Getter for all the Categories.

**Returns**

List of the chosen categories.

**Return type**

*list[category.Category]*

**create\_category()**

Creates a new category, that will be empty.

**Returns**

The newly created category.

**Return type**

*category.Category*

**remove\_category(*category*)**

Removes the given category from the categories list, if element is inside the List.

**Parameters**

**category** (*category.Category*) – Category that will be removed.

**Returns**

True, if the element was removed correctly, else false.

**Return type**

*bool*

**override\_categories(*new\_category\_list*)**

Overwrites the list of categories with the given list, if both lists are not identical.

**Parameters**

**new\_category\_list** (*list[category.Category]*) – List of categories, that will overwrite the already existing list.

**Returns**

True, if the replacement was successful, else false.

**Return type**

*bool*

**merge\_categories(*category\_input\_list*)**

Merges the existing category list with the given list if both lists are not identical. If two categories conflict in their name, the newer category will be used.

**Parameters**

**category\_input\_list** (*list[category.Category]*) – New list of categories that will be merged into the existing list.

**Returns**

True, if the merging was successful, else False.

**Return type**

`bool`

**create\_map(*cut\_out*)**

This method to create a map from to given cut-out.

**Parameters**

**cut\_out** (`cut_out_configuration.CutOutConfiguration`) – The cut-out configuration from which the map should be created.

**Returns**

True if creating the map works, otherwise false.

**Return type**

`bool`

**create\_boxplot(*data*)**

This method is to visualize the data by creating a boxplot. It is used to visualize the calculated end result via a boxplot.

**Parameters**

**data** (`matplotlib.axes.Axes`) – A plot of the data which we want to visualize.

**Returns**

True if creating the boxplot works, otherwise false.

**Return type**

`bool`

**get\_location()**

Getter for the location of the active project on the disk.

**Returns**

The location of the active project

**Return type**

`pathlib.Path`

**set\_name(*new\_name*)**

This method changes the name of the project.

**Parameters**

**new\_name** (`str`) – The new name of the project

**Returns**

true if change was successful, false else

**Return type**

`bool`

**get\_name()**

This method returns the name of the project.

**Returns**

name of the project

**Return type**

`str`

**set\_description**(*new\_description*)

This method changes the description of the project.

**Parameters**

**new\_description** (*str*) – The new description of the project

**Returns**

true if change successful, false else

**Return type**

*bool*

**get\_description**()

This method returns the description of the project.

**Returns**

The description of the project

**Return type**

*str*

**export\_project**(*path*)

Exports the whole project to the given path.

**Parameters**

**path** (*pathlib.Path*) – The path where the project shall be exported to

**Returns**

true, if export was successful, otherwise false.

**Return type**

*bool*

**export\_configuration**(*path*)

Exports the configuration to the given path. More specific, exports all the categories and their configurations, to the given path.

**Parameters**

**path** (*pathlib.Path*) – The path where the configurations shall be exported to

**Returns**

true, if export was successful, otherwise false.

**Return type**

*bool*

**export\_calculation**(*path*)

Exports the results of the calculation to the given path. Whereby the calculation are a folder with all the different results from each calculation step in it.

**Parameters**

**path** (*pathlib.Path*) – The path where the results of the calculation shall be exported to

**Returns**

true, if export was successful, otherwise false.

**Return type**

*bool*

**export\_map**(*path*)

Exports an HTML-Data with the map in it, to the given path.

**Parameters**

**path** (*pathlib.Path*) – The path, where the map shall be exported to

**Returns**

true, if export was successful, otherwise false.

**Return type**

bool

**src.osm\_configurator.model.project.config\_phase\_enum module**

**class ConfigPhase(value)**

Bases: *Enum*

This enum stores the different phases of the configuration and is used to restore the last step the user was working on.

DATA\_CONFIG\_PHASE = 'Data Configuration Phase'

CATEGORY\_CONFIG\_PHASE = 'Category Configuration Phase'

REDUCTION\_CONFIG\_PHASE = 'Reduction Configuration Phase'

ATTRACTIVITY\_CONFIG\_PHASE = 'Attractivity Configuration Phase'

AGGREGATION\_CONFIG\_PHASE = 'Aggregation Configuration Phase'

CALCULATION\_CONFIG\_PHASE = 'Calculation Configuration Phase'

**get\_name()**

Getter for the name of the phase.

**Returns**

Name of the Phase.

**Return type**

str

**src.osm\_configurator.model.project.data\_visualizer module**

**class DataVisualizer**

Bases: *object*

This class job is to visualize the cut-out file or data of the project.

**\_\_init\_\_()**

Creates a new instance of the DataVisualizer.

**create\_map(cut\_out)**

This method to create a map from to given cut-out.

**Parameters**

**cut\_out** (*cut\_out\_configuration.CutOutConfiguration*) – The cut-out configuration from which the map should be created.

**Returns**

True if creating the map works, otherwise false.

**Return type**`bool`**create\_boxplot**(*data*)

This method is to visualize the data by creating a boxplot. It is used to visualize the calculated end result via a boxplot.

**Parameters**

**data** (`matplotlib.axes.Axes`) – A plot of the data which we want to visualize.

**Returns**

True if creating the boxplot works, otherwise false.

**Return type**`bool`**src.osm\_configurator.model.project.export module****class Export**(*project*)

Bases: `object`

This class provides different export features based on the current project. Whereby exporting means, saving data from the currently active project somewhere else on the system on the disk.

**\_\_init\_\_**(*project*)

Creates a new instance of the “Export” class.

**Parameters**

**project** (`project.ActiveProject`) – The project to make exports from

**export\_project**(*path*)

Exports the whole project to the given path.

**Parameters**

**path** (`pathlib.Path`) – The path where the project shall be exported to

**Returns**

true, if export was successful, otherwise false.

**Return type**`bool`**export\_configuration**(*path*)

Exports the configuration to the given path. More specific, exports all the categories and their configurations, to the given path.

**Parameters**

**path** (`pathlib.Path`) – The path where the configurations shall be exported to

**Returns**

true, if export was successful, otherwise false.

**Return type**`bool`**export\_calculation**(*path*)

Exports the results of the calculation to the given path. Whereby the calculation are a folder with all the different results from each calculation step in it.

**Parameters**

**path** (*Path*) – The path where the results of the calculation shall be exported to

**Returns**

true, if export was successful, otherwise false.

**Return type**

bool

**export\_map**(*path*)

Exports an HTML-Data with the map in it, to the given path.

**Parameters**

**path** (*Path*) – The path, where the map shall be exported to

**Returns**

true, if export was successful, otherwise false.

**Return type**

bool

**src.osm\_configurator.model.project.project\_io\_handler module**

**class ProjectIOHandler**(*active\_project*)

Bases: `object`

This class handles the I/O of a project. This includes: - loading a project from disk into memory that the user selected in the main menu - creating a project on the disk, if the user selected that

**\_\_init\_\_**(*active\_project*)

Creates a new instance of the ProjectLoader. Therefore, it gets the current active project, which should be loaded if not newly created.

**Parameters**

**active\_project** (`active_project.ActiveProject`) – The project the ProjectLoader shall load.

**build\_project**(*path*)

This method is to build the given project. To do this it reads out the configurations and builds a folder structure on the disk.

**Parameters**

**path** (`pathlib.Path`) – The path pointing towards the project folder.

**Returns**

True if creating the project works, otherwise false.

**Return type**

bool

**load\_project**(*path*)

Loads a project in from the disk into the memory.

**Parameters**

**path** (`pathlib.Path`) – The path pointing towards the project folder.

**Returns**

True if creating the project works, otherwise false.

**Return type**

bool



**src.osm\_configurator.model.project.project\_saver module****class ProjectSaver**(*active\_project*)Bases: `object`

The ProjectSaver is responsible for saving the internal representation of the project onto the disk.

**\_\_init\_\_**(*active\_project*)

Creates a new instance of the ProjectSaver. Therefore, it gets the current active project, which should be loaded if not newly created.

**Parameters****active\_project** (`active_project.ActiveProject`) – The project the ProjectSaver shall load.**save\_project**(*path*)

Stores all the configurations of the project. The information about the configuration of the project are stored to the disk.

**Parameters****path** (`pathlib.Path`) – The path pointing towards the project folder. The data will be stored here**Returns**

True, if the project was stored successfully, False, if an error occurred.

**Return type**`bool`**src.osm\_configurator.model.project.project\_settings module****class ProjectSettings**(*location, project\_name, description*)Bases: `object`

This class saves all the different settings of a project and provides methods to view and change them.

**\_\_init\_\_**(*location, project\_name, description*)

This method creates a new ProjectSettings class with the given settings.

**Parameters**

- **location** (`pathlib.Path`) – The location, the project is stored
- **project\_name** (`str`) – The name of the project
- **description** (`str`) – A description of the project

**get\_location**()

Getter for the location of the Project on the disk.

**Returns**

The location of the project

**Return type**`pathlib.Path`**set\_location**(*new\_location*)

This method changes the location where the project will be stored.

**Parameters**

**new\_location** (*pathlib.Path*) – The new location for the project

**Returns**

true, if location change was successful, false else

**Return type**

bool

**set\_name**(*new\_name*)

This method changes the name of the project.

**Parameters**

**new\_name** (*str*) – The new name of the project

**Returns**

true if change was successful, false else

**Return type**

bool

**get\_name**()

This method returns the name of the project.

**Returns**

name of the project

**Return type**

str

**set\_description**(*new\_description*)

This method changes the description of the project.

**Parameters**

**new\_description** (*str*) – The new description of the project

**Returns**

true if change successful, false else

**Return type**

bool

**get\_description**()

This method returns the description of the project.

**Returns**

The description of the project

**Return type**

str

## Module contents

## Module contents

src.osm\_configurator.view package

## Subpackages

src.osm\_configurator.view.popups package

## Submodules

src.osm\_configurator.view.popups.alert\_pop\_up module

**class** `AlertPopUp`(*message*)

Bases: `CTkToplevel`

This class creates popups, that will pop up in front of the GUI. This instance is an Alert-PopUp. It provides a message and one 'OK' button, to close the PopUp again.

**\_\_init\_\_**(*message*)

This constructor will create an AlertPopUp. It will provide the given message and an 'OK' button to close the PopUp again.

### Parameters

**message** (*str*) – The message that will be shown by the AlertPopUp

src.osm\_configurator.view.popups.yes\_no\_pop\_up module

**class** `YesNoPopUp`(*message, func*)

Bases: `CTkToplevel`

This class creates PopUps, that will pop up in front of the GUI. This instance is a YesNoPopUp: It will provide a message, an 'OK' and an 'Cancel' button. Pressing a button will return the information back to the creating instance and close the PopUp.

**\_\_init\_\_**(*message, func*)

This constructor will create an YesNoPopUp, that will show the given message, as well as an 'OK' and 'Cancel' button. If one of the buttons is pressed or the PopUp closed, it will send a message back via the given function, what button had been pressed. If 'OK' has been pressed, the given function will be called with a boolean = true. If 'Cancel' has been pressed, or the PopUp was closed otherwise, the given function will be called with a boolean = false.

### Parameters

- **message** (*str*) – The message to be shown in the PopUp.
- **func** (*Callable*) – A Function that takes one Boolean and has no return, for the PopUp to send a message back.

### Module contents

#### src.osm\_configurator.view.states package

### Submodules

#### src.osm\_configurator.view.states.main\_window module

**class** `MainWindow`(*control*)

Bases: `object`

This class provides the GUI, the user will be working on. It is made dynamic and can change between different frames, to show different information and buttons to the user. Its job is to just show the frames of different states and create the window the GUI will be used on.

