

MiniCity

Generated by Doxygen 1.10.0

1 Hierarchical Index	1
1.1 Class Hierarchy	1
2 Class Index	3
2.1 Class List	3
3 File Index	5
3.1 File List	5
4 Class Documentation	7
4.1 Building Class Reference	7
4.2 City Class Reference	8
4.2.1 Detailed Description	9
4.2.2 Member Function Documentation	9
4.2.2.1 add_max_population()	9
4.2.2.2 add_population()	9
4.2.2.3 get_balance()	10
4.2.2.4 get_dimensions()	10
4.2.2.5 get_layout()	10
4.2.2.6 get_max_population()	10
4.2.2.7 get_name()	10
4.2.2.8 get_population()	11
4.2.2.9 get_year()	11
4.2.2.10 set_balance()	11
4.2.2.11 set_dimensions()	11
4.2.2.12 set_layout()	11
4.2.2.13 set_name()	12
4.2.2.14 set_population()	12
4.3 Dimensions Struct Reference	12
4.3.1 Detailed Description	13
4.4 Factory Class Reference	13
4.5 Hospital Class Reference	14
4.6 House Class Reference	15
4.7 Industrial_zone Class Reference	15
4.7.1 Member Function Documentation	16
4.7.1.1 generate_characteristic_value()	16
4.8 Infrastructure Class Reference	17
4.9 Menu Class Reference	17
4.10 Population_determinant Class Reference	18
4.11 Population_determinant_context Class Reference	18
4.12 Population_determinant_tier_1 Class Reference	19
4.12.1 Member Function Documentation	19
4.12.1.1 find_interest()	19

4.13 Population_determinant_tier_2 Class Reference	19
4.13.1 Member Function Documentation	20
4.13.1.1 find_interest()	20
4.14 Population_determinant_tier_3 Class Reference	20
4.14.1 Member Function Documentation	20
4.14.1.1 find_interest()	20
4.15 Random Class Reference	21
4.16 Residential_zone Class Reference	21
4.16.1 Member Function Documentation	22
4.16.1.1 generate_characteristic_value()	22
4.17 Road Class Reference	22
4.18 School Class Reference	23
4.19 Special_element Class Reference	24
4.20 View Class Reference	24
4.20.1 Detailed Description	26
4.20.2 Member Function Documentation	26
4.20.2.1 build_factory()	26
4.20.2.2 build_hospital()	26
4.20.2.3 build_house()	26
4.20.2.4 build_school()	26
4.20.2.5 build_zone()	26
4.20.2.6 loop_buildings()	27
4.20.2.7 loop_special_buildings()	27
4.20.2.8 loop_zones()	27
4.20.2.9 show_menu()	27
4.20.2.10 switch_building()	27
4.20.2.11 switch_infrastructure()	28
4.20.2.12 switch_main()	28
4.20.2.13 switch_roads()	28
4.20.2.14 switch_special_building()	29
4.20.2.15 switch_zone()	29
4.21 Zone< T > Class Template Reference	29
5 File Documentation	31
5.1 building.h	31
5.2 city.h	31
5.3 factory.h	32
5.4 hospital.h	32
5.5 house.h	32
5.6 industrial_zone.h	32
5.7 infrastructure.h	33
5.8 main_menu.h	33

5.9 menu.h	33
5.10 population_determinant.h	34
5.11 population_determinant_context.h	34
5.12 population_determinant_tier_1.h	34
5.13 population_determinant_tier_2.h	35
5.14 population_determinant_tier_3.h	35
5.15 population_determinant_tiers.h	35
5.16 random.h	35
5.17 residential_zone.h	35
5.18 road.h	36
5.19 school.h	36
5.20 special_element.h	36
5.21 view.h	36
5.22 zone.h	37
Index	39

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

City	8
Dimensions	12
Infrastructure	17
Zone< double >	29
Industrial_zone	15
Zone< int >	29
Residential_zone	21
Building	7
Factory	13
House	15
Hospital	14
Road	22
School	23
Zone< T >	29
Menu	17
Population_determinant	18
Population_determinant_tier_1	19
Population_determinant_tier_2	19
Population_determinant_tier_3	20
Population_determinant_context	18
Random	21
Special_element	24
Hospital	14
School	23
View	24

Chapter 2

Class Index

2.1 Class List

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

Building	7
City	
Represents a city in the simulation	8
Dimensions	
Represents the dimensions of the city grid	12
Factory	13
Hospital	14
House	15
Industrial_zone	15
Infrastructure	17
Menu	17
Population_determinant	18
Population_determinant_context	18
Population_determinant_tier_1	19
Population_determinant_tier_2	19
Population_determinant_tier_3	20
Random	21
Residential_zone	21
Road	22
School	23
Special_element	24
View	
Manages the user interface and interaction for managing city infrastructure	24
Zone< T >	29

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

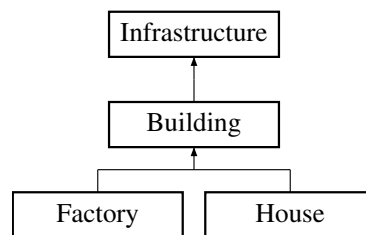
C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/building.h	31
C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/city.h	31
C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/factory.h	32
C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/hospital.h	32
C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/house.h	32
C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/industrial_zone.h	32
C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/infrastructure.h	33
C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/main_menu.h	33
C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/menu.h	33
C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/population_determinant.h . . .	34
C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/population_determinant_context.h	
34	
C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/population_determinant_tier_1.h	
34	
C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/population_determinant_tier_2.h	
35	
C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/population_determinant_tier_3.h	
35	
C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/population_determinant_tiers.h	
35	
C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/random.h	35
C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/residential_zone.h	35
C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/road.h	36
C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/school.h	36
C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/special_element.h	36
C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/view.h	36
C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/zone.h	37

