

My Project

Generated by Doxygen 1.13.2

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 GameMap Class Reference	7
4.1.1 Detailed Description	10
4.1.2 Constructor & Destructor Documentation	10
4.1.2.1 GameMap()	10
4.1.3 Member Function Documentation	10
4.1.3.1 canUpgradeStreet()	10
4.1.3.2 displayMap()	11
4.1.3.3 movebahn()	12
4.1.3.4 moveHub()	12
4.2 Graph Class Reference	13
4.2.1 Detailed Description	14
4.2.2 Constructor & Destructor Documentation	15
4.2.2.1 Graph()	15
4.2.3 Member Function Documentation	15
4.2.3.1 getNextNode()	15
4.2.3.2 getNodeName()	15
4.2.3.3 getTile()	16
4.2.3.4 hasEdge()	16
4.2.4 Member Data Documentation	17
4.2.4.1 adjacencyMatrix	17
4.2.4.2 groups	17
4.2.4.3 nodeNames	17
4.2.4.4 PropertyTiles	18
4.2.4.5 tiles	18
4.3 PropertyTile Class Reference	18
4.3.1 Constructor & Destructor Documentation	21
4.3.1.1 PropertyTile()	21
4.3.2 Member Function Documentation	22
4.3.2.1 calculateRent()	22
4.3.2.2 displayInfo()	22
4.3.2.3 getBuildingLevel()	22
4.3.2.4 getGroupId()	22
4.3.2.5 getOwnerId()	23

4.3.2.6 getPrice()	23
4.3.2.7 getPropertyType()	23
4.3.2.8 getRent()	23
4.3.2.9 getRentLevels()	23
4.3.2.10 getTypeString()	23
4.3.2.11 setOwner()	24
4.3.2.12 setRentLevels()	24
4.4 SpecialTile Class Reference	24
4.4.1 Constructor & Destructor Documentation	27
4.4.1.1 SpecialTile()	27
4.4.2 Member Function Documentation	27
4.4.2.1 displayInfo()	27
4.4.2.2 getSpecialType()	28
4.4.2.3 getTypeString()	28
4.5 TestRunner Class Reference	29
4.5.1 Member Function Documentation	29
4.5.1.1 myAssert()	29
4.5.1.2 printSummary()	30
4.5.1.3 runTest()	30
4.5.2 Member Data Documentation	31
4.5.2.1 failed	31
4.5.2.2 passed	31
4.6 Tile Class Reference	31
4.6.1 Constructor & Destructor Documentation	33
4.6.1.1 Tile()	33
4.6.1.2 ~Tile()	34
4.6.2 Member Function Documentation	34
4.6.2.1 displayInfo()	34
4.6.2.2 getId()	34
4.6.2.3 getName()	34
4.6.2.4 getTypeString()	34
4.6.3 Member Data Documentation	34
4.6.3.1 id	34
4.6.3.2 name	34
5 File Documentation	35
5.1 Graph.cpp File Reference	35
5.2 Graph.hpp File Reference	35
5.2.1 Macro Definition Documentation	37
5.2.1.1 Adlerstraße	37
5.2.1.2 Amalienstraße	37
5.2.1.3 Brauerstraße	38

5.2.1.4 DurlacherAllee	38
5.2.1.5 Ebertstraße	38
5.2.1.6 Einkommensteuer	38
5.2.1.7 Elektrizitätswerk	38
5.2.1.8 Erbprinzenstraße	38
5.2.1.9 Ereignisfeld	38
5.2.1.10 Ereignisfeld2	38
5.2.1.11 Ettlingerstraße	38
5.2.1.12 Fastplatz	38
5.2.1.13 FreiParken	39
5.2.1.14 Gefängnis	39
5.2.1.15 GeheinsGefängnis	39
5.2.1.16 Gemeinschaftsfeld	39
5.2.1.17 Gemeinschaftsfeld2	39
5.2.1.18 Gemeinschaftsfeld3	39
5.2.1.19 Gemeinschaftsfeld4	39
5.2.1.20 Hauptbahnhof	39
5.2.1.21 Herrenstraße	39
5.2.1.22 Hildapromenade	39
5.2.1.23 Hirschstraße	40
5.2.1.24 Hubschrauberlandeplatze	40
5.2.1.25 KaiserAllee	40
5.2.1.26 Kaiserstraße	40
5.2.1.27 Karlfriedrichstraße	40
5.2.1.28 Karlstraße	40
5.2.1.29 Kriegsstraße	40
5.2.1.30 Kronenstraße	40
5.2.1.31 LOS	40
5.2.1.32 Moltkestraße	40
5.2.1.33 OstBahnhof	41
5.2.1.34 Rüppurrerstraße	41
5.2.1.35 Schlossplatz	41
5.2.1.36 TOTAL_NODES	41
5.2.1.37 Waldstraße	41
5.2.1.38 Wasserwerk	41
5.2.1.39 WestBahnhof	41
5.2.1.40 Zirkel	41
5.2.1.41 Zusatzsteuer	41
5.3 Graph.hpp	42
5.4 Map.cpp File Reference	43
5.5 Map.hpp File Reference	43
5.6 Map.hpp	44