**\_\_init\_\_**(*control*)

This method creates a MainWindow with a connection to the given control.

#### Parameters

**control** (`control_interface.IControl`) – The control the GUI shall be working with, to get access to information on the model.

**change\_state**(*last\_state*, *new\_state*)

This method changes from an old given state to a new given state to show on the MainWindow.

#### Parameters

- **last\_state** (`state.State`) – The state that needs to be removed from the MainWindow
- **new\_state** (`state.State`) – The state that shall be shown by the MainWindow

#### Returns

true, if the state change was successful, false if not.

#### Return type

`bool`

#### src.osm\_configurator.view.states.positioned\_frame module

**class** `PositionedFrame`(*frame*, *column*, *line*, *state\_manager*, *control*)

Bases: `object`

This Class gives a Frame a position, via Coordinates, to tell in what position the Frame wants to be in, when it is shown on a Window.

**\_\_init\_\_**(*frame*, *column*, *line*, *state\_manager*, *control*)

This method creates a PositionedFrame, which is a Frame and coordinates to its Position.

#### Parameters

- **frame** (`top_level_frame.TopLevelFrame`) – The frame you want to give a position
- **column** (`int`) – The Column the Frame shall be placed in
- **line** (`int`) – The Line the Frame shall be placed in
- **state\_manager** (`state_manager.StateManager`) – The StateManager the frame will use to change states if needed

- **control** (`control_interface.IControl`) – The control that a frame shall call, if it needs access to the model

#### **get\_frame()**

This method returns the frame this PositionedFrame holds.

##### **Returns**

The Frame this PositionedFrame holds

##### **Return type**

(`top_level_frame.TopLevelFrame`)

#### **get\_column()**

This method Returns the column the frame is placed in.

##### **Returns**

The Column the Frame is placed in.

##### **Return type**

(`int`)

#### **get\_line()**

This method returns the line the frame is placed in.

##### **Returns**

The Line the frame is placed in

##### **Return type**

(`int`)

### **src.osm\_configurator.view.states.state module**

**class State**(*active\_frames, own\_state\_name, default\_left, default\_right*)

Bases: `object`

This class models a state. A State consist of - a list of frames, that shall be visible on a window, when this state is active - a default state to its right - a default state to its left All states have one state to their left and to their right, to model the basic state change. Those states can be 'none' in order so signify that there is no further right or left.

**\_\_init\_\_**(*active\_frames, own\_state\_name, default\_left, default\_right*)

This method creates a new state, that holds the given frames, has the given state name, and has a default left and right.

##### **Parameters**

- **active\_frames** (`list[positioned_frame.PositionedFrame]`) – A list of frames, that this state holds
- **own\_state\_name** (`state_name_enum.StateName`) – The name that defines this state
- **default\_left** (`state_name_enum.StateName`) – The name of the state on this states left
- **default\_right** (`state_name_enum.StateName`) – The name of the state on this states right

#### **get\_active\_frames()**

The list of frames this state holds and shall be shown on a window, if it is active.

**Returns**

List of frames this state holds

**Return type**

*list[[positioned\\_frame.PositionedFrame](#)]*

**get\_default\_left()**

The name of the state on this states left.

**Returns**

This states left state name

**Return type**

*([state\\_name\\_enum.StateName](#))*

**get\_default\_right()**

The id of the state on this states right.

**Returns**

This states right state name

**Return type**

*([state\\_name\\_enum.StateName](#))*

**get\_state\_name()**

The name of this state.

**Returns**

The name of this state

**Return type**

*([state\\_name\\_enum.StateName](#))*

**equals(*state*)**

Test if two states are equal. Two states are defined as equal, if their name is equal.

**Parameters**

**state** ([state.State](#)) – The state you want to know if it is equal to this one

**Returns**

true if the given state and this state ist equal. false if not

**Return type**

*bool*

## **src.osm\_configurator.view.states.state\_manager module**

### **class StateManager(*control, main\_window*)**

Bases: [object](#)

This class manages the different states, that can be shown on a window. It knows what state is currently active and provides methods to change the state.

#### **\_\_init\_\_(*control, main\_window*)**

This method creates a StateManager, that will control what state is currently active and manages the changes between states. It will create all states, as well all the frames that exist and put them in the state they belong.

**Parameters**

- **control** ([control\\_interface.IControl](#)) – The connection to the control, so the frames of each state can access the model.

- **main\_window** (`main_window.MainWindow`) – The MainWindow, where the frames of the state shall be shown on.

#### **default\_go\_right()**

This method changes to the State that is the default\_right state of the current one.

##### **Returns**

true, if a state change was successfully made, false if there was no state change or something went wrong

##### **Return type**

`bool`

#### **default\_go\_left()**

This method changes to the state that is the default\_left State of the current one.

##### **Returns**

true, if a state change was successfully made, false if there was no state change or something went wrong

##### **Return type**

`bool`

#### **change\_state(new\_state)**

This method changes to the given state and deactivate the old one.

##### **Parameters**

**new\_state** (`state_name_enum.StateName`) – The id of the new state that shall be activated.

##### **Returns**

true if state change was successful, false if not.

##### **Return type**

`bool`

#### **get\_state()**

This method returns the currently active state.

##### **Returns**

the currently active state.

##### **Return type**

`state.State`

#### **close\_program()**

This method closes the program and shuts the whole application down.

### **src.osm\_configurator.view.states.state\_name\_enum module**

#### **class StateName(value)**

Bases: `Enum`

This enum saves the different state possibilities that exist, to define a state by this enum and not, by a number. This enum gives a state a name.

**MAIN\_MENU** = 1

**CREATE\_PROJECT** = 2

DATA = 3  
CATEGORY = 4  
REDUCTION = 5  
ATTRACTIVITY\_EDIT = 6  
ATTRACTIVITY\_VIEW = 7  
AGGREGATION = 8  
CALCULATION = 9  
SETTINGS = 10

## Module contents

`src.osm_configurator.view.toplevelframes` package

## Submodules

`src.osm_configurator.view.toplevelframes.aggregation_frame` module

**class** `AggregationFrame`(*state\_manager, control*)

Bases: `TopLevelFrame`

This frame shows the aggregation page the user will interact on. This window provides the checkboxes to choose calculation methods and methods on how the aggregation will be calculated.

**\_\_init\_\_**(*state\_manager, control*)

This method creates an `AggregationFrame`, that will be used to edit the aggregation method.

### Parameters

- **state\_manager** (`state_manager.StateManager`) – The `StateManager`, the frame will call, when it wants to change to another state.
- **control** (`control_interface.IControl`) – The control, the Frame will call, to get access to the model.

**activate**()

Tells the current frame to activate and collect all the data it needs.

### Returns

True, if activation was successful, false else

### Return type

`bool`



**src.osm\_configurator.view.toplevelframes.attractivity\_edit\_frame module****class AttractivityEditFrame**(*state\_manager, control*)Bases: *TopLevelFrame*

This frame lets the user edit, create and delete attractivity attributes for the categories. Two drop-down menus will be shown: One will select the category, the other the attractivity attributes. Editing options for the attractivity attributes will be offered in a textbox to change the name. Smaller boxes provide the means of changing the factors for different attributes. Two buttons provide creation and deletion tools.

**\_\_init\_\_**(*state\_manager, control*)

This method creates an AttractivityEditFrame, where the attractivity attributes of categories can be edited, created or be deleted.

**Parameters**

- **state\_manager** (*state\_manager.StateManager*) – The StateManager the frame will call, when it wants to change to another state.
- **control** (*control\_interface.IControl*) – The control the frame will call to get access to the model.

**activate**()

Tells the current frame to activate and collect all the data it needs.

**Returns**

True, if activation was successful, false else

**Return type**

bool

**src.osm\_configurator.view.toplevelframes.attractivity\_view\_frame module****class AttractivityViewFrame**(*state\_manager, control*)Bases: *TopLevelFrame*

This frame shows a list with all categories, their attractivity attributes and how they are calculated. This is only a visualisation and therefore a non-edit Frame.

**\_\_init\_\_**(*state\_manager, control*)

This method creates an AttractivityViewFrame showing a IList of containing all categories, their according attractivity attributes and how they are calculated.

**Parameters**

- **state\_manager** (*state\_manager.StateManager*) – The StateManager the Frame will call, if it tries to switch to another atate.
- **control** (*control\_interface.IControl*) – The control the frame will call to get access to the model.

**activate**()

Tells the current frame to activate and collect all the data it needs.

**Returns**

True, if activation was successful, false else

**Return type**

bool

### src.osm\_configurator.view.toplevelframes.calculation\_frame module

**class** `CalculationFrame`(*state\_manager, control*)

Bases: `TopLevelFrame`

This frame lets the user start the calculation from a selected phase, indicated by different buttons. Once a calculation is started, there will be a progressbar shown, the different buttons will be deactivated and the current calculation-phase will be shown. A cancel-Button is provided to stop the calculation. The CalculationFrame shows popups, if an error accures in the calculations.

**\_\_init\_\_**(*state\_manager, control*)

This method creates a CalculationFrame that will let the user start the calculation and shows the calculation progress.

#### Parameters

- **state\_manager** (`state_manager.StateManager`) – The StateManager the frame will call, if it wants to switch to another state.
- **control** (`control_interface.IControl`) – The control the frame will call to get access to the model.

**activate**()

Tells the current frame to activate and collect all the data it needs.

#### Returns

True, if activation was successful, false else

#### Return type

`bool`

### src.osm\_configurator.view.toplevelframes.category\_frame module

**class** `CategoryFrame`(*state\_manager, control*)

Bases: `TopLevelFrame`

This frame lets the user create, delete and edit categories. It shows the name of a category, as well as their black- and white-List. Categories also can be turned on and off with Checkboxes. There will also be key-recommendations be shown for the black- and white-List.

**\_\_init\_\_**(*state\_manager, control*)

This method creates an CategoryFrame so the user can create, delete and edit categories.

#### Parameters

- **state\_manager** (`state_manager.StateManager`) – The StateManager the frame will call, when it wants to change to another state.
- **control** (`control_interface.IControl`) – The control the frame will call to get access to the model.

**activate**()

Tells the current frame to activate and collect all the data it needs.

#### Returns

True, if activation was successful, false else

#### Return type

`bool`

**src.osm\_configurator.view.toplevelframes.create\_project\_frame module****class CreateProjectFrame**(*state\_manager, control*)Bases: *TopLevelFrame*

This frame shows the project creation page to the User. A name, a description and a path for storing the project can be set here. The user can cancel the creation-process.

**\_\_init\_\_**(*state\_manager, control*)

This method creates a CreateProjectFrame where a user can create a new project.

**Parameters**

- **state\_manager** (*state\_manager.StateManager*) – The StateManager the frame will call, if it wants to change to another State.
- **control** (*control\_interface.IControl*) – The frame will call the control, to gain access to the model.

**activate**()

Tells the current frame to activate and collect all the data it needs.

**Returns**

True, if activation was successful, false else

**Return type**

bool

**src.osm\_configurator.view.toplevelframes.data\_frame module****class DataFrame**(*state\_manager, control*)Bases: *TopLevelFrame*

This frame lets the user edit various following Data: - Selection of the OSM-Data - Selection of the Cut-Out - Select, if buildings on the edge shall be included or not - A download button to download the OSM data after a cut-out was selected - Copy in category configurations

**\_\_init\_\_**(*state\_manager, control*)

This method creates a DataFrame, that lets the User input data into the project.

**Parameters**

- **state\_manager** (*state\_manager.StateManager*) – The frame will call the StateManager, if it wants to switch states.
- **control** (*control\_interface.IControl*) – The frame will call the control, to gain access to the model.

