MiniCity

Generated by Doxygen 1.10.0

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

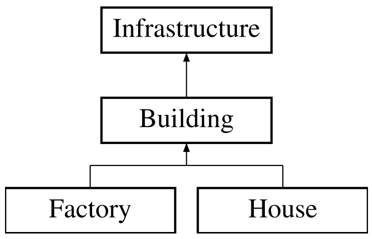
Here is a list of all documented files with brief descriptions:

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	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	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	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

d	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

s	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	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

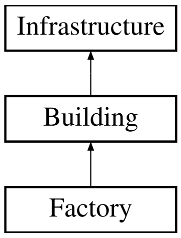
Represents the dimensions of the city grid.

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

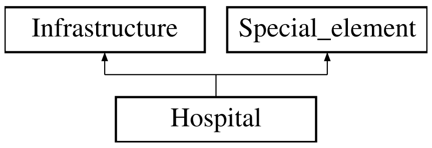
- **Infrastructure** (char s)

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

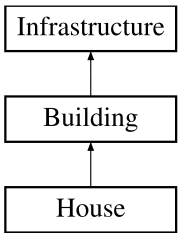
- int **maintenance_cost**

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

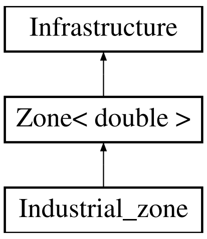
- **Infrastructure** (char s)

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]
```

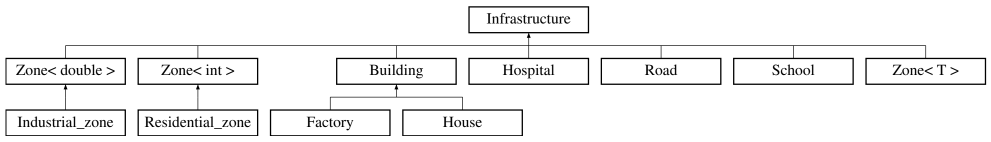
Implements Zone< double >.

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

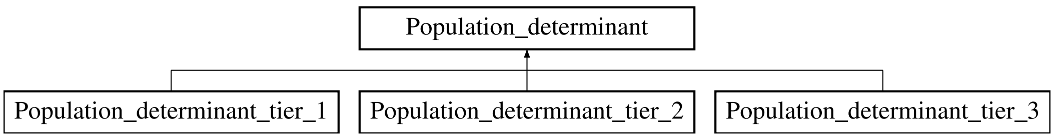
- Options **main** = { "Change Name", "City Information", "Infrastructure", "Exit" }
- Options **infrastructure** = { "Roads", "Zones", "Buildings", "Go Back"}
- Options **road** = { "Build Road", "Go Back" }
- Options **zone** = { "Chart Residential Zone", "Chart Industrial Zone","Go Back" }
- Options **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:


Population_determinant
Population_determinant_tier_1
Population_determinant_tier_2
Population_determinant_tier_3

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

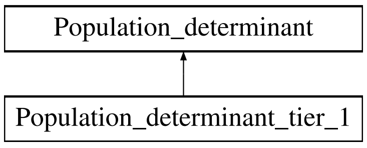
- std::unique_ptr< Population_determinant > **strategy**

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]
```

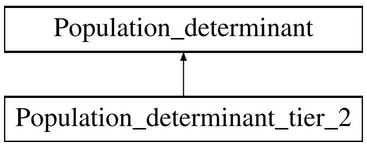
Implements Population_determinant.

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]
```

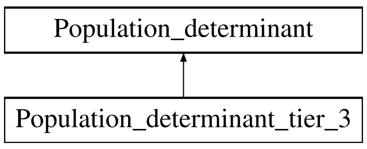
Implements Population_determinant.

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

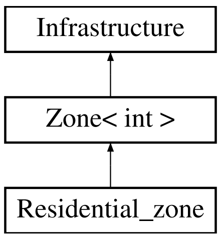
- int **rand_int** (int low, int high)

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]
```

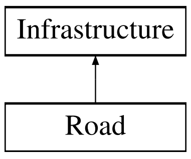
Implements Zone< int >.

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

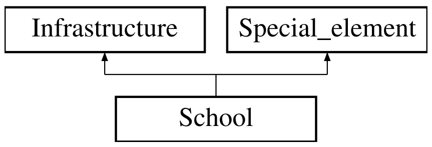
- **Infrastructure** (char s)

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

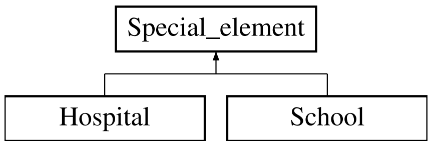
- int **maintenance_cost**

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

T	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

menu	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

ch	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

ch	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

ch	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

ch	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

ch	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

ch	The character representing the user's choice.