Chapter 4

Class Documentation

4.1 Building Class Reference

Inheritance diagram for Building:



Public Member Functions

- **Building** (char s)

Public Member Functions inherited from [Infrastructure](#)

- char **get_symbol** () const

Additional Inherited Members

Static Public Member Functions inherited from [Infrastructure](#)

- static std::map< char, int > **get_number_of_elements_map** ()
- static int **get_number_of** (char s)

Protected Member Functions inherited from [Infrastructure](#)

- **Infrastructure** (char s)

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

- C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/building.h

4.2 City Class Reference

Represents a city in the simulation.

```
#include <city.h>
```

Public Member Functions

- **City ()**
Constructs a new [City](#) object with default values.
- void [set_name](#) (std::string s)
Sets the name of the city.
- std::string [get_name](#) ()
Gets the name of the city.
- void [set_balance](#) (int i)
Sets the balance of the city.
- int [get_balance](#) ()
Gets the balance of the city.
- void [set_population](#) (int i)
Sets the population of the city.
- void [add_population](#) (int i)
Adds to the current population of the city.
- int [get_population](#) ()
Gets the current population of the city.
- void [add_max_population](#) (int i)
Adds to the maximum population capacity of the city.
- int [get_max_population](#) ()
Gets the maximum population capacity of the city.
- int [get_year](#) ()
Gets the current year in the city simulation.
- void **increment_year** ()
Increments the current year by one.
- void [set_dimensions](#) ([Dimensions](#) d)
Sets the dimensions of the city grid.
- [Dimensions](#) [get_dimensions](#) ()
Gets the dimensions of the city grid.
- Grid & [set_layout](#) ()
Sets the layout of the city.
- const Grid & [get_layout](#) ()
Gets the layout of the city.

Private Attributes

- `std::string name = "MiniCity"`
The name of the city.
- `int balance = 20000`
The financial balance of the city.
- `int population = 0`
The current population of the city.
- `int max_population = 0`
The maximum population the city can support.
- `int year = 2024`
The current year in the city simulation.
- `Dimensions dimensions = { 10, 10 }`
The dimensions of the city grid.
- `Grid layout`
The layout of the city, represented as a grid of infrastructures.

4.2.1 Detailed Description

Represents a city in the simulation.

4.2.2 Member Function Documentation

4.2.2.1 add_max_population()

```
void City::add_max_population (
    int i )
```

Adds to the maximum population capacity of the city.

Parameters

<i>i</i>	The amount to add to the maximum population.
----------	--

4.2.2.2 add_population()

```
void City::add_population (
    int i )
```

Adds to the current population of the city.

Parameters

<i>i</i>	The amount to add to the population.
----------	--------------------------------------

4.2.2.3 get_balance()

```
int City::get_balance ( )
```

Gets the balance of the city.

Returns

The balance of the city.

4.2.2.4 get_dimensions()

```
Dimensions City::get_dimensions ( )
```

Gets the dimensions of the city grid.

Returns

The dimensions of the city grid.

4.2.2.5 get_layout()

```
const Grid & City::get_layout ( )
```

Gets the layout of the city.

Returns

A const reference to the layout grid of the city.

4.2.2.6 get_max_population()

```
int City::get_max_population ( )
```

Gets the maximum population capacity of the city.

Returns

The maximum population capacity of the city.

4.2.2.7 get_name()

```
std::string City::get_name ( )
```

Gets the name of the city.

Returns

The name of the city.

4.2.2.8 get_population()

```
int City::get_population ( )
```

Gets the current population of the city.

Returns

The current population of the city.

4.2.2.9 get_year()

```
int City::get_year ( )
```

Gets the current year in the city simulation.

Returns

The current year.

4.2.2.10 set_balance()

```
void City::set_balance (
    int i )
```

Sets the balance of the city.

Parameters

<i>i</i>	The new balance of the city.
----------	------------------------------

4.2.2.11 set_dimensions()

```
void City::set_dimensions (
    Dimensions d )
```

Sets the dimensions of the city grid.

Parameters

<i>d</i>	The new dimensions of the city grid.
----------	--------------------------------------

4.2.2.12 set_layout()

```
Grid & City::set_layout ( )
```

Sets the layout of the city.

Returns

A reference to the layout grid of the city.

4.2.2.13 set_name()

```
void City::set_name (
    std::string s )
```

Sets the name of the city.

Parameters

<i>s</i>	The new name of the city.
----------	---------------------------

4.2.2.14 set_population()

```
void City::set_population (
    int i )
```

Sets the population of the city.

Parameters

<i>i</i>	The new population of the city.
----------	---------------------------------

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

- C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/city.h
- C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/city.cpp

4.3 Dimensions Struct Reference

Represents the dimensions of the city grid.

```
#include <city.h>
```

Public Attributes

- int **x**
The width of the city grid.
- int **y**
The height of the city grid.

4.3.1 Detailed Description

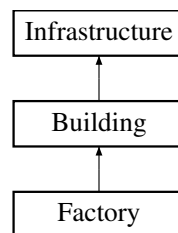
Represents the dimensions of the city grid.