**activate**()

Tells the current frame to activate and collect all the data it needs.

**Returns**

True, if activation was successful, false else

**Return type**

bool

### src.osm\_configurator.view.toplevelframes.main\_menu\_frame module

**class MainMenuFrame**(*state\_manager, control*)

Bases: *TopLevelFrame*

This frame shows the application's main menu. The user can create a new project, or load an already existing project. Projects stored in the default folder will be shown in a list and can be selected / opened.

**\_\_init\_\_**(*state\_manager, control*)

This method creates a MainMenuFrame showing the MainMenu of the application.

#### Parameters

- **state\_manager** (*state\_manager.StateManager*) – The frame will call the StateManager, if it wants to switch states.
- **control** (*control\_interface.IControl*) – The frame will call the control to gain access to the model.

**activate**()

Tells the current frame to activate and collect all the data it needs.

#### Returns

True, if activation was successful, false else

#### Return type

*bool*

### src.osm\_configurator.view.toplevelframes.project\_foot\_frame module

**class ProjectFootFrame**(*state\_manager, control*)

Bases: *TopLevelFrame*

This frame shows two arrows on the bottom of the Window. The user can navigate the pipeline by going left or right.

**\_\_init\_\_**(*state\_manager, control*)

This method creates a ProjectFootFrame, that lets the user navigate the pipeline by going left or right.

#### Parameters

- **state\_manager** (*state\_manager.StateManager*) – The StateManager the frame will call, if it wants to switch states.
- **control** (*control\_interface.IControl*) – The control the frame will call to get access to the model.

**activate**()

Tells the current frame to activate and collect all the data it needs.

#### Returns

True, if activation was successful, false else

#### Return type

*bool*

**src.osm\_configurator.view.toplevelframes.project\_head\_frame module****class ProjectHeadFrame**(*state\_manager, control*)Bases: *TopLevelFrame*

This frame shows the header pipeLine of the application, if a project is opened. Functionality the user can use: - Exit to the main menu - Save the project - Go to the settings - Change between different frames to edit configurations - Use exports

This frame is always on the top of the window. below it will be presented a frame to edit some part of the project and below that Frame will be a FootFrame. Exceptions are the MainMenu and the creation of a new project without this header.

**\_\_init\_\_**(*state\_manager, control*)

This method creates a ProjectHeadFrame, letting the user navigate the pipeline and exit back to the main menu. The user can also open the settings, save the project or export the project.

**Parameters**

- **state\_manager** (*state\_manager.StateManager*) – The frame will call the StateManager, if it wants to switch states.
- **control** (*control\_interface.IControl*) – The frame will call the control, to gain access to the model.

**activate**()

Tells the current frame to activate and collect all the data it needs.

**Returns**

True, if activation was successful, false else

**Return type**

bool

**src.osm\_configurator.view.toplevelframes.reduction\_frame module****class ReductionFrame**(*state\_manager, control*)Bases: *TopLevelFrame*

This frame lets the user edit the reduction of all the categories. It will consist of a list on the left to choose a category. On the right will be two sub-frames to change inbetween. On the right are two interchangeable sub-frames: One frame provides the configuration-options on how to calculate the Reduction. The other frame provides the default calculation-values.

**\_\_init\_\_**(*state\_manager, control*)

This method creates a ReductionFrame, that lets the user edit the reduction of all the categories.

**Parameters**

- **state\_manager** (*state\_manager.StateManager*) – The frame will call the StateManager, if it wants to switch states.
- **control** (*control\_interface.IControl*) – The control the frame will call to get access to the model.

**activate**()

Tells the current frame to activate and collect all the data it needs.

**Returns**

True, if activation was successful, false else

**Return type**

bool

**src.osm\_configurator.view.toplevelframes.settings\_frame module**

**class SettingsFrame**(*state\_manager, control*)

Bases: *TopLevelFrame*

This frame shows the user the settings for: - The application - The current project

It either can be both or only the application settings.

**\_\_init\_\_**(*state\_manager, control*)

This method creates a SettingsFrame, that lets the user set the application and project settings.

**Parameters**

- **state\_manager** (*state\_manager.StateManager*) – The StateManager the frame will call, if it wants to switch states.
- **control** (*control\_interface.IControl*) – The control the frame will call to get access to the model.

**activate**()

Tells the current frame to activate and collect all the data it needs.

**Returns**

True, if activation was successful, false else

**Return type**

bool

**src.osm\_configurator.view.toplevelframes.top\_level\_frame module**

**class TopLevelFrame**(*state\_manager, control*)

Bases: *ABC*

This class describes a frame, that has a fully developed functionality and that can be placed on a window. A TopLevelFrame might have manageable frames below him.

**\_\_init\_\_**(*state\_manager, control*)

This method defines how a TopLevelFrame is created.

**Parameters**

- **state\_manager** (*state\_manager.StateManager*) – The StateManager the frame will call, if it wants to switch states.
- **control** (*control\_interface.IControl*) – The control the frame will call, to gain access to the model.

**abstract activate**()

Tells the current frame to activate and collect all the data it needs.

**Returns**

True, if activation was successful, false else

**Return type**`bool`**Module contents****src.osm\_configurator.view.utilityframes package****Submodules****src.osm\_configurator.view.utilityframes.export\_frame module****class** `ExportFrame`(*parent\_frame, control*)Bases: `object`

The ExportFrame provides a dropdown menu that providing the following Options: - project export - calculation export - Configurations Export

`__init__`(*parent\_frame, control*)

This method creates an ExportFrame, that provides the user with different export options.

**Parameters**

- **parent\_frame** (`project_head_frame.ProjectHeadFrame`) – The parent of the ExportFrame is the HeadFrame, where the export feature is located.
- **control** (`control_interface.IControl`) – The control the frame calls, to gain access to the model to export.

**src.osm\_configurator.view.utilityframes.reduction\_calculation\_frame module****class** `ReductionCalculationFrame`(*parent, control*)Bases: `object`

This frame provides the ability to the user to set how the calculation of the reduction of a category will be done. This is a subframe from the ReductionFrame.

`__init__`(*parent, control*)

This method creates a ReductionCalculationFrame, that lets the user edit the calculation of the reduction of Categories. :param parent: This is the parent frame of this frame. The frame will be located here. :type parent: `reduction_frame.ReductionFrame` :param control: The control the frame will call to get access to the model. :type control: `control_interface.IControl`

**activate()**

Tells the current frame to activate and collect all the data it needs.

**Returns**

True, if activation was successful, false else

**Return type**`bool`

## src.osm\_configurator.view.utilityframes.reduction\_default\_value\_frame module

**class** `ReductionDefaultValueFrame`(*parent, control*)

Bases: `object`

This frame shows a list of tags in a priority order, that can be expanded by adding or removing tags. These tags can hold default values on attributes, that can be used in the calculation.

**\_\_init\_\_**(*parent, control*)

This method creates a `ReductionDefaultValueFrame` where the User can edit default-values on tags for categories.

### Parameters

- **parent** (`reduction_frame.ReductionFrame`) – This is the parent frame of this frame. The frame will be located here.
- **control** (`control_interface.IControl`) – The control the frame calls, to gain access to the model.

**activate**()

Tells the current frame to activate and collect all the data it needs.

### Returns

True, if activation was successful, false else

### Return type

`bool`

## src.osm\_configurator.view.utilityframes.settings\_application\_frame module

**class** `SettingsApplicationFrame`(*parent, control*)

Bases: `object`

This frame shows the settings of the application.

**\_\_init\_\_**(*parent, control*)

This method creates a `SettingsApplicationFrame`, showing the settings of the application.

### Parameters

- **parent** (`settings_frame.SettingsFrame`) – The parent of this frame, where this frame will be located.
- **control** (`control_interface.IControl`) – The control the frame will call, to gain access to the Model and application settings.

**activate**()

Tells the current frame to activate and collect all the data it needs.

### Returns

True, if activation was successful, false else

### Return type

`bool`



**src.osm\_configurator.view.utilityframes.settings\_project\_frame module****class SettingsProjectFrame**(parent, control)Bases: `object`

This frame shows the current project settings.

**\_\_init\_\_**(parent, control)

This method creates a SettingsProjectFrame, showing the current project settings.

**Parameters**

- **parent** (`settings_frame.SettingsFrame`) – The parent of this frame, where this frame will be located.
- **control** (`control_interface.IControl`) – The control the frame will call, to gain access to the model.

**activate**()

Tells the current frame to activate and collect all the data it needs.

**Returns**

True, if activation was successful, false else

**Return type**`bool`**src.osm\_configurator.view.utilityframes.tag\_list\_frame module****class TagListFrame**(entries)Bases: `object`

This frame shows a textbox with a List of editable strings.

**\_\_init\_\_**(entries)

This method creates a textbox with the given entries.

**Parameters****list[str]** (entries) – A list of strings, that will be written in the textbox.**set\_text\_list**(entries)

Replaces all shown textbox entries with the given text.

**Parameters****list[str]** (entries) – A list of strings, that will be shown on the textbox.**Returns**

True if the replacement was successful, false else.

**Return type**`bool`**get\_text\_list**()

This method returns a list of strings containing the current textbox entries

**Returns**

List of strings containing the current textbox entries

**Return type**`list[str]`

**src.osm\_configurator.view.utilityframes.tag\_list\_priority\_frame module****class TagListPriorityFrame**(*entries*)Bases: `object`

This frame shows a list of tags (represented as strings). The tag-priority can be changed with arrows. The higher an entry is on the list, the lower its priority. A non-deletable DEFAULT-entry will always have the lowest priority.

**\_\_init\_\_**(*entries*)

This method will create a TagListPriorityFrame, showing a list of the given entries, ordered like the priorities.

**Parameters****list[str]** (*entries*) – List of strings, that shall be the entries on the priority list.**set\_tag\_list**(*entries*)

Replaces all the entries with the new given entries, ordered in priority as the given List of strings is.

**Parameters****list[str]** (*entries*) – The list of strings, that shall be the entries on the priority list.**Returns**

True, if the replacement was successful, false if not.

**Return type**`bool`**get\_tag\_list**()

Returns the current list of entries the priority list holds, ordered from lowest to highest.

**Returns**

The entry list of strings on the list, ordered from the lowest to highest priority.