Returns

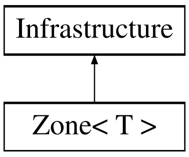
true if the loop should continue, false if it should break.

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

```
#pragma once

#include "infrastructure.h"

class Building : public Infrastructure
{
public:
    Building(char s) : Infrastructure(s) {}
};
```

5.2 city.h

```
#pragma once
#include <string>
#include <vector>
#include <memory>
#include "road.h"
#include "residential_zone.h"
#include "industrial_zone.h"
#include "house.h"
#include "factory.h"
#include "school.h"
#include "hospital.h"

struct Dimensions
{
    int x;
    int y;
};

typedef std::vector<std::vector<std::unique_ptr<Infrastructure»> Grid;

class City
{
    std::string name = "MiniCity";
    int balance = 20000;
    int population = 0;
    int max_population = 0;
    int year = 2024;
    Dimensions dimensions = { 10, 10 };
    Grid layout;

public:
    City();

    void set_name(std::string s);

    std::string get_name();

    void set_balance(int i);

    int get_balance();

    void set_population(int i);

    void add_population(int i);

    int get_population();

    void add_max_population(int i);

    int get_max_population();

    int get_year();

    void increment_year();

    void set_dimensions(Dimensions d);

    Dimensions get_dimensions();

    Grid& set_layout();

    const Grid& get_layout();
};
```

5.3 factory.h

```
#pragma once

#include "building.h"

class Factory : public Building
{
public:
    Factory(char s = 'F') : Building(s) {}

};
```

5.4 hospital.h

```
#pragma once

#include "infrastructure.h"
#include "special_element.h"

class Hospital : public Infrastructure, public Special_element
{
    const int maintenance_cost = 800;
public:
    Hospital(char s = 'M') : Infrastructure(s) {}
};
```

5.5 house.h

```
#pragma once

#include "building.h"

class House : public Building
{
public:
    House(char s = 'H') : Building(s) {}

};
```

5.6 industrial_zone.h

```
#pragma once

#include "zone.h"
#include "random.h"

class Industrial_zone: public Zone<double>
{
    void generate_characteristic_value() override
    {
        Random random;
        double value = random.rand_int(10, 100) / 10.0;
        set_characteristic_value(value);
    }
public:
    Industrial_zone(char s = 'i', double c_v = 1.0, std::string c_d = "air pollution") : Zone(s, c_v,
c_d)
    {
        generate_characteristic_value();
    }
};
```

5.7 infrastructure.h

```
#pragma once

#include <map>

class Infrastructure
{
    const char symbol;
    static std::map<char, int> number_of_elements;
protected:
    Infrastructure(char s) : symbol(s)
    {
        number_of_elements[s]++;
    }
public:
    Infrastructure() : Infrastructure(' ') {}

    virtual ~Infrastructure() = default;

    char get_symbol() const
    {
        return symbol;
    }

    static std::map<char, int> get_number_of_elements_map()
    {
        return number_of_elements;
    }

    static int get_number_of(char s)
    {
        return number_of_elements[s];
    }
};
```

5.8 main_menu.h

5.9 menu.h

```
#pragma once

#include <string>
#include <vector>

typedef std::vector<std::string> Options;

class Menu
{
    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"};
public:
    Options get_main()
    {
        return main;
    }

    Options get_infrastructure()
    {
        return infrastructure;
    }

    Options get_road()
    {
        return road;
    }

    Options get_zone()
    {
        return zone;
    }

    Options get_building()
    {
        return building;
    }

    Options get_special_building()
    {
        return special_building;
    }
};
```

5.10 population_determinant.h

```
#pragma once

#include <map>

class Population_determinant
{
public:
    virtual ~Population_determinant() = default;
    virtual int find_interest(const int schools, const int hospitals) const = 0;
};
```

5.11 population_determinant_context.h

```
#pragma once

#include <memory>
#include "population_determinant.h"

class Population_determinant_context
{
    std::unique_ptr<Population_determinant> strategy;
public:
    void set_determinant(std::unique_ptr<Population_determinant> &&strategy_)
    {
        strategy = std::move(strategy_);
    }

    int execute_interest_finding(const int schools, const int hospitals) const
    {
        return strategy->find_interest(schools, hospitals);
    }
};
```