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

- C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/city.h

4.4 Factory Class Reference

Inheritance diagram for Factory:



Public Member Functions

- **Factory** (char s='F')

Public Member Functions inherited from [Building](#)

- **Building** (char s)

Public Member Functions inherited from [Infrastructure](#)

- char **get_symbol** () const

Additional Inherited Members

Static Public Member Functions inherited from [Infrastructure](#)

- static std::map< char, int > **get_number_of_elements_map** ()
- static int **get_number_of** (char s)

Protected Member Functions inherited from [Infrastructure](#)

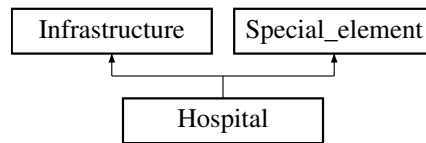
- **Infrastructure** (char s)

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

- C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/factory.h

4.5 Hospital Class Reference

Inheritance diagram for Hospital:



Public Member Functions

- **Hospital** (char s='M')

Public Member Functions inherited from [Infrastructure](#)

- char **get_symbol** () const

Public Member Functions inherited from [Special_element](#)

- int **get_maintenance_cost** () const

Private Attributes

- const int **maintenance_cost** = 800

Additional Inherited Members

Static Public Member Functions inherited from [Infrastructure](#)

- static std::map< char, int > **get_number_of_elements_map** ()
- static int **get_number_of** (char s)

Protected Member Functions inherited from [Infrastructure](#)

- **Infrastructure** (char s)

Protected Attributes inherited from [Special_element](#)

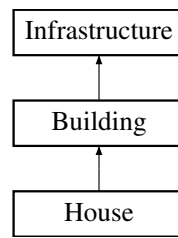
- int **maintenance_cost**

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

- C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/hospital.h

4.6 House Class Reference

Inheritance diagram for House:



Public Member Functions

- **House** (char s='H')

Public Member Functions inherited from [Building](#)

- **Building** (char s)

Public Member Functions inherited from [Infrastructure](#)

- char **get_symbol** () const

Additional Inherited Members

Static Public Member Functions inherited from [Infrastructure](#)

- static std::map< char, int > **get_number_of_elements_map** ()
- static int **get_number_of** (char s)

Protected Member Functions inherited from [Infrastructure](#)

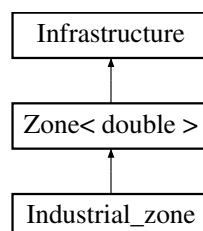
- **Infrastructure** (char s)

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

- C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/house.h

4.7 Industrial_zone Class Reference

Inheritance diagram for Industrial_zone:



Public Member Functions

- **Industrial_zone** (char s='i', double c_v=1.0, std::string c_d="air pollution")

Public Member Functions inherited from [Zone< double >](#)

- **Zone** (char s, double c_v, std::string c_d)
- void **set_characteristic_value** (double c_v)
- void **set_characteristic_description** (std::string s)
- double **get_characteristic_value** ()

Public Member Functions inherited from [Infrastructure](#)

- char **get_symbol** () const

Private Member Functions

- void [generate_characteristic_value](#) () override

Additional Inherited Members

Static Public Member Functions inherited from [Infrastructure](#)

- static std::map< char, int > **get_number_of_elements_map** ()
- static int **get_number_of** (char s)

Protected Member Functions inherited from [Infrastructure](#)

- **Infrastructure** (char s)

4.7.1 Member Function Documentation

4.7.1.1 [generate_characteristic_value\(\)](#)

```
void Industrial_zone::generate_characteristic_value ( ) [inline], [override], [private],  
[virtual]
```

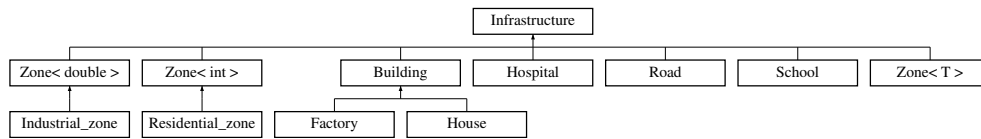
Implements [Zone< double >](#).

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

- C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/industrial_zone.h

4.8 Infrastructure Class Reference

Inheritance diagram for Infrastructure:



Public Member Functions

- char **get_symbol** () const

Static Public Member Functions

- static std::map< char, int > **get_number_of_elements_map** ()
- static int **get_number_of** (char s)

Protected Member Functions

- **Infrastructure** (char s)

Private Attributes

- const char **symbol**

Static Private Attributes

- static std::map< char, int > **number_of_elements**

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

- C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/infrastructure.h
- C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/city.cpp

4.9 Menu Class Reference

Public Member Functions

- Options **get_main** ()
- Options **get_infrastructure** ()
- Options **get_road** ()
- Options **get_zone** ()
- Options **get_building** ()
- Options **get_special_building** ()

Private Attributes

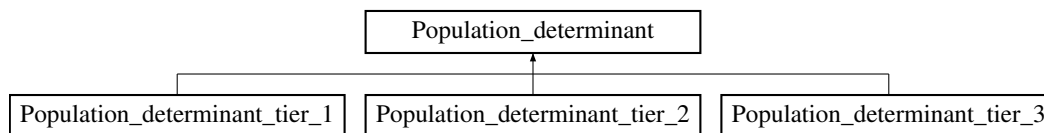
- Options **main** = { "Change Name", "City Information", "Infrastructure", "Exit" }
- Options **infrastructure** = { "Roads", "Zones", "Buildings", "Go Back" }
- Options **road** = { "Build Road", "Go Back" }
- Options **zone** = { "Chart Residential Zone", "Chart Industrial Zone", "Go Back" }
- Options **building** = { "Build a House", "Build a Factory", "Build a Special Building", "Go Back" }
- Options **special_building** = { "School", "Hospital", "Go Back" }

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

- C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/menu.h

4.10 Population_determinant Class Reference

Inheritance diagram for Population_determinant:



Public Member Functions

- virtual int **find_interest** (const int schools, const int hospitals) const =0

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

- C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/population_determinant.h

4.11 Population_determinant_context Class Reference

Public Member Functions

- void **set_determinant** (std::unique_ptr< [Population_determinant](#) > &&strategy_)
- int **execute_interest_finding** (const int schools, const int hospitals) const