**Return type**`list[str]`**Module contents****Module contents****Module contents****Module contents**

## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`



## PYTHON MODULE INDEX

### S

src, 102	src.osm_configurator.model.parser.calculation_parser_interface, 48
src.osm_configurator, 102	src.osm_configurator.model.parser.category_parser, 48
src.osm_configurator.control, 33	src.osm_configurator.model.parser.category_parser_interface, 49
src.osm_configurator.control.aggregation_controller, 5	src.osm_configurator.model.parser.cutOut_parser, 49
src.osm_configurator.control.application_controller, 6	src.osm_configurator.model.parser.cutOut_parser_interface, 50
src.osm_configurator.control.calculation_controller, 7	src.osm_configurator.model.parser.osm_data_parser, 50
src.osm_configurator.control.category_controller, 8	src.osm_configurator.model.parser.osm_data_parser_interface, 51
src.osm_configurator.control.control, 9	src.osm_configurator.model.project, 87
src.osm_configurator.control.control_interface, 18	src.osm_configurator.model.project.active_project, 76
src.osm_configurator.control.cut_out_controller, 26	src.osm_configurator.model.project.calculation, 60
src.osm_configurator.control.data_visualization_controller, 27	src.osm_configurator.model.project.calculation.aggregation, 51
src.osm_configurator.control.export_controller, 28	src.osm_configurator.model.project.calculation.aggregation, 52
src.osm_configurator.control.osm_data_controller, 29	src.osm_configurator.model.project.calculation.attractiveness, 53
src.osm_configurator.control.project_controller, 30	src.osm_configurator.model.project.calculation.building_order, 53
src.osm_configurator.control.settings_controller, 32	src.osm_configurator.model.project.calculation.calculation, 54
src.osm_configurator.model, 87	src.osm_configurator.model.project.calculation.calculation, 54
src.osm_configurator.model.application, 47	src.osm_configurator.model.project.calculation.calculation, 55
src.osm_configurator.model.application.application, 33	src.osm_configurator.model.project.calculation.calculation, 56
src.osm_configurator.model.application.application_interface, 39	src.osm_configurator.model.project.calculation.geo_data_ph, 57
src.osm_configurator.model.application.application_settings, 45	src.osm_configurator.model.project.calculation.osm_file_co, 57
src.osm_configurator.model.application.passive_project, 46	src.osm_configurator.model.project.calculation.osm_file_fo, 58
src.osm_configurator.model.application.recommender_system, 47	src.osm_configurator.model.project.calculation.reduction_p, 57
src.osm_configurator.model.parser, 51	
src.osm_configurator.model.parser.calculation_parser, 47	

58	88
src.osm_configurator.model.project.calculations	src.osm_configurator.view.states.state, 89
59	src.osm_configurator.view.states.state_manager,
src.osm_configurator.model.project.calculation.tag_filter	filter_phase,
59	src.osm_configurator.view.states.state_name_enum,
src.osm_configurator.model.project.calculation.traffic	cell,
60	src.osm_configurator.view.toplevelframes, 99
src.osm_configurator.model.project.config_phase_enum	src.osm_configurator.view.toplevelframes.aggregation_frame,
82	92
src.osm_configurator.model.project.configuration	src.osm_configurator.view.toplevelframes.attractivity_edit,
76	93
src.osm_configurator.model.project.configuration	src.osm_configurator.view.toplevelframes.attractivity_view,
60	93
src.osm_configurator.model.project.configuration	src.osm_configurator.view.toplevelframes.calculation_frame,
61	94
src.osm_configurator.model.project.configuration	src.osm_configurator.view.toplevelframes.category_frame,
63	94
src.osm_configurator.model.project.configuration	src.osm_configurator.view.toplevelframes.create_project_fr,
63	95
src.osm_configurator.model.project.configuration	src.osm_configurator.view.toplevelframes.data_frame,
64	95
src.osm_configurator.model.project.configuration	src.osm_configurator.view.toplevelframes.main_menu_frame,
68	96
src.osm_configurator.model.project.configuration	src.osm_configurator.view.toplevelframes.project_foot_fr,
69	96
src.osm_configurator.model.project.configuration	src.osm_configurator.view.toplevelframes.project_head_fr,
72	97
src.osm_configurator.model.project.configuration	src.osm_configurator.view.toplevelframes.reduction_frame,
73	97
src.osm_configurator.model.project.configuration	src.osm_configurator.view.toplevelframes.settings_frame,
74	98
src.osm_configurator.model.project.configuration	src.osm_configurator.view.toplevelframes.top_level_frame,
75	98
src.osm_configurator.model.project.configuration	src.osm_data_configuration, utilityframes, 102
75	src.osm_configurator.view.utilityframes.export_frame,
src.osm_configurator.model.project.data_visualizer,	99
82	src.osm_configurator.view.utilityframes.reduction_calculat,
src.osm_configurator.model.project.export,	83 99
src.osm_configurator.model.project.project_io_handler,	src.osm_configurator.view.utilityframes.reduction_default,
84	100
src.osm_configurator.model.project.project_saver,	src.osm_configurator.view.utilityframes.settings_applicati,
85	100
src.osm_configurator.model.project.project_settings	src.osm_configurator.view.utilityframes.settings_project_f,
85	101
src.osm_configurator.view,	102
src.osm_configurator.view.popups,	88 101
src.osm_configurator.view.popups.alert_pop_up,	src.osm_configurator.view.utilityframes.tag_list_priority,
87	102
src.osm_configurator.view.popups.yes_no_pop_up,	
87	
src.osm_configurator.view.states,	92
src.osm_configurator.view.states.main_window,	
88	
src.osm_configurator.view.states.positioned_frame,	

## Symbols

[\\_\\_init\\_\\_\(\)](#) (*ActiveProject* method), 76  
[\\_\\_init\\_\\_\(\)](#) (*AggregationConfiguration* method), 60  
[\\_\\_init\\_\\_\(\)](#) (*AggregationController* method), 5  
[\\_\\_init\\_\\_\(\)](#) (*AggregationFrame* method), 92  
[\\_\\_init\\_\\_\(\)](#) (*AlertPopUp* method), 87  
[\\_\\_init\\_\\_\(\)](#) (*Application* method), 33  
[\\_\\_init\\_\\_\(\)](#) (*ApplicationController* method), 6  
[\\_\\_init\\_\\_\(\)](#) (*ApplicationSettings* method), 45  
[\\_\\_init\\_\\_\(\)](#) (*AttractivityAttribute* method), 61  
[\\_\\_init\\_\\_\(\)](#) (*AttractivityEditFrame* method), 93  
[\\_\\_init\\_\\_\(\)](#) (*AttractivityViewFrame* method), 93  
[\\_\\_init\\_\\_\(\)](#) (*BuildingOnEdgeManager* method), 53  
[\\_\\_init\\_\\_\(\)](#) (*CalculationController* method), 7  
[\\_\\_init\\_\\_\(\)](#) (*CalculationFrame* method), 94  
[\\_\\_init\\_\\_\(\)](#) (*CalculationManager* method), 54  
[\\_\\_init\\_\\_\(\)](#) (*CalculationParser* method), 47  
[\\_\\_init\\_\\_\(\)](#) (*Category* method), 64  
[\\_\\_init\\_\\_\(\)](#) (*CategoryController* method), 8  
[\\_\\_init\\_\\_\(\)](#) (*CategoryFrame* method), 94  
[\\_\\_init\\_\\_\(\)](#) (*CategoryManager* method), 68  
[\\_\\_init\\_\\_\(\)](#) (*CategoryParser* method), 48, 50  
[\\_\\_init\\_\\_\(\)](#) (*ConfigurationManager* method), 69  
[\\_\\_init\\_\\_\(\)](#) (*Control* method), 9  
[\\_\\_init\\_\\_\(\)](#) (*CreateProjectFrame* method), 95  
[\\_\\_init\\_\\_\(\)](#) (*CutOutConfiguration* method), 72  
[\\_\\_init\\_\\_\(\)](#) (*CutOutController* method), 26  
[\\_\\_init\\_\\_\(\)](#) (*DataFrame* method), 95  
[\\_\\_init\\_\\_\(\)](#) (*DataVisualizationController* method), 27  
[\\_\\_init\\_\\_\(\)](#) (*DataVisualizer* method), 82  
[\\_\\_init\\_\\_\(\)](#) (*DefaultValueEntry* method), 74  
[\\_\\_init\\_\\_\(\)](#) (*DownloadData* method), 75  
[\\_\\_init\\_\\_\(\)](#) (*Export* method), 83  
[\\_\\_init\\_\\_\(\)](#) (*ExportController* method), 28  
[\\_\\_init\\_\\_\(\)](#) (*ExportFrame* method), 99  
[\\_\\_init\\_\\_\(\)](#) (*MainMenuFrame* method), 96  
[\\_\\_init\\_\\_\(\)](#) (*MainWindow* method), 88  
[\\_\\_init\\_\\_\(\)](#) (*OSMDataConfiguration* method), 75  
[\\_\\_init\\_\\_\(\)](#) (*OSMDataController* method), 29  
[\\_\\_init\\_\\_\(\)](#) (*OSMFileConverter* method), 57  
[\\_\\_init\\_\\_\(\)](#) (*PassiveProject* method), 46  
[\\_\\_init\\_\\_\(\)](#) (*PositionedFrame* method), 88

[\\_\\_init\\_\\_\(\)](#) (*ProjectController* method), 30  
[\\_\\_init\\_\\_\(\)](#) (*ProjectFootFrame* method), 96  
[\\_\\_init\\_\\_\(\)](#) (*ProjectHeadFrame* method), 97  
[\\_\\_init\\_\\_\(\)](#) (*ProjectIOHandler* method), 84  
[\\_\\_init\\_\\_\(\)](#) (*ProjectSaver* method), 85  
[\\_\\_init\\_\\_\(\)](#) (*ProjectSettings* method), 85  
[\\_\\_init\\_\\_\(\)](#) (*RecommenderSystem* method), 47  
[\\_\\_init\\_\\_\(\)](#) (*ReductionCalculationFrame* method), 99  
[\\_\\_init\\_\\_\(\)](#) (*ReductionDefaultValueFrame* method), 100  
[\\_\\_init\\_\\_\(\)](#) (*ReductionFrame* method), 97  
[\\_\\_init\\_\\_\(\)](#) (*SettingsApplicationFrame* method), 100  
[\\_\\_init\\_\\_\(\)](#) (*SettingsController* method), 32  
[\\_\\_init\\_\\_\(\)](#) (*SettingsFrame* method), 98  
[\\_\\_init\\_\\_\(\)](#) (*SettingsProjectFrame* method), 101  
[\\_\\_init\\_\\_\(\)](#) (*SplitUpFile* method), 59  
[\\_\\_init\\_\\_\(\)](#) (*State* method), 89  
[\\_\\_init\\_\\_\(\)](#) (*StateManager* method), 90  
[\\_\\_init\\_\\_\(\)](#) (*TagListFrame* method), 101  
[\\_\\_init\\_\\_\(\)](#) (*TagListPriorityFrame* method), 102  
[\\_\\_init\\_\\_\(\)](#) (*TopLevelFrame* method), 98  
[\\_\\_init\\_\\_\(\)](#) (*TrafficCell* method), 60  
[\\_\\_init\\_\\_\(\)](#) (*YesNoPopUp* method), 87