5.12 population_determinant_tier_1.h

```
#pragma once

#include <algorithm>
#include "population_determinant.h"
#include "random.h"

class Population_determinant_tier_1 : public Population_determinant
{
    int find_interest(const int schools, const int hospitals) const override
    {
        Random random;
        return random.rand_int(0, 4);
    }
};
```

5.13 population_determinant_tier_2.h

```
#pragma once

#include <algorithm>
#include "population_determinant.h"
#include "random.h"

class Population_determinant_tier_2 : public Population_determinant
{
    int find_interest(const int schools, const int hospitals) const override
    {
        Random random;
        return random.rand_int(1, 6) + schools * 2 + hospitals * 2;
    }
};
```

5.14 population_determinant_tier_3.h

```
#pragma once

#include <algorithm>
#include "population_determinant.h"
#include "random.h"

class Population_determinant_tier_3 : public Population_determinant
{
    int find_interest(const int schools, const int hospitals) const override
    {
        Random random;
        return random.rand_int(5, 10) + schools * 3 + hospitals * 3;
    }
};
```

5.15 population_determinant_tiers.h

```
#pragma once

#include "population_determinant_tier_1.h"
#include "population_determinant_tier_2.h"
#include "population_determinant_tier_3.h"
```

5.16 random.h

```
#pragma once

#include <random>

class Random
{
public:
    int rand_int(int low, int high)
    {
        static std::default_random_engine re{ std::random_device{}() };
        using Dist = std::uniform_int_distribution<int>;
        static Dist uid{};
        return uid(re, Dist::param_type{ low,high });
    }
};
```

5.17 residential_zone.h

```
#pragma once

#include "zone.h"
#include "random.h"

class Residential_zone : public Zone<int>
{
    void generate_characteristic_value() override
    {
        Random random;
        int value = random.rand_int(10, 100);
        set_characteristic_value(value);
    }
public:
    Residential_zone(char s = 'r', int c_v = 1, std::string c_d = "tile value") : Zone(s, c_v, c_d)
    {
        generate_characteristic_value();
    }
};
```

5.18 road.h

```
#pragma once

#include "infrastructure.h"

class Road : public Infrastructure
{
public:
    Road(char s = 'R') : Infrastructure(s) {}
};
```

5.19 school.h

```
#pragma once

#include "infrastructure.h"
#include "special_element.h"

class School : public Infrastructure, public Special_element
{
    const int maintenance_cost = 300;
public:
    School(char s = 'S') : Infrastructure(s) {}
};
```

5.20 special_element.h

```
#pragma once

class Special_element
{
protected:
    int maintenance_cost;
public:
    virtual ~Special_element() = default;

    int get_maintenance_cost() const
    {
        return maintenance_cost;
    }

};
```

5.21 view.h

```
#pragma once
#include <iostream>
#include <windows.h>
#include <cmath>
#include <utility>
#include <memory>
#include <algorithm>
#include "conio.h"
#include "city.h"
#include "menu.h"
#include "population_determinant_context.h"
#include "population_determinant_tiers.h"
#include "random.h"
```

```

class View
{
    City city;
    Menu menu;

public:
    View();

    void loop();

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

    bool switch_main(char ch);

    void greeting();

    void press_space();

    void print_name();

    void change_name();


    /* ------------------------------ City Information ------------------------------ */

    void city_information();

    void print_city();

    void next_year();

    void goodbye();

    /* ------------------------------ Infrastructure ------------------------------ */

    void loop_infrastructure();

    bool switch_infrastructure(char ch);

    /* ------------------------------ Roads ------------------------------ */

    void loop_roads();

    bool switch_roads(char ch);

    void build_road();

    /* ------------------------------ Zones ------------------------------ */

    void loop_zones();

    bool switch_zone(char ch);

    template <typename T>
    void build_zone();

    void loop_buildings();

    bool switch_building(char ch);

    void build_house();

    void build_factory();

    void loop_special_buildings();

    bool switch_special_building(char ch);

    void build_school();

    void build_hospital();
};
```

5.22 zone.h

```
#pragma once

#include "infrastructure.h"
#include "string"

template <typename T>
```

```
class Zone : public Infrastructure
{
    T characteristic_value;
    std::string characteristic_description;
    virtual void generate_characteristic_value() = 0;
public:
    Zone(char s, T c_v, std::string c_d) : Infrastructure(s), characteristic_value(c_v),
characteristic_description(c_d) {}

    void set_characteristic_value(T c_v)
    {
        characteristic_value = c_v;
    }

    void set_characteristic_description(std::string s)
    {
        characteristic_description = s;
    }

    T get_characteristic_value()
    {
        return characteristic_value;
    }

};
```

Index