Private Attributes

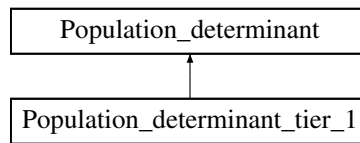
- std::unique_ptr< [Population_determinant](#) > **strategy**

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

- C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/population_determinant_↔
context.h

4.12 Population_determinant_tier_1 Class Reference

Inheritance diagram for Population_determinant_tier_1:



Private Member Functions

- int [find_interest](#) (const int schools, const int hospitals) const override

Additional Inherited Members

4.12.1 Member Function Documentation

4.12.1.1 find_interest()

```
int Population_determinant_tier_1::find_interest (
    const int schools,
    const int hospitals ) const [inline], [override], [private], [virtual]
```

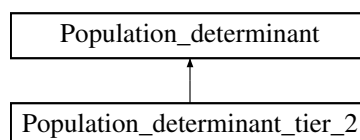
Implements [Population_determinant](#).

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

- C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/population_determinant_tier_1.h

4.13 Population_determinant_tier_2 Class Reference

Inheritance diagram for Population_determinant_tier_2:



Private Member Functions

- int [find_interest](#) (const int schools, const int hospitals) const override

Additional Inherited Members

4.13.1 Member Function Documentation

4.13.1.1 find_interest()

```
int Population_determinant_tier_2::find_interest (
    const int schools,
    const int hospitals ) const [inline], [override], [private], [virtual]
```

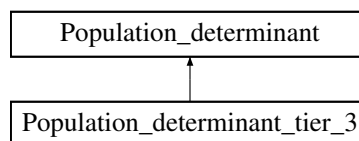
Implements [Population_determinant](#).

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

- C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/population_determinant_tier_↵
2.h

4.14 Population_determinant_tier_3 Class Reference

Inheritance diagram for Population_determinant_tier_3:



Private Member Functions

- int [find_interest](#) (const int schools, const int hospitals) const override

Additional Inherited Members

4.14.1 Member Function Documentation

4.14.1.1 find_interest()

```
int Population_determinant_tier_3::find_interest (
    const int schools,
    const int hospitals ) const [inline], [override], [private], [virtual]
```

Implements [Population_determinant](#).

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

- C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/population_determinant_tier_↵
3.h

4.15 Random Class Reference

Public Member Functions

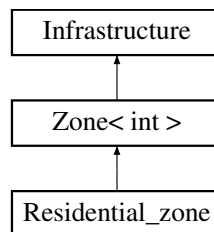
- `int rand_int (int low, int high)`

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

- `C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/random.h`

4.16 Residential_zone Class Reference

Inheritance diagram for `Residential_zone`:



Public Member Functions

- **`Residential_zone`** (`char s='r'`, `int c_v=1`, `std::string c_d="tile value"`)

Public Member Functions inherited from `Zone< int >`

- **`Zone`** (`char s`, `int c_v`, `std::string c_d`)
- void **`set_characteristic_value`** (`int c_v`)
- void **`set_characteristic_description`** (`std::string s`)
- int **`get_characteristic_value`** ()

Public Member Functions inherited from `Infrastructure`

- char **`get_symbol`** () const

Private Member Functions

- void **`generate_characteristic_value`** () override

Additional Inherited Members

Static Public Member Functions inherited from `Infrastructure`

- static `std::map< char, int >` **`get_number_of_elements_map`** ()
- static int **`get_number_of`** (`char s`)

Protected Member Functions inherited from [Infrastructure](#)

- **Infrastructure** (char s)

4.16.1 Member Function Documentation

4.16.1.1 generate_characteristic_value()

```
void Residential_zone::generate_characteristic_value ( ) [inline], [override], [private],  
[virtual]
```

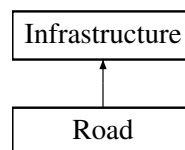
Implements [Zone< int >](#).

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

- C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/residential_zone.h

4.17 Road Class Reference

Inheritance diagram for Road:



Public Member Functions

- **Road** (char s='R')

Public Member Functions inherited from [Infrastructure](#)

- char **get_symbol** () const

Additional Inherited Members

Static Public Member Functions inherited from [Infrastructure](#)

- static std::map< char, int > **get_number_of_elements_map** ()
- static int **get_number_of** (char s)

Protected Member Functions inherited from [Infrastructure](#)

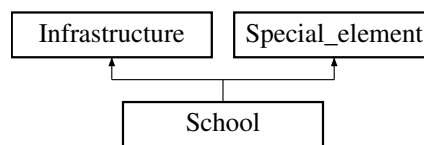
- **Infrastructure** (char s)

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

- C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/road.h

4.18 School Class Reference

Inheritance diagram for School:



Public Member Functions

- **School** (char s='S')

Public Member Functions inherited from [Infrastructure](#)

- char **get_symbol** () const

Public Member Functions inherited from [Special_element](#)

- int **get_maintenance_cost** () const

Private Attributes

- const int **maintenance_cost** = 300

Additional Inherited Members

Static Public Member Functions inherited from [Infrastructure](#)

- static std::map< char, int > **get_number_of_elements_map** ()
- static int **get_number_of** (char s)

Protected Member Functions inherited from [Infrastructure](#)

- **Infrastructure** (char s)

Protected Attributes inherited from [Special_element](#)

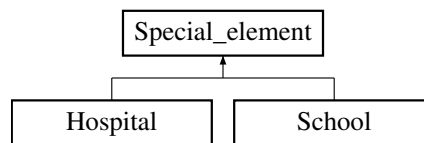
- int **maintenance_cost**

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

- C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/school.h

4.19 [Special_element](#) Class Reference

Inheritance diagram for [Special_element](#):