## A

[activate\(\)](#) (*AggregationFrame* method), 92  
[activate\(\)](#) (*AttractivityEditFrame* method), 93  
[activate\(\)](#) (*AttractivityViewFrame* method), 93  
[activate\(\)](#) (*CalculationFrame* method), 94  
[activate\(\)](#) (*Category* method), 64  
[activate\(\)](#) (*CategoryFrame* method), 94  
[activate\(\)](#) (*CreateProjectFrame* method), 95  
[activate\(\)](#) (*DataFrame* method), 95  
[activate\(\)](#) (*MainMenuFrame* method), 96  
[activate\(\)](#) (*ProjectFootFrame* method), 96  
[activate\(\)](#) (*ProjectHeadFrame* method), 97  
[activate\(\)](#) (*ReductionCalculationFrame* method), 99  
[activate\(\)](#) (*ReductionDefaultValueFrame* method), 100  
[activate\(\)](#) (*ReductionFrame* method), 97  
[activate\(\)](#) (*SettingsApplicationFrame* method), 100  
[activate\(\)](#) (*SettingsFrame* method), 98

activate() (*SettingsProjectFrame* method), 101  
activate() (*TopLevelFrame* method), 98  
ActiveProject (class in *src.osm\_configurator.model.project.active\_project*), 76  
add\_attractivity\_attribute() (*Category* method), 66  
add\_default\_value\_entry() (*Category* method), 67  
AGGREGATION (*StateName* attribute), 92  
AGGREGATION\_CONFIG\_PHASE (*ConfigPhase* attribute), 82  
AGGREGATION\_PHASE (*CalculationPhase* attribute), 55  
AggregationConfiguration (class in *src.osm\_configurator.model.project.configuration.aggregation\_configuration*), 60  
AggregationController (class in *src.osm\_configurator.control.aggregation\_controller*), 5  
AggregationFrame (class in *src.osm\_configurator.view.toplevelframes.aggregation\_frame*), 92  
AggregationMethod (class in *src.osm\_configurator.model.project.calculation.aggregation\_method\_enum*), 51  
AggregationPhase (class in *src.osm\_configurator.model.project.calculation.aggregation\_phase*), 52  
AlertPopUp (class in *src.osm\_configurator.view.popups.alert\_pop\_up*), 87  
Application (class in *src.osm\_configurator.model.application.application*), 33  
ApplicationController (class in *src.osm\_configurator.control.application\_controller*), 6  
ApplicationSettings (class in *src.osm\_configurator.model.application.application\_settings*), 45  
ATTRACTIVITY\_CONFIG\_PHASE (*ConfigPhase* attribute), 82  
ATTRACTIVITY\_EDIT (*StateName* attribute), 92  
ATTRACTIVITY\_PHASE (*CalculationPhase* attribute), 55  
ATTRACTIVITY\_VIEW (*StateName* attribute), 92  
AttractivityAttribute (class in *src.osm\_configurator.model.project.configuration.attractivity\_attribute*), 61  
AttractivityEditFrame (class in *src.osm\_configurator.view.toplevelframes.attractivity\_edit\_frame*), 93  
AttractivityPhase (class in *src.osm\_configurator.model.project.calculation.attractivity\_phase*), 53  
AttractivityViewFrame (class in *src.osm\_configurator.view.toplevelframes.attractivity\_view\_frame*), 54  
Attribute (class in *src.osm\_configurator.model.project.configuration.attribute*), 93  
AVERAGE (*AggregationMethod* attribute), 51  
**B**  
build\_project() (*ProjectIOHandler* method), 84  
BuildingOnEdgeManager (class in *src.osm\_configurator.model.project.calculation.building\_on\_edge*), 53  
BUILDINGS\_ON\_EDGE\_ACCEPTED (*CutOutMode* attribute), 73  
BUILDINGS\_ON\_EDGE\_NOT\_ACCEPTED (*CutOutMode* attribute), 73  
BZZ (*OSMFileFormat* attribute), 58  
**C**  
calculate() (*AggregationPhase* method), 52  
calculate() (*AttractivityPhase* method), 53  
calculate() (*GeoDataPhase* method), 57  
calculate() (*ICalculationPhase* method), 55  
calculate() (*ReductionPhase* method), 58  
calculate() (*TagFilterPhase* method), 59  
calculate\_aggregation() (*AggregationMethod* method), 52  
CALCULATE\_BUILDING\_AREA (*CalculationMethodOfArea* attribute), 63  
CALCULATE\_SITE\_AREA (*CalculationMethodOfArea* attribute), 63  
CALCULATION (*StateName* attribute), 92  
CALCULATION\_CONFIG\_PHASE (*ConfigPhase* attribute), 82  
CalculationController (class in *src.osm\_configurator.control.calculation\_controller*), 7  
CalculationFrame (class in *src.osm\_configurator.view.toplevelframes.calculation\_frame*), 94  
CalculationManager (class in *src.osm\_configurator.model.project.calculation.calculation\_manager*), 54  
CalculationMethodOfArea (class in *src.osm\_configurator.model.project.configuration.calculation\_method\_of\_area*), 63  
CalculationParser (class in *src.osm\_configurator.model.parser.calculation\_parser*), 47  
CalculationParserInterface (class in *src.osm\_configurator.model.parser.calculation\_parser\_interface*), 48  
CalculationPhase (class in *src.osm\_configurator.model.project.calculation.calculation\_phase*), 54



CalculationState (class in create\_boxplot() (ActiveProject method), 80  
src.osm\_configurator.model.project.calculation.calculation\_boxplot() (Application method), 37  
56 create\_boxplot() (DataVisualizer method), 83  
cancel\_calculation() (CalculationManager create\_boxplot() (IApplication method), 43  
method), 54 create\_category() (ActiveProject method), 79  
cancel\_calculations() (CalculationController create\_category() (Application method), 36  
method), 7 create\_category() (CategoryController method), 9  
cancel\_calculations() (Control method), 15 create\_category() (CategoryManager method), 68  
cancel\_calculations() (IControl method), 23 create\_category() (ConfigurationManager method),  
Category (class in src.osm\_configurator.model.project.configuration.category), 64  
CATEGORY (StateName attribute), 92 create\_category() (Control method), 13  
CATEGORY\_CONFIG\_PHASE (ConfigPhase attribute), 82 create\_category() (IApplication method), 42  
CategoryController (class in create\_map() (ActiveProject method), 80  
src.osm\_configurator.control.category\_controller), 8 create\_map() (Application method), 37  
create\_map() (DataVisualizer method), 82  
CategoryFrame (class in create\_map() (IApplication method), 43  
src.osm\_configurator.view.toplevelframes.category\_frame), 94  
CATEGORY\_PROJECT (StateName attribute), 91  
create\_project() (Application method), 34  
CategoryManager (class in create\_project() (Control method), 10  
src.osm\_configurator.model.project.configuration.category\_manager), 68  
create\_project() (IApplication method), 39  
create\_project() (IControl method), 18  
CategoryParser (class in create\_project() (ProjectController method), 30  
src.osm\_configurator.model.parser.category\_parser), 48  
createProjectFrame (class in  
src.osm\_configurator.view.toplevelframes.create\_project\_frame), 95  
CategoryParser (class in  
src.osm\_configurator.model.parser.osm\_data\_parser), 50  
CutOutConfiguration (class in  
src.osm\_configurator.model.project.configuration.cut\_out\_configuration), 72  
CategoryParserInterface (class in  
src.osm\_configurator.model.parser.category\_parser\_interface), 49  
CutOutController (class in  
src.osm\_configurator.control.cut\_out\_controller), 26  
change\_state() (MainWindow method), 88  
change\_state() (StateManager method), 91  
check\_conflicts\_in\_category\_configuration() (CategoryController method), 8  
check\_conflicts\_in\_category\_configuration() (Control method), 12  
check\_conflicts\_in\_category\_configuration() (IControl method), 21  
check\_validity\_of\_calculation\_step() (CalculationParser method), 47  
check\_validity\_of\_calculation\_step() (CalculationParserInterface method), 48  
close\_program() (StateManager method), 91  
ConfigPhase (class in  
src.osm\_configurator.model.project.config\_phase\_enum), 82  
ConfigurationManager (class in  
src.osm\_configurator.model.project.configuration.configuration\_manager), 69  
Control (class in src.osm\_configurator.control.control), 9  
convert\_file() (OSMFileConverter method), 57  
create() (ActiveProject method), 76  
create\_boxplot() (ActiveProject method), 80  
create\_boxplot() (Application method), 37  
create\_boxplot() (DataVisualizer method), 83  
create\_boxplot() (IApplication method), 43  
create\_category() (ActiveProject method), 79  
create\_category() (Application method), 36  
create\_category() (CategoryController method), 9  
create\_category() (CategoryManager method), 68  
create\_category() (ConfigurationManager method),  
create\_category() (Control method), 13  
create\_category() (IApplication method), 42  
create\_category() (IControl method), 21  
create\_map() (ActiveProject method), 80  
create\_map() (Application method), 37  
create\_map() (DataVisualizer method), 82  
create\_map() (IApplication method), 43  
CREATE\_PROJECT (StateName attribute), 91  
create\_project() (Application method), 34  
create\_project() (Control method), 10  
create\_project() (IApplication method), 39  
create\_project() (IControl method), 18  
create\_project() (ProjectController method), 30  
createProjectFrame (class in  
src.osm\_configurator.view.toplevelframes.create\_project\_frame), 95  
CutOutConfiguration (class in  
src.osm\_configurator.model.project.configuration.cut\_out\_configuration), 72  
CutOutController (class in  
src.osm\_configurator.control.cut\_out\_controller), 26  
CutOutMode (class in src.osm\_configurator.model.project.configuration.cut\_out\_mode), 73  
CutOutParser (class in  
src.osm\_configurator.model.parser.cutOut\_parser), 49  
CutOutParserInterface (class in  
src.osm\_configurator.model.parser.cutOut\_parser\_interface), 50  
**D**  
DATA (StateName attribute), 91  
DATA\_CONFIG\_PHASE (ConfigPhase attribute), 82  
DataFrame (class in src.osm\_configurator.view.toplevelframes.data\_frame), 95  
DataVisualizationController (class in  
src.osm\_configurator.control.data\_visualization\_controller), 47  
DataVisualizer (class in  
src.osm\_configurator.model.project.data\_visualizer), 82  
deactivate() (Category method), 64  
default\_go\_left() (StateManager method), 91

- default\_go\_right() (*StateManager method*), 91
- DefaultValueEntry (class in *src.osm\_configurator.model.project.configuration*), 74
- delete\_category() (*CategoryController method*), 9
- delete\_category() (*Control method*), 13
- delete\_category() (*IControl method*), 21
- delete\_passive\_project() (*Control method*), 10
- delete\_passive\_project() (*IControl method*), 18
- delete\_passive\_project() (*ProjectController method*), 31
- download\_data() (*DownloadData method*), 75
- download\_data() (*OSMDataConfiguration method*), 75
- download\_osm\_data() (*IControl method*), 20
- download\_osm\_data() (*OSMDataController method*), 29
- DownloadData (class in *src.osm\_configurator.model.project.configuration*), 75
- ## E
- ENDED\_SUCCESSFULLY (*CalculationState attribute*), 56
- equals() (*State method*), 90
- ERROR\_INVALID\_CATEGORIES (*CalculationState attribute*), 56
- ERROR\_INVALID\_CUT\_OUT\_DATA (*CalculationState attribute*), 56
- ERROR\_INVALID\_OSM\_DATA (*CalculationState attribute*), 56
- ERROR\_INVALID\_PREVIOUS\_CALCULATIONS (*CalculationState attribute*), 56
- Export (class in *src.osm\_configurator.model.project.export*), 83
- export\_calculation() (*ActiveProject method*), 81
- export\_calculation() (*Application method*), 39
- export\_calculation() (*Export method*), 83
- export\_calculation() (*IApplication method*), 45
- export\_calculations() (*Control method*), 15
- export\_calculations() (*ExportController method*), 28
- export\_calculations() (*IControl method*), 24
- export\_configuration() (*ActiveProject method*), 81
- export\_configuration() (*Application method*), 38
- export\_configuration() (*Export method*), 83
- export\_configuration() (*IApplication method*), 45
- export\_configurations() (*Control method*), 16
- export\_configurations() (*ExportController method*), 28
- export\_configurations() (*IControl method*), 24
- export\_cut\_out\_map() (*ExportController method*), 28
- export\_cut\_out\_map() (*IControl method*), 24
- export\_map() (*ActiveProject method*), 81
- export\_map() (*Application method*), 39
- export\_map() (*Export method*), 84
- export\_map() (*IApplication method*), 45
- export\_project() (*ActiveProject method*), 81
- export\_project() (*Application method*), 38
- export\_project() (*Control method*), 15
- export\_project() (*Export method*), 83
- export\_project() (*ExportController method*), 28
- export\_project() (*IApplication method*), 44
- export\_project() (*IControl method*), 23
- ExportController (class in *src.osm\_configurator.control.export\_controller*), 28
- ExportFrame (class in *src.osm\_configurator.view.utilityframes.export\_frame*), 99
- ## F
- FIRST\_FLOOR\_AREA (*Attribute attribute*), 63
- ## G
- generate\_cut\_out\_map() (*Control method*), 17
- generate\_cut\_out\_map() (*DataVisualizationController method*), 27
- generate\_cut\_out\_map() (*IControl method*), 25
- GEO\_DATA\_PHASE (*CalculationPhase attribute*), 55
- GeoDataPhase (class in *src.osm\_configurator.model.project.calculation.geo\_data\_phase*), 57
- get\_active\_frames() (*State method*), 89
- get\_active\_project() (*ConfigurationManager method*), 72
- get\_aggregation\_methods() (*AggregationController method*), 5
- get\_aggregation\_methods() (*Control method*), 14
- get\_aggregation\_methods() (*IControl method*), 22
- get\_all\_aggregation\_methods() (*ActiveProject method*), 77
- get\_all\_aggregation\_methods() (*AggregationConfiguration method*), 60
- get\_all\_aggregation\_methods() (*Application method*), 35
- get\_all\_aggregation\_methods() (*ConfigurationManager method*), 70
- get\_all\_aggregation\_methods() (*IApplication method*), 41
- get\_attractivities\_of\_category() (*CategoryController method*), 9
- get\_attractivities\_of\_category() (*Control method*), 13
- get\_attractivities\_of\_category() (*IControl method*), 22
- get\_attractivity\_attribute\_list() (*AttractivityAttribute method*), 62
- get\_attractivity\_attribute\_name() (*AttractivityAttribute method*), 62