5.7 PropertyTile.cpp File Reference	45
5.8 PropertyTile.hpp File Reference	45
5.8.1 Enumeration Type Documentation	47
5.8.1.1 PropertyType	47
5.9 PropertyTile.hpp	47
5.10 SpecialTile.cpp File Reference	48
5.11 SpecialTile.hpp File Reference	48
5.11.1 Enumeration Type Documentation	49
5.11.1.1 SpecialType	49
5.12 SpecialTile.hpp	50
5.13 test.cpp File Reference	50
5.13.1 Function Documentation	51
5.13.1.1 main()	51
5.13.1.2 testCanUpgradeStreet()	51
5.13.1.3 testDisplayMap()	52
5.13.1.4 testGraphInitialization()	53
5.13.1.5 testMovementMethods()	53
5.14 Tile.hpp File Reference	54
5.15 Tile.hpp	55
Index	57

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Graph	13
GameMap	7
TestRunner	29
Tile	31
PropertyTile	18
SpecialTile	24

Chapter 2

Class Index

2.1 Class List

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

GameMap	This is the implementation of the Map class	7
Graph	This is the Graph class	13
PropertyTile	18
SpecialTile	24
TestRunner	29
Tile	31

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Graph.cpp	35
Graph.hpp	35
Map.cpp	43
Map.hpp	43
PropertyTile.cpp	45
PropertyTile.hpp	45
SpecialTile.cpp	48
SpecialTile.hpp	48
test.cpp	50
Tile.hpp	54

Chapter 4

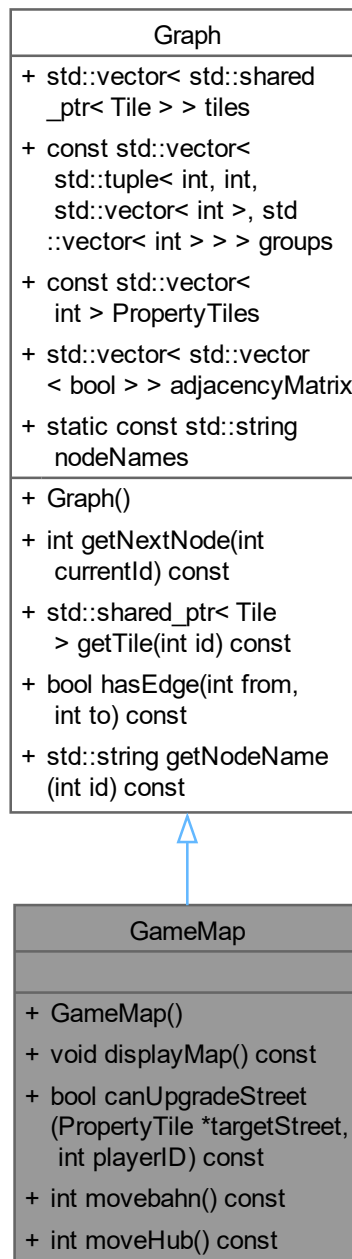
Class Documentation

4.1 GameMap Class Reference

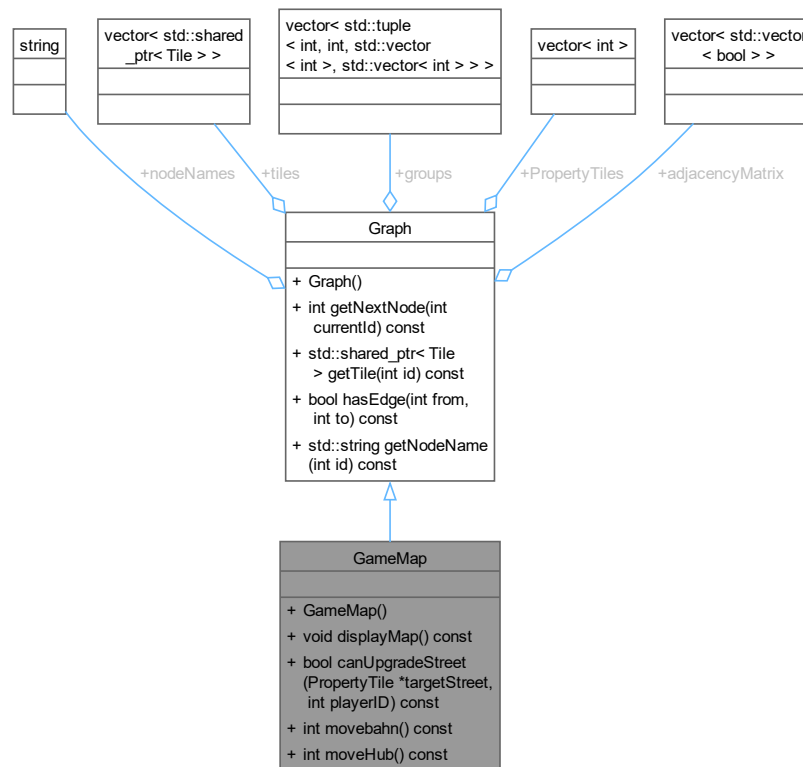
This is the implementation of the Map class.

```
#include <Map.hpp>
```