Public Member Functions

- int **get_maintenance_cost** () const

Protected Attributes

- int **maintenance_cost**

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

- C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/special_element.h

4.20 [View](#) Class Reference

The [View](#) class manages the user interface and interaction for managing city infrastructure.

```
#include <view.h>
```

Public Member Functions

- **View ()**
Displays greeting and calls loop.
- void **loop ()**
Loops through main menu.
- void **show_menu** (const std::vector< std::string > &menu)
Displays the menu options.
- bool **switch_main** (char ch)
Handles the main menu switch-case logic.
- void **greeting ()**
Displays the greeting message at the start of the program.
- void **press_space ()**
Waits for the user to press the spacebar to continue.
- void **print_name ()**
Clears the screen and prints the city's name.
- void **change_name ()**
Changes the name of the city based on user input.
- void **city_information ()**
Displays the city information.
- void **print_city ()**
Prints the layout of the city.
- void **next_year ()**
Advances the city by one year and performs necessary updates.
- void **goodbye ()**
Displays the goodbye message.
- void **loop_infrastructure ()**
Starts the infrastructure loop.
- bool **switch_infrastructure** (char ch)
Handles the infrastructure menu options.
- void **loop_roads ()**
Main loop for managing road construction.
- bool **switch_roads** (char ch)
Handles the road construction based on user input.
- void **build_road ()**
Initiates the road construction process.
- void **loop_zones ()**
Initializes a loop for building zones in the city.
- bool **switch_zone** (char ch)
Switches between different types of zones based on user input.
- template<typename T >
void **build_zone** ()
Builds a rectangular zone of a specified type (Residential or Industrial).
- void **loop_buildings ()**
Initializes a loop for building different types of buildings in the city.
- bool **switch_building** (char ch)
Switches between different types of buildings based on user input.
- void **build_house ()**
Builds a residential house in the city.
- void **build_factory** ()
Builds an industrial factory in the city.

- void `loop_special_buildings` ()
Initializes a loop for building special buildings in the city.
- bool `switch_special_building` (char ch)
Switches between different types of special buildings based on user input.
- void `build_school` ()
Builds a school in the city.
- void `build_hospital` ()
Builds a hospital in the city.

Private Attributes

- `City` city
- `Menu` menu

4.20.1 Detailed Description

The `View` class manages the user interface and interaction for managing city infrastructure.

4.20.2 Member Function Documentation

4.20.2.1 `build_factory()`

```
void View::build_factory ( )
```

Builds an industrial factory in the city.

This function prompts the user for coordinates and validates the build conditions before constructing a factory.

4.20.2.2 `build_hospital()`

```
void View::build_hospital ( )
```

Builds a hospital in the city.

This function prompts the user for coordinates and validates the build conditions before constructing a hospital.

4.20.2.3 `build_house()`

```
void View::build_house ( )
```

Builds a residential house in the city.

This function prompts the user for coordinates and validates the build conditions before constructing a house.

4.20.2.4 `build_school()`

```
void View::build_school ( )
```

Builds a school in the city.

This function prompts the user for coordinates and validates the build conditions before constructing a school.

4.20.2.5 `build_zone()`

```
template<typename T >  
void View::build_zone ( )
```

Builds a rectangular zone of a specified type (Residential or Industrial).

This templated function prompts the user for coordinates and validates the build conditions.

Template Parameters

<i>T</i>	Type of zone to build (Residential_zone or Industrial_zone).
----------	--

4.20.2.6 loop_buildings()

```
void View::loop_buildings ( )
```

Initializes a loop for building different types of buildings in the city.

This function displays instructions and allows the user to select and build various types of buildings.

4.20.2.7 loop_special_buildings()

```
void View::loop_special_buildings ( )
```

Initializes a loop for building special buildings in the city.

This function displays instructions and allows the user to select and build special buildings.

4.20.2.8 loop_zones()

```
void View::loop_zones ( )
```

Initializes a loop for building zones in the city.

This function displays instructions and allows the user to build rectangular zones.

4.20.2.9 show_menu()

```
void View::show_menu (
    const std::vector< std::string > & menu )
```

Displays the menu options.

Parameters

<i>menu</i>	A vector of strings representing the menu options.
-------------	--

4.20.2.10 switch_building()

```
bool View::switch_building (
    char ch )
```

Switches between different types of buildings based on user input.

Parameters

<i>ch</i>	The character representing the user's choice.
-----------	---

Returns

true if the loop should continue, false if it should break.

4.20.2.11 switch_infrastructure()

```
bool View::switch_infrastructure (  
    char ch )
```

Handles the infrastructure menu options.

Parameters

<i>ch</i>	The character input representing the menu option.
-----------	---

Returns

True if the loop should continue, false otherwise.

4.20.2.12 switch_main()

```
bool View::switch_main (  
    char ch )
```

Handles the main menu switch-case logic.

Parameters

<i>ch</i>	The character input for the menu selection.
-----------	---

Returns

true if the menu loop should continue, false if it should exit.

4.20.2.13 switch_roads()

```
bool View::switch_roads (  
    char ch )
```

Handles the road construction based on user input.

Parameters

<i>ch</i>	Character input for menu selection.
-----------	-------------------------------------

Returns

False if the user chooses to exit, true otherwise.

4.20.2.14 switch_special_building()

```
bool View::switch_special_building (
    char ch )
```

Switches between different types of special buildings based on user input.

Parameters

<i>ch</i>	The character representing the user's choice.
-----------	---

Returns

true if the loop should continue, false if it should break.

4.20.2.15 switch_zone()

```
bool View::switch_zone (
    char ch )
```

Switches between different types of zones based on user input.