get\_attractivity\_attributes() (*Category method*), 66  
get\_attribute\_default() (*DefaultValueEntry method*), 74  
get\_base\_factor() (*AttractivityAttribute method*), 62  
get\_blacklist() (*Category method*), 64  
get\_bounding\_box() (*TrafficCell method*), 60  
get\_calculate\_area() (*Category method*), 65  
get\_calculate\_floor\_area() (*Category method*), 65  
get\_calculation\_method() (*CalculationMethodOfArea method*), 63  
get\_calculation\_method\_of\_area() (*Category method*), 65  
get\_calculation\_state() (*CalculationController method*), 7  
get\_calculation\_state() (*Control method*), 15  
get\_calculation\_state() (*IControl method*), 23  
get\_calculation\_visualization() (*Control method*), 17  
get\_calculation\_visualization() (*DataVisualizationController method*), 27  
get\_calculation\_visualization() (*IControl method*), 26  
get\_categories() (*ActiveProject method*), 79  
get\_categories() (*Application method*), 36  
get\_categories() (*CategoryManager method*), 68  
get\_categories() (*ConfigurationManager method*), 71  
get\_categories() (*IApplication method*), 42  
get\_category() (*ActiveProject method*), 78  
get\_category() (*Application method*), 36  
get\_category() (*CategoryManager method*), 68  
get\_category() (*ConfigurationManager method*), 71  
get\_category() (*IApplication method*), 42  
get\_category\_name() (*Category method*), 65  
get\_column() (*PositionedFrame method*), 89  
get\_current\_calculation\_phase() (*CalculationController method*), 7  
get\_current\_calculation\_phase() (*Control method*), 15  
get\_current\_calculation\_phase() (*IControl method*), 23  
get\_current\_calculation\_process() (*CalculationController method*), 7  
get\_current\_calculation\_process() (*Control method*), 15  
get\_current\_calculation\_process() (*IControl method*), 23  
get\_current\_config\_phase() (*Control method*), 11  
get\_current\_config\_phase() (*IControl method*), 19  
get\_current\_config\_phase() (*ProjectController method*), 31  
get\_cut\_out\_mode() (*ActiveProject method*), 78  
get\_cut\_out\_mode() (*Application method*), 35  
get\_cut\_out\_mode() (*ConfigurationManager method*), 70  
get\_cut\_out\_mode() (*Control method*), 12  
get\_cut\_out\_mode() (*CutOutConfiguration method*), 72  
get\_cut\_out\_mode() (*CutOutController method*), 26  
get\_cut\_out\_mode() (*IApplication method*), 41  
get\_cut\_out\_mode() (*IControl method*), 20  
get\_cut\_out\_path() (*ActiveProject method*), 78  
get\_cut\_out\_path() (*Application method*), 36  
get\_cut\_out\_path() (*ConfigurationManager method*), 71  
get\_cut\_out\_path() (*CutOutConfiguration method*), 73  
get\_cut\_out\_path() (*IApplication method*), 42  
get\_cut\_out\_reference() (*Control method*), 12  
get\_cut\_out\_reference() (*CutOutController method*), 27  
get\_cut\_out\_reference() (*IControl method*), 20  
get\_default\_left() (*State method*), 90  
get\_default\_location() (*ApplicationSettings method*), 45  
get\_default\_location() (*IApplication method*), 44  
get\_default\_right() (*State method*), 90  
get\_default\_value\_entry\_tag() (*DefaultValueEntry method*), 74  
get\_default\_value\_list() (*Category method*), 67  
get\_description() (*ActiveProject method*), 81  
get\_description() (*Application method*), 38  
get\_description() (*CalculationState method*), 56  
get\_description() (*IApplication method*), 44  
get\_description() (*PassiveProject method*), 46  
get\_description() (*ProjectSettings method*), 86  
get\_edit\_date() (*PassiveProject method*), 46  
get\_file\_extension() (*OSMFileFormat method*), 58  
get\_folder\_name\_for\_results() (*CalculationPhase method*), 55  
get\_frame() (*PositionedFrame method*), 89  
get\_is\_data\_downloaded() (*ConfigurationManager method*), 72  
get\_key\_recommendation() (*Application method*), 33  
get\_key\_recommendation() (*IApplication method*), 39  
get\_last\_step() (*ActiveProject method*), 76  
get\_line() (*PositionedFrame method*), 89  
get\_list\_of\_categories() (*CategoryController method*), 8  
get\_list\_of\_categories() (*Control method*), 13  
get\_list\_of\_categories() (*IControl method*), 21  
get\_list\_of\_key\_recommendations() (*CategoryController method*), 9  
get\_list\_of\_key\_recommendations() (*Control method*), 13

[get\\_list\\_of\\_key\\_recommendations\(\)](#) (*IControl method*), 22  
[get\\_list\\_of\\_passive\\_projects\(\)](#) (*Control method*), 10  
[get\\_list\\_of\\_passive\\_projects\(\)](#) (*IControl method*), 18  
[get\\_list\\_of\\_passive\\_projects\(\)](#) (*ProjectController method*), 30  
[get\\_location\(\)](#) (*ActiveProject method*), 80  
[get\\_location\(\)](#) (*Application method*), 37  
[get\\_location\(\)](#) (*IApplication method*), 43  
[get\\_location\(\)](#) (*ProjectSettings method*), 85  
[get\\_name\(\)](#) (*ActiveProject method*), 80  
[get\\_name\(\)](#) (*AggregationMethod method*), 52  
[get\\_name\(\)](#) (*Application method*), 38  
[get\\_name\(\)](#) (*Attribute method*), 63  
[get\\_name\(\)](#) (*CalculationPhase method*), 55  
[get\\_name\(\)](#) (*CalculationState method*), 56  
[get\\_name\(\)](#) (*ConfigPhase method*), 82  
[get\\_name\(\)](#) (*CutOutMode method*), 73  
[get\\_name\(\)](#) (*IApplication method*), 44  
[get\\_name\(\)](#) (*PassiveProject method*), 46  
[get\\_name\(\)](#) (*ProjectSettings method*), 86  
[get\\_name\(\)](#) (*TrafficCell method*), 60  
[get\\_order\(\)](#) (*CalculationPhase method*), 55  
[get\\_osm\\_data\(\)](#) (*ActiveProject method*), 77  
[get\\_osm\\_data\(\)](#) (*Application method*), 34  
[get\\_osm\\_data\(\)](#) (*ConfigurationManager method*), 69  
[get\\_osm\\_data\(\)](#) (*IApplication method*), 40  
[get\\_osm\\_data\(\)](#) (*OSMDataConfiguration method*), 75  
[get\\_osm\\_data\\_reference\(\)](#) (*Control method*), 11  
[get\\_osm\\_data\\_reference\(\)](#) (*IControl method*), 19  
[get\\_osm\\_data\\_reference\(\)](#) (*OSMDataController method*), 29  
[get\\_passive\\_project\\_list\(\)](#) (*Application method*), 33  
[get\\_passive\\_project\\_list\(\)](#) (*IApplication method*), 39  
[get\\_project\\_default\\_folder\(\)](#) (*Control method*), 17  
[get\\_project\\_default\\_folder\(\)](#) (*IControl method*), 25  
[get\\_project\\_default\\_folder\(\)](#) (*SettingsController method*), 32  
[get\\_project\\_description\(\)](#) (*Control method*), 16  
[get\\_project\\_description\(\)](#) (*IControl method*), 25  
[get\\_project\\_description\(\)](#) (*SettingsController method*), 32  
[get\\_project\\_folder\\_path\(\)](#) (*PassiveProject method*), 46  
[get\\_project\\_name\(\)](#) (*Control method*), 16  
[get\\_project\\_name\(\)](#) (*IControl method*), 24  
[get\\_project\\_name\(\)](#) (*SettingsController method*), 32  
[get\\_project\\_path\(\)](#) (*ActiveProject method*), 77  
[get\\_state\(\)](#) (*StateManager method*), 91  
[get\\_state\\_name\(\)](#) (*State method*), 90  
[get\\_strictly\\_use\\_default\\_values\(\)](#) (*Category method*), 66  
[get\\_tag\\_list\(\)](#) (*TagListPriorityFrame method*), 102  
[get\\_text\\_list\(\)](#) (*TagListFrame method*), 101  
[get\\_whitelist\(\)](#) (*Category method*), 64

**I**

[IApplication](#) (class in *src.osm\_configurator.model.application.application\_interface*), 39  
[ICalculationPhase](#) (class in *src.osm\_configurator.model.project.calculation.calculation\_phase*), 55  
[IControl](#) (class in *src.osm\_configurator.control.control\_interface*), 18  
[import\\_category\\_configuration\(\)](#) (*CategoryController method*), 8  
[import\\_category\\_configuration\(\)](#) (*Control method*), 13  
[import\\_category\\_configuration\(\)](#) (*IControl method*), 21  
[is\\_active\(\)](#) (*Category method*), 64  
[is\\_aggregation\\_method\\_active\(\)](#) (*ActiveProject method*), 77  
[is\\_aggregation\\_method\\_active\(\)](#) (*Aggregation-Configuration method*), 61  
[is\\_aggregation\\_method\\_active\(\)](#) (*Aggregation-Controller method*), 5  
[is\\_aggregation\\_method\\_active\(\)](#) (*Application method*), 35  
[is\\_aggregation\\_method\\_active\(\)](#) (*Configuration-Manager method*), 70  
[is\\_aggregation\\_method\\_active\(\)](#) (*Control method*), 14  
[is\\_aggregation\\_method\\_active\(\)](#) (*IApplication method*), 41  
[is\\_aggregation\\_method\\_active\(\)](#) (*IControl method*), 22  
[is\\_project\\_loaded\(\)](#) (*Control method*), 11  
[is\\_project\\_loaded\(\)](#) (*IControl method*), 19  
[is\\_project\\_loaded\(\)](#) (*ProjectController method*), 31

**L**

[load\\_project\(\)](#) (*Application method*), 34  
[load\\_project\(\)](#) (*Control method*), 10  
[load\\_project\(\)](#) (*IApplication method*), 40  
[load\\_project\(\)](#) (*IControl method*), 18  
[load\\_project\(\)](#) (*ProjectController method*), 30  
[load\\_project\(\)](#) (*ProjectIOHandler method*), 84  
[LOWER\\_QUARTILE](#) (*AggregationMethod attribute*), 52