Inheritance diagram for GameMap:



Collaboration diagram for GameMap:



Public Member Functions

- [GameMap](#) ()
- void [displayMap](#) () const
//Function: loop to display all node information
- bool [canUpgradeStreet](#) ([PropertyTile](#) *targetStreet, int playerId) const
//Function: Verify if the street can be updated //HKI-9 Map: Implementierung der Strassenfelder HKI-10 Map↔ : Entwicklung der Strassen-Eigenschaften
- int [movebahn](#) () const
HKI-11 Map: Implementierung der Bahnhofsfelder.
- int [moveHub](#) () const
HKI-12 Map: Implementierung des Hubschrauberlandeplatzes.

Public Member Functions inherited from [Graph](#)

- [Graph](#) ()
//Constructor, initialize the adjacency matrix to construct the edges of the graph, and use a loop to insert nodes ([Tile](#)) according to the rent table
- int [getNextNode](#) (int currentId) const
Function: Given the current position ID, returns the ID of the next feasible tile.
- std::shared_ptr< [Tile](#) > [getTile](#) (int id) const
Function: Given a [Tile](#) ID, return a [Tile](#) object.

- bool [hasEdge](#) (int from, int to) const
Function: Given a [Tile](#) ID, return a [Tile](#) name.
- std::string [getNodeName](#) (int id) const
Function: Determine whether two Tiles are connected.

Additional Inherited Members

Public Attributes inherited from [Graph](#)

- std::vector< std::shared_ptr< [Tile](#) > > [tiles](#)
- const std::vector< std::tuple< int, int, std::vector< int >, std::vector< int > > > [groups](#)
- const std::vector< int > [PropertyTiles](#)
street
- std::vector< std::vector< bool > > [adjacencyMatrix](#)

Static Public Attributes inherited from [Graph](#)

- static const std::string [nodeNames](#) [[TOTAL_NODES](#)]

4.1.1 Detailed Description

This is the implementation of the Map class.

Contains functions for displaying map information, moving train stations and airports, and determining whether streets are upgradeable
HKI-8 Map: Erstellung einer Spielfeld-Klasse

4.1.2 Constructor & Destructor Documentation

4.1.2.1 GameMap()

```
GameMap::GameMap ()
```

Here is the call graph for this function:



4.1.3 Member Function Documentation

4.1.3.1 canUpgradeStreet()

```
bool GameMap::canUpgradeStreet (
    PropertyTile * targetStreet,
    int playerId) const
```

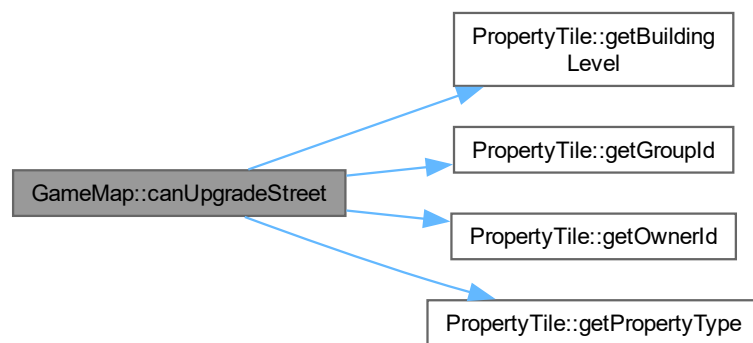
//Function: Verify if the street can be updated //HKI-9 Map: Implementierung der Strassenfelder HKI-10 Map↔
: Entwicklung der Strassen-Eigenschaften

Parameters

	<i>targetStreet</i>
	<i>playerID</i>

Returns