Parameters

<i>ch</i>	The character representing the user's choice.
-----------	---

Returns

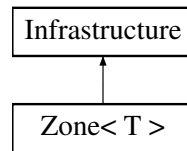
true if the loop should continue, false if it should break.

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

- C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/view.h
- C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/view.cpp

4.21 Zone< T > Class Template Reference

Inheritance diagram for Zone< T >:



Public Member Functions

- **Zone** (char s, T c_v, std::string c_d)
- void **set_characteristic_value** (T c_v)
- void **set_characteristic_description** (std::string s)
- T **get_characteristic_value** ()

Public Member Functions inherited from [Infrastructure](#)

- char **get_symbol** () const

Private Member Functions

- virtual void **generate_characteristic_value** ()=0

Private Attributes

- T **characteristic_value**
- std::string **characteristic_description**

Additional Inherited Members

Static Public Member Functions inherited from [Infrastructure](#)

- static std::map< char, int > **get_number_of_elements_map** ()
- static int **get_number_of** (char s)

Protected Member Functions inherited from [Infrastructure](#)

- **Infrastructure** (char s)

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

- C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/zone.h

Chapter 5

File Documentation

5.1 building.h

```
00001 #pragma once
00002
00003 #include "infrastructure.h"
00004
00005 class Building : public Infrastructure
00006 {
00007 public:
00008     Building(char s) : Infrastructure(s) {}
00009 };
```

5.2 city.h

```
00001 #pragma once
00002 #include <string>
00003 #include <vector>
00004 #include <memory>
00005 #include "road.h"
00006 #include "residential_zone.h"
00007 #include "industrial_zone.h"
00008 #include "house.h"
00009 #include "factory.h"
00010 #include "school.h"
00011 #include "hospital.h"
00012
00017 struct Dimensions
00018 {
00019     int x;
00020     int y;
00021 };
00022
00023 typedef std::vector<std::vector<std::unique_ptr<Infrastructure>>> Grid;
00024
00029 class City
00030 {
00031     std::string name = "MiniCity";
00032     int balance = 20000;
00033     int population = 0;
00034     int max_population = 0;
00035     int year = 2024;
00036     Dimensions dimensions = { 10, 10 };
00037     Grid layout;
00038
00039 public:
00043     City();
00044
00049     void set_name(std::string s);
00050
00055     std::string get_name();
00056
00061     void set_balance(int i);
00062
00067     int get_balance();
00068
00073     void set_population(int i);
```

```
00074
00079     void add_population(int i);
00080
00085     int get_population();
00086
00091     void add_max_population(int i);
00092
00097     int get_max_population();
00098
00103     int get_year();
00104
00108     void increment_year();
00109
00114     void set_dimensions(Dimensions d);
00115
00120     Dimensions get_dimensions();
00121
00126     Grid& set_layout();
00127
00132     const Grid& get_layout();
00133 };
```

5.3 factory.h

```
00001 #pragma once
00002
00003 #include "building.h"
00004
00005 class Factory : public Building
00006 {
00007 public:
00008     Factory(char s = 'F') : Building(s) {}
00009
00010 };
```

5.4 hospital.h

```
00001 #pragma once
00002
00003 #include "infrastructure.h"
00004 #include "special_element.h"
00005
00006 class Hospital : public Infrastructure, public Special_element
00007 {
00008     const int maintenance_cost = 800;
00009 public:
00010     Hospital(char s = 'M') : Infrastructure(s) {}
00011 };
```

5.5 house.h

```
00001 #pragma once
00002
00003 #include "building.h"
00004
00005 class House : public Building
00006 {
00007 public:
00008     House(char s = 'H') : Building(s) {}
00009
00010 };
```

5.6 industrial_zone.h

```
00001 #pragma once
00002
00003 #include "zone.h"
00004 #include "random.h"
00005
00006 class Industrial_zone: public Zone<double>
00007 {
```

```

00008     void generate_characteristic_value() override
00009     {
00010         Random random;
00011         double value = random.rand_int(10, 100) / 10.0;
00012         set_characteristic_value(value);
00013     }
00014 public:
00015     Industrial_zone(char s = 'i', double c_v = 1.0, std::string c_d = "air pollution") : Zone(s, c_v,
00016         c_d)
00017     {
00018         generate_characteristic_value();
00019     }
00019 };

```

5.7 infrastructure.h

```

00001 #pragma once
00002
00003 #include <map>
00004
00005 class Infrastructure
00006 {
00007     const char symbol;
00008     static std::map<char, int> number_of_elements;
00009 protected:
00010     Infrastructure(char s) : symbol(s)
00011     {
00012         number_of_elements[s]++;
00013     }
00014 public:
00015     Infrastructure() : Infrastructure(' ') {}
00016
00017     virtual ~Infrastructure() = default;
00018
00019     char get_symbol() const
00020     {
00021         return symbol;
00022     }
00023
00024     static std::map<char, int> get_number_of_elements_map()
00025     {
00026         return number_of_elements;
00027     }
00028
00029     static int get_number_of(char s)
00030     {
00031         return number_of_elements[s];
00032     }
00033 };

```

5.8 main_menu.h

5.9 menu.h

```

00001 #pragma once
00002
00003 #include <string>
00004 #include <vector>
00005
00006 typedef std::vector<std::string> Options;
00007
00008 class Menu
00009 {
00010     Options main = { "Change Name", "City Information", "Infrastructure", "Exit" };
00011     Options infrastructure = { "Roads", "Zones", "Buildings", "Go Back" };
00012     Options road = { "Build Road", "Go Back" };
00013     Options zone = { "Chart Residential Zone", "Chart Industrial Zone", "Go Back" };
00014     Options building = { "Build a House", "Build a Factory", "Build a Special Building", "Go Back" };
00015     Options special_building = { "School", "Hospital", "Go Back" };
00016 public:
00017     Options get_main()
00018     {
00019         return main;
00020     }
00021
00022     Options get_infrastructure()