## M

main() (*ApplicationController* method), 6  
 MAIN\_MENU (*StateName* attribute), 91  
 MainMenuFrame (class in *src.osm\_configurator.view.toplevelframes.main\_menu\_frame*), 96  
 MainWindow (class in *src.osm\_configurator.view.states.main\_window*), 88  
 MAXIMUM (*AggregationMethod* attribute), 52  
 MEAN (*AggregationMethod* attribute), 51  
 merge\_categories() (*ActiveProject* method), 79  
 merge\_categories() (*Application* method), 37  
 merge\_categories() (*CategoryManager* method), 69  
 merge\_categories() (*ConfigurationManager* method), 72  
 merge\_categories() (*IApplication* method), 43  
 MINIMUM (*AggregationMethod* attribute), 52  
 module  
     src, 102  
     src.osm\_configurator, 102  
     src.osm\_configurator.control, 33  
     src.osm\_configurator.control.aggregation\_controller, 5  
     src.osm\_configurator.control.application\_controller, 6  
     src.osm\_configurator.control.calculation\_controller, 7  
     src.osm\_configurator.control.category\_controller, 8  
     src.osm\_configurator.control.control, 9  
     src.osm\_configurator.control.control\_interface, 18  
     src.osm\_configurator.control.cut\_out\_controller, 26  
     src.osm\_configurator.control.data\_visualization\_controller, 27  
     src.osm\_configurator.control.export\_controller, 28  
     src.osm\_configurator.control.osm\_data\_controller, 29  
     src.osm\_configurator.control.project\_controller, 30  
     src.osm\_configurator.control.settings\_controller, 32  
     src.osm\_configurator.model, 87  
     src.osm\_configurator.model.application, 47  
     src.osm\_configurator.model.application.application, 33  
     src.osm\_configurator.model.application.application\_interface, 39  
     src.osm\_configurator.model.application.application\_settings, 45  
     src.osm\_configurator.model.application.passive\_project, 46  
     src.osm\_configurator.model.application.recommender\_system, 47  
     src.osm\_configurator.model.parser, 51  
     src.osm\_configurator.model.parser.calculation\_parser, 47  
     src.osm\_configurator.model.parser.calculation\_parser\_interface, 48  
     src.osm\_configurator.model.parser.category\_parser, 48  
     src.osm\_configurator.model.parser.category\_parser\_interface, 49  
     src.osm\_configurator.model.parser.cutOut\_parser, 49  
     src.osm\_configurator.model.parser.cutOut\_parser\_interface, 50  
     src.osm\_configurator.model.parser.osm\_data\_parser, 50  
     src.osm\_configurator.model.parser.osm\_data\_parser\_interface, 51  
     src.osm\_configurator.model.project, 87  
     src.osm\_configurator.model.project.active\_project, 76  
     src.osm\_configurator.model.project.calculation, 60  
     src.osm\_configurator.model.project.calculation.aggregation, 51  
     src.osm\_configurator.model.project.calculation.aggregation\_interface, 52  
     src.osm\_configurator.model.project.calculation.attraction, 53  
     src.osm\_configurator.model.project.calculation.building, 53  
     src.osm\_configurator.model.project.calculation.calculation, 54  
     src.osm\_configurator.model.project.calculation.calculation\_interface, 54  
     src.osm\_configurator.model.project.calculation.calculation\_settings, 55  
     src.osm\_configurator.model.project.calculation.calculation\_settings\_interface, 56  
     src.osm\_configurator.model.project.calculation.geo\_data, 57  
     src.osm\_configurator.model.project.calculation.osm\_file, 57  
     src.osm\_configurator.model.project.calculation.osm\_file\_interface, 58  
     src.osm\_configurator.model.project.calculation.reduction, 58  
     src.osm\_configurator.model.project.calculation.split\_up, 59  
     src.osm\_configurator.model.project.calculation.tag\_file, 59

```

src.osm_configurator.model.project.calculation.traffic_cell,
60 src.osm_configurator.view.states.state_name_enum,
src.osm_configurator.model.project.config_phase_enum,
82 src.osm_configurator.view.toplevelframes,
src.osm_configurator.model.project.configuration, 99
76 src.osm_configurator.view.toplevelframes.aggregation_f
src.osm_configurator.model.project.configuration.aggregation_configuration,
60 src.osm_configurator.view.toplevelframes.attractivity_
src.osm_configurator.model.project.configuration.attractivity_attribute,
61 src.osm_configurator.view.toplevelframes.attractivity_
src.osm_configurator.model.project.configuration.attribute_enum,
63 src.osm_configurator.view.toplevelframes.calculation_f
src.osm_configurator.model.project.configuration.calculation_method_of_area_enum,
63 src.osm_configurator.view.toplevelframes.category_fram
src.osm_configurator.model.project.configuration.category,
64 src.osm_configurator.view.toplevelframes.create_projec
src.osm_configurator.model.project.configuration.category_manager,
68 src.osm_configurator.view.toplevelframes.data_frame,
src.osm_configurator.model.project.configuration.configuration_manager,
69 src.osm_configurator.view.toplevelframes.main_menu_fra
src.osm_configurator.model.project.configuration.cut_out_configuration,
72 src.osm_configurator.view.toplevelframes.project_foot
src.osm_configurator.model.project.configuration.cut_out_mode_enum,
73 src.osm_configurator.view.toplevelframes.project_head
src.osm_configurator.model.project.configuration.default_value_entry,
74 src.osm_configurator.view.toplevelframes.reduction_fra
src.osm_configurator.model.project.configuration.download_data,
75 src.osm_configurator.view.toplevelframes.settings_fram
src.osm_configurator.model.project.configuration.osm_data_configuration,
75 src.osm_configurator.view.toplevelframes.top_level_fra
src.osm_configurator.model.project.data_visualizer, 98
82 src.osm_configurator.view.utilityframes,
src.osm_configurator.model.project.export, 102
83 src.osm_configurator.view.utilityframes.export_frame,
src.osm_configurator.model.project.project_io_handler,
84 src.osm_configurator.view.utilityframes.reduction_calc
src.osm_configurator.model.project.project_saver, 99
85 src.osm_configurator.view.utilityframes.reduction_defa
src.osm_configurator.model.project.project_settings, 100
85 src.osm_configurator.view.utilityframes.settings_appli
src.osm_configurator.view, 102
100
src.osm_configurator.view.popups, 88
src.osm_configurator.view.popups.alert_pop_up, 101
87 src.osm_configurator.view.utilityframes.tag_list_frame
src.osm_configurator.view.popups.yes_no_pop_up, 101
87 src.osm_configurator.view.utilityframes.tag_list_prior
src.osm_configurator.view.states, 92
102
src.osm_configurator.view.states.main_window.move_default_value_entry_down() (Category
88 method), 67
src.osm_configurator.view.states.positioned_frame.move_default_value_entry_up() (Category
88 method), 67
src.osm_configurator.view.states.state,
89
src.osm_configurator.view.states.state_manager, 54

```

## N

NOT\_STARTED\_YET (*CalculationState* attribute), 56  
 NUMER\_OF\_FLOOR (*Attribute* attribute), 63

## O

OSM (*OSMFileFormat* attribute), 58  
 OSMDDataConfiguration (class in *src.osm\_configurator.model.project.configuration.osm\_data\_configuration*), 75  
 OSMDDataController (class in *src.osm\_configurator.control.osm\_data\_controller*), 29  
 OSMDDataParserInterface (class in *src.osm\_configurator.model.parser.osm\_data\_parser\_interface*), 51  
 OSMFileConverter (class in *src.osm\_configurator.model.project.calculation.osm\_file\_converter*), 57  
 OSMFileFormat (class in *src.osm\_configurator.model.project.calculation.osm\_file\_format*), 58  
 override\_categories() (*ActiveProject* method), 79  
 override\_categories() (*Application* method), 37  
 override\_categories() (*CategoryManager* method), 69  
 override\_categories() (*ConfigurationManager* method), 72  
 override\_categories() (*IApplication* method), 43

## P

parse\_category\_file() (*CategoryParser* method), 48  
 parse\_category\_file() (*CategoryParserInterface* method), 49  
 parse\_cutout\_file() (*CutOutParser* method), 49  
 parse\_cutout\_file() (*CutOutParserInterface* method), 50  
 parse\_osm\_data\_file() (*CategoryParser* method), 50  
 parse\_osm\_data\_file() (*OSMDDataParserInterface* method), 51  
 PassiveProject (class in *src.osm\_configurator.model.application.passive\_project*), 46  
 PBF (*OSMFileFormat* attribute), 58  
 PositionedFrame (class in *src.osm\_configurator.view.states.positioned\_frame*), 88  
 ProjectController (class in *src.osm\_configurator.control.project\_controller*), 30  
 ProjectFootFrame (class in *src.osm\_configurator.view.toplevelframes.project\_foot\_frame*), 96  
 ProjectHeadFrame (class in *src.osm\_configurator.view.toplevelframes.project\_head\_frame*), 97  
 ProjectIOHandler (class in *src.osm\_configurator.model.project.project\_io\_handler*), 84  
 ProjectSaver (class in *src.osm\_configurator.model.project.project\_saver*), 85  
 ProjectSettings (class in *src.osm\_configurator.model.project.project\_settings*), 85  
 PROPERTY\_AREA (*Attribute* attribute), 63

## R

recommend() (*RecommenderSystem* method), 47  
 RecommenderSystem (class in *src.osm\_configurator.model.application.recommender\_system*), 47  
 REDUCTION (*StateName* attribute), 92  
 REDUCTION\_CONFIG\_PHASE (*ConfigPhase* attribute), 82  
 REDUCTION\_PHASE (*CalculationPhase* attribute), 55  
 ReductionCalculationFrame (class in *src.osm\_configurator.view.utilityframes.reduction\_calculation\_frame*), 99  
 ReductionDefaultValueFrame (class in *src.osm\_configurator.view.utilityframes.reduction\_default\_value\_frame*), 100  
 ReductionFrame (class in *src.osm\_configurator.view.toplevelframes.reduction\_frame*), 97  
 ReductionPhase (class in *src.osm\_configurator.model.project.calculation.reduction\_phase*), 58  
 remove\_attractivity\_attribute() (*Category* method), 66  
 remove\_buildings\_on\_edge() (*BuildingOnEdgeManager* method), 53  
 remove\_category() (*ActiveProject* method), 79  
 remove\_category() (*Application* method), 36  
 remove\_category() (*CategoryManager* method), 68  
 remove\_category() (*ConfigurationManager* method), 71  
 remove\_category() (*IApplication* method), 42  
 remove\_default\_value\_entry() (*Category* method), 67  
 RUNNING (*CalculationState* attribute), 56