```

```

00023     {
00024         return infrastructure;
00025     }
00026
00027     Options get_road()
00028     {
00029         return road;
00030     }
00031
00032     Options get_zone()
00033     {
00034         return zone;
00035     }
00036
00037     Options get_building()
00038     {
00039         return building;
00040     }
00041
00042     Options get_special_building()
00043     {
00044         return special_building;
00045     }
00046 };

```

5.10 population_determinant.h

```

00001 #pragma once
00002
00003 #include <map>
00004
00005 class Population_determinant
00006 {
00007 public:
00008     virtual ~Population_determinant() = default;
00009     virtual int find_interest(const int schools, const int hospitals) const = 0;
00010 };

```

5.11 population_determinant_context.h

```

00001 #pragma once
00002
00003 #include <memory>
00004 #include "population_determinant.h"
00005
00006 class Population_determinant_context
00007 {
00008     std::unique_ptr<Population_determinant> strategy;
00009 public:
00010     void set_determinant(std::unique_ptr<Population_determinant> &&strategy_)
00011     {
00012         strategy = std::move(strategy_);
00013     }
00014
00015     int execute_interest_finding(const int schools, const int hospitals) const
00016     {
00017         return strategy->find_interest(schools, hospitals);
00018     }
00019 };

```

5.12 population_determinant_tier_1.h

```

00001 #pragma once
00002
00003 #include <algorithm>
00004 #include "population_determinant.h"
00005 #include "random.h"
00006
00007 class Population_determinant_tier_1 : public Population_determinant
00008 {
00009     int find_interest(const int schools, const int hospitals) const override
00010     {
00011         Random random;
00012         return random.rand_int(0, 4);
00013     }
00014 };

```


5.13 population_determinant_tier_2.h

```

00001 #pragma once
00002
00003 #include <algorithm>
00004 #include "population_determinant.h"
00005 #include "random.h"
00006
00007 class Population_determinant_tier_2 : public Population_determinant
00008 {
00009     int find_interest(const int schools, const int hospitals) const override
00010     {
00011         Random random;
00012         return random.rand_int(1, 6) + schools * 2 + hospitals * 2;
00013     }
00014 };

```

5.14 population_determinant_tier_3.h

```

00001 #pragma once
00002
00003 #include <algorithm>
00004 #include "population_determinant.h"
00005 #include "random.h"
00006
00007 class Population_determinant_tier_3 : public Population_determinant
00008 {
00009     int find_interest(const int schools, const int hospitals) const override
00010     {
00011         Random random;
00012         return random.rand_int(5, 10) + schools * 3 + hospitals * 3;
00013     }
00014 };

```

5.15 population_determinant_tiers.h

```

00001 #pragma once
00002
00003 #include "population_determinant_tier_1.h"
00004 #include "population_determinant_tier_2.h"
00005 #include "population_determinant_tier_3.h"

```

5.16 random.h

```

00001 #pragma once
00002
00003 #include <random>
00004
00005 class Random
00006 {
00007 public:
00008     int rand_int(int low, int high)
00009     {
00010         static std::default_random_engine re{ std::random_device{}() };
00011         using Dist = std::uniform_int_distribution<int>;
00012         static Dist uid{};
00013         return uid(re, Dist::param_type{ low, high });
00014     }
00015 };

```

5.17 residential_zone.h

```

00001 #pragma once
00002
00003 #include "zone.h"
00004 #include "random.h"
00005
00006 class Residential_zone : public Zone<int>
00007 {
00008     void generate_characteristic_value() override

```

```

00009     {
00010         Random random;
00011         int value = random.rand_int(10, 100);
00012         set_characteristic_value(value);
00013     }
00014 public:
00015     Residential_zone(char s = 'r', int c_v = 1, std::string c_d = "tile value") : Zone(s, c_v, c_d)
00016     {
00017         generate_characteristic_value();
00018     }
00019 };

```

5.18 road.h

```

00001 #pragma once
00002
00003 #include "infrastructure.h"
00004
00005 class Road : public Infrastructure
00006 {
00007 public:
00008     Road(char s = 'R') : Infrastructure(s) {}
00009 };

```

5.19 school.h

```

00001 #pragma once
00002
00003 #include "infrastructure.h"
00004 #include "special_element.h"
00005
00006 class School : public Infrastructure, public Special_element
00007 {
00008     const int maintenance_cost = 300;
00009 public:
00010     School(char s = 'S') : Infrastructure(s) {}
00011 };

```

5.20 special_element.h

```

00001 #pragma once
00002
00003 class Special_element
00004 {
00005 protected:
00006     int maintenance_cost;
00007 public:
00008     virtual ~Special_element() = default;
00009
00010     int get_maintenance_cost() const
00011     {
00012         return maintenance_cost;
00013     }
00014
00015 };

```

5.21 view.h

```

00001 #pragma once
00002 #include <iostream>
00003 #include <windows.h>
00004 #include <cmath>
00005 #include <utility>
00006 #include <memory>
00007 #include <algorithm>
00008 #include "conio.h"
00009 #include "city.h"
00010 #include "menu.h"
00011 #include "population_determinant_context.h"
00012 #include "population_determinant_tiers.h"
00013 #include "random.h"

```

```

00014
00015
00019 class View
00020 {
00021     City city;
00022     Menu menu;
00023
00024 public:
00028     View();
00029
00033     void loop();
00034
00040     void show_menu(const std::vector<std::string>& menu);
00041
00048     bool switch_main(char ch);
00049
00053     void greeting();
00054
00058     void press_space();
00059
00063     void print_name();
00064
00068     void change_name();
00069
00070
00071     /* ----- City Information ----- */
00072
00076     void city_information();
00077
00081     void print_city();
00082
00086     void next_year();
00087
00091     void goodbye();
00092
00093     /* ----- Infrastructure ----- */
00094
00098     void loop_infrastructure();
00099
00105     bool switch_infrastructure(char ch);
00106
00107     /* ----- Roads ----- */
00108
00112     void loop_roads();
00113
00119     bool switch_roads(char ch);
00120
00124     void build_road();
00125
00126     /* ----- Zones ----- */
00127
00133     void loop_zones();
00134
00141     bool switch_zone(char ch);
00142
00149     template <typename T>
00150     void build_zone();
00151
00157     void loop_buildings();
00158
00165     bool switch_building(char ch);
00166
00173     void build_house();
00174
00181     void build_factory();
00182
00188     void loop_special_buildings();
00189
00196     bool switch_special_building(char ch);
00197
00204     void build_school();
00205
00212     void build_hospital();
00213 };

```

5.22 zone.h

```

00001 #pragma once
00002
00003 #include "infrastructure.h"
00004 #include "string"
00005
00006 template <typename T>

```

```
00007 class Zone : public Infrastructure
00008 {
00009     T characteristic_value;
00010     std::string characteristic_description;
00011     virtual void generate_characteristic_value() = 0;
00012 public:
00013     Zone(char s, T c_v, std::string c_d) : Infrastructure(s), characteristic_value(c_v),
characteristic_description(c_d) {}
00014
00015     void set_characteristic_value(T c_v)
00016     {
00017         characteristic_value = c_v;
00018     }
00019
00020     void set_characteristic_description(std::string s)
00021     {
00022         characteristic_description = s;
00023     }
00024
00025     T get_characteristic_value()
00026     {
00027         return characteristic_value;
00028     }
00029
00030 };
```

Index

add_max_population
City, [9](#)

add_population
City, [9](#)

build_factory
View, [26](#)

build_hospital
View, [26](#)

build_house
View, [26](#)

build_school
View, [26](#)

build_zone
View, [26](#)

Building, [7](#)

C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/
[31](#)

C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/
[31](#)

C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/
[32](#)

C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/
[32](#)

C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/
[32](#)

C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/industrial_zone.h,
[32](#)

C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/infrastructure.h,
[33](#)

C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/main_menu.h,
[33](#)

C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/population_determinant.h,
[33](#)

C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/population_determinant.h,
[34](#)

C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/population_determinant_context.h,
[34](#)

C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/population_determinant_tier_1.h,
[34](#)

C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/population_determinant_tier_2.h,
[35](#)

C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/population_determinant_tier_3.h,
[35](#)

C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/population_determinant_tiers.h,
[35](#)

C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/random.h,
[35](#)

C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/
[35](#)

C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/
[36](#)

C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/
[36](#)

C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/
[36](#)

C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/
[36](#)

C:/Users/Jarek/source/repos/f5292847-gr24-repo/Projekt/MiniCity/MiniCity/
[37](#)

City, [8](#)

add_max_population, [9](#)

add_population, [9](#)

get_balance, [9](#)

get_dimensions, [10](#)

get_layout, [10](#)

get_max_population, [10](#)

get_name, [10](#)

get_population, [10](#)

get_year, [11](#)

set_balance, [11](#)

set_dimensions, [11](#)

set_layout, [11](#)

set_name, [12](#)

set_population, [12](#)

Dimensions, [12](#)

Factory, [13](#)

MiniCity, [13](#)

MiniCity, [13](#)

Population_determinant_tier_1, [19](#)

Population_determinant_tier_2, [20](#)

Population_determinant_tier_3, [20](#)

generate_characteristic_value

Industrial_zone, [16](#)

Residential_zone, [22](#)

get_balance

get_dimensions

get_layout

get_max_population

get_name

get_population

City, [10](#)
 get_year
 City, [11](#)

 Hospital, [14](#)
 House, [15](#)

 Industrial_zone, [15](#)
 generate_characteristic_value, [16](#)
 Infrastructure, [17](#)

 loop_buildings
 View, [27](#)
 loop_special_buildings
 View, [27](#)
 loop_zones
 View, [27](#)

 Menu, [17](#)

 Population_determinant, [18](#)
 Population_determinant_context, [18](#)
 Population_determinant_tier_1, [19](#)
 find_interest, [19](#)
 Population_determinant_tier_2, [19](#)
 find_interest, [20](#)
 Population_determinant_tier_3, [20](#)
 find_interest, [20](#)

 Random, [21](#)
 Residential_zone, [21](#)
 generate_characteristic_value, [22](#)
 Road, [22](#)

 School, [23](#)
 set_balance
 City, [11](#)
 set_dimensions
 City, [11](#)
 set_layout
 City, [11](#)
 set_name
 City, [12](#)
 set_population
 City, [12](#)
 show_menu
 View, [27](#)
 Special_element, [24](#)
 switch_building
 View, [27](#)
 switch_infrastructure
 View, [28](#)
 switch_main
 View, [28](#)
 switch_roads
 View, [28](#)
 switch_special_building
 View, [29](#)
 switch_zone
 View, [29](#)

 View, [24](#)
 build_factory, [26](#)
 build_hospital, [26](#)
 build_house, [26](#)
 build_school, [26](#)
 build_zone, [26](#)
 loop_buildings, [27](#)
 loop_special_buildings, [27](#)
 loop_zones, [27](#)
 show_menu, [27](#)
 switch_building, [27](#)
 switch_infrastructure, [28](#)
 switch_main, [28](#)
 switch_roads, [28](#)
 switch_special_building, [29](#)
 switch_zone, [29](#)

 Zone< T >, [29](#)