## S

save\_project() (*Control* method), 11  
 save\_project() (*IControl* method), 19  
 save\_project() (*ProjectController* method), 31  
 save\_project() (*ProjectSaver* method), 85  
 set\_aggregation\_method\_active() (*ActiveProject* method), 77  
 set\_aggregation\_method\_active() (*Aggregation-Configuration* method), 61

set\_aggregation\_method\_active() (Aggregation-Controller method), 6  
 set\_aggregation\_method\_active() (Application method), 35  
 set\_aggregation\_method\_active() (Configuration-Manager method), 70  
 set\_aggregation\_method\_active() (Control method), 14  
 set\_aggregation\_method\_active() (IApplication method), 41  
 set\_aggregation\_method\_active() (IControl method), 22  
 set\_attractivity\_attribute\_list() (Attractivity-Attribute method), 62  
 set\_attractivity\_attribute\_name() (Attractivity-Attribute method), 62  
 set\_attribute\_default() (DefaultValueEntry method), 74  
 set\_base\_factor() (AttractivityAttribute method), 62  
 set\_blacklist() (Category method), 65  
 set\_calculate\_floor\_area() (Category method), 66  
 set\_calculation\_method\_of\_area() (Category method), 65  
 set\_category\_name() (Category method), 65  
 set\_current\_config\_phase() (Control method), 11  
 set\_current\_config\_phase() (IControl method), 19  
 set\_current\_config\_phase() (ProjectController method), 31  
 set\_cut\_out\_mode() (ActiveProject method), 78  
 set\_cut\_out\_mode() (Application method), 35  
 set\_cut\_out\_mode() (ConfigurationManager method), 70  
 set\_cut\_out\_mode() (Control method), 12  
 set\_cut\_out\_mode() (CutOutConfiguration method), 73  
 set\_cut\_out\_mode() (CutOutController method), 26  
 set\_cut\_out\_mode() (IApplication method), 41  
 set\_cut\_out\_mode() (IControl method), 20  
 set\_cut\_out\_path() (ActiveProject method), 78  
 set\_cut\_out\_path() (Application method), 36  
 set\_cut\_out\_path() (ConfigurationManager method), 71  
 set\_cut\_out\_path() (CutOutConfiguration method), 73  
 set\_cut\_out\_path() (IApplication method), 42  
 set\_cut\_out\_reference() (Control method), 12  
 set\_cut\_out\_reference() (CutOutController method), 26  
 set\_cut\_out\_reference() (IControl method), 20  
 set\_default\_location() (ApplicationSettings method), 46  
 set\_default\_location() (IApplication method), 44  
 set\_description() (ActiveProject method), 80  
 set\_description() (Application method), 38  
 set\_description() (IApplication method), 44  
 set\_description() (ProjectSettings method), 86  
 set\_location() (ProjectSettings method), 85  
 set\_name() (ActiveProject method), 80  
 set\_name() (Application method), 38  
 set\_name() (IApplication method), 44  
 set\_name() (ProjectSettings method), 86  
 set\_osm\_data() (ActiveProject method), 77  
 set\_osm\_data() (Application method), 34  
 set\_osm\_data() (ConfigurationManager method), 69  
 set\_osm\_data() (IApplication method), 40  
 set\_osm\_data() (OSMDataConfiguration method), 75  
 set\_osm\_data\_reference() (Control method), 11  
 set\_osm\_data\_reference() (IControl method), 19  
 set\_osm\_data\_reference() (OSMDataController method), 29  
 set\_project\_default\_folder() (Control method), 17  
 set\_project\_default\_folder() (IControl method), 25  
 set\_project\_default\_folder() (SettingsController method), 33  
 set\_project\_description() (Control method), 16  
 set\_project\_description() (IControl method), 25  
 set\_project\_description() (SettingsController method), 32  
 set\_project\_name() (Control method), 16  
 set\_project\_name() (IControl method), 25  
 set\_project\_name() (SettingsController method), 32  
 set\_strictly\_use\_default\_values() (Category method), 66  
 set\_tag() (DefaultValueEntry method), 74  
 set\_tag\_list() (TagListPriorityFrame method), 102  
 set\_text\_list() (TagListFrame method), 101  
 set\_whitelist() (Category method), 64  
 SETTINGS (StateName attribute), 92  
 SettingsApplicationFrame (class in src.osm\_configurator.view.utilityframes.settings\_application\_frame), 100  
 SettingsController (class in src.osm\_configurator.control.settings\_controller), 32  
 SettingsFrame (class in src.osm\_configurator.view.toplevelframes.settings\_frame), 98  
 SettingsProjectFrame (class in src.osm\_configurator.view.utilityframes.settings\_project\_frame), 101  
 split\_up\_files() (SplitUpFile method), 59  
 SplitUpFile (class in src.osm\_configurator.model.project.calculation.split\_up\_files), 59  
 src module, 102



src.osm_configurator	src.osm_configurator.model.parser.cutOut_parser_interface
module, 102	module, 50
src.osm_configurator.control	src.osm_configurator.model.parser.osm_data_parser
module, 33	module, 50
src.osm_configurator.control.aggregation_controller	src.osm_configurator.model.parser.osm_data_parser_interface
module, 5	module, 51
src.osm_configurator.control.application_controller	src.osm_configurator.model.project
module, 6	module, 87
src.osm_configurator.control.calculation_controller	src.osm_configurator.model.project.active_project
module, 7	module, 76
src.osm_configurator.control.category_controller	src.osm_configurator.model.project.calculation
module, 8	module, 60
src.osm_configurator.control.control	src.osm_configurator.model.project.calculation.aggregation
module, 9	module, 51
src.osm_configurator.control.control_interfaces	src.osm_configurator.model.project.calculation.aggregation
module, 18	module, 52
src.osm_configurator.control.cut_out_controller	src.osm_configurator.model.project.calculation.attractiv
module, 26	module, 53
src.osm_configurator.control.data_visualization	src.osm_configurator.model.project.calculation.building_or
module, 27	module, 53
src.osm_configurator.control.export_controllers	src.osm_configurator.model.project.calculation.calculation
module, 28	module, 54
src.osm_configurator.control.osm_data_controller	src.osm_configurator.model.project.calculation.calculation
module, 29	module, 54
src.osm_configurator.control.project_controller	src.osm_configurator.model.project.calculation.calculation
module, 30	module, 55
src.osm_configurator.control.settings_controller	src.osm_configurator.model.project.calculation.calculation
module, 32	module, 56
src.osm_configurator.model	src.osm_configurator.model.project.calculation.geo_data_ph
module, 87	module, 57
src.osm_configurator.model.application	src.osm_configurator.model.project.calculation.osm_file_co
module, 47	module, 57
src.osm_configurator.model.application.application	src.osm_configurator.model.project.calculation.osm_file_fo
module, 33	module, 58
src.osm_configurator.model.application.application_interface	src.osm_configurator.model.project.calculation.reduction_p
module, 39	module, 58
src.osm_configurator.model.application.application_settings	src.osm_configurator.model.project.calculation.split_up_fi
module, 45	module, 59
src.osm_configurator.model.application.passive_project	src.osm_configurator.model.project.calculation.tag_filter
module, 46	module, 59
src.osm_configurator.model.application.recommender_system	src.osm_configurator.model.project.calculation.traffic_cel
module, 47	module, 60
src.osm_configurator.model.parser	src.osm_configurator.model.project.config_phase_enum
module, 51	module, 82
src.osm_configurator.model.parser.calculation_parser	src.osm_configurator.model.project.configuration
module, 47	module, 76
src.osm_configurator.model.parser.calculation_parser_interface	src.osm_configurator.model.project.configuration.aggregati
module, 48	module, 60
src.osm_configurator.model.parser.category_parser	src.osm_configurator.model.project.configuration.attractiv
module, 48	module, 61
src.osm_configurator.model.parser.category_parser_interface	src.osm_configurator.model.project.configuration.attribute
module, 49	module, 63
src.osm_configurator.model.parser.cutOut_parser	src.osm_configurator.model.project.configuration.calculati
module, 49	module, 63

src.osm\_configurator.model.project.configuration.module, 64

src.osm\_configurator.model.project.configuration.module, 68

src.osm\_configurator.model.project.configuration.module, 69

src.osm\_configurator.model.project.configuration.module, 72

src.osm\_configurator.model.project.configuration.module, 73

src.osm\_configurator.model.project.configuration.module, 74

src.osm\_configurator.model.project.configuration.module, 75

src.osm\_configurator.model.project.configuration.module, 75

src.osm\_configurator.model.project.data\_visualization.module, 82

src.osm\_configurator.model.project.export.module, 83

src.osm\_configurator.model.project.project\_io\_handler.module, 84

src.osm\_configurator.model.project.project\_saver.module, 85

src.osm\_configurator.model.project.project\_settings.module, 85

src.osm\_configurator.view.module, 102

src.osm\_configurator.view.popups.module, 88

src.osm\_configurator.view.popups.alert\_pop\_up.module, 87

src.osm\_configurator.view.popups.yes\_no\_pop\_up.module, 87

src.osm\_configurator.view.states.module, 92

src.osm\_configurator.view.states.main\_window.module, 88

src.osm\_configurator.view.states.positioned\_frame.module, 88

src.osm\_configurator.view.states.state.module, 89

src.osm\_configurator.view.states.state\_manager.module, 90

src.osm\_configurator.view.states.state\_name\_enum.module, 91

src.osm\_configurator.view.toplevelframes.module, 99

src.osm\_configurator.view.toplevelframes.aggregation\_frame.module, 92

src.osm\_configurator.view.toplevelframes.attraction\_frame.module, 93

src.osm\_configurator.view.toplevelframes.attraction\_frame.module, 93

src.osm\_configurator.view.toplevelframes.calculation\_frame.module, 94

src.osm\_configurator.view.toplevelframes.category\_frame.module, 94

src.osm\_configurator.view.toplevelframes.create\_project\_frame.module, 95

src.osm\_configurator.view.toplevelframes.data\_frame.module, 95

src.osm\_configurator.view.toplevelframes.main\_menu\_frame.module, 96

src.osm\_configurator.view.toplevelframes.project\_foot\_frame.module, 96

src.osm\_configurator.view.toplevelframes.project\_head\_frame.module, 97

src.osm\_configurator.view.toplevelframes.reduction\_frame.module, 97

src.osm\_configurator.view.toplevelframes.settings\_frame.module, 98

src.osm\_configurator.view.toplevelframes.top\_level\_frame.module, 98

src.osm\_configurator.view.utilityframes.module, 102

src.osm\_configurator.view.utilityframes.export\_frame.module, 99

src.osm\_configurator.view.utilityframes.reduction\_calculation\_frame.module, 99

src.osm\_configurator.view.utilityframes.reduction\_default\_frame.module, 100

src.osm\_configurator.view.utilityframes.settings\_application\_frame.module, 100

src.osm\_configurator.view.utilityframes.settings\_project\_frame.module, 101

src.osm\_configurator.view.utilityframes.tag\_list\_frame.module, 101

src.osm\_configurator.view.utilityframes.tag\_list\_priority\_frame.module, 102

start\_calculation() (*ActiveProject method*), 76

start\_calculation() (*Application method*), 34

start\_calculation() (*CalculationManager method*), 54

start\_calculation() (*IApplication method*), 40

start\_calculations() (*CalculationController method*), 7

start\_calculations() (*Control method*), 14

start\_calculations() (*IControl method*), 23

State (class in src.osm\_configurator.view.states.state), 89

StateManager (class in src.osm\_configurator.view.states.state\_manager), 90

StateNameEnum (class in src.osm\_configurator.view.states.state\_name\_enum), 91

STM (Attribute of src.osm\_configurator.view.toplevelframes.attraction\_frame), 51

## T

`TAG_FILTER_PHASE` (*CalculationPhase* attribute), 55

`TagFilterPhase` (class in *src.osm\_configurator.model.project.calculation.tag\_filter\_phase*), 59

`TagListFrame` (class in *src.osm\_configurator.view.utilityframes.tag\_list\_frame*), 101

`TagListPriorityFrame` (class in *src.osm\_configurator.view.utilityframes.tag\_list\_priority\_frame*), 102

`TopLevelFrame` (class in *src.osm\_configurator.view.toplevelframes.top\_level\_frame*), 98

`TrafficCell` (class in *src.osm\_configurator.model.project.calculation.traffic\_cell*), 60

## U

`UPPER_QUARTILE` (*AggregationMethod* attribute), 51

## Y

`YesNoPopUp` (class in *src.osm\_configurator.view.popups.yes\_no\_pop\_up*), 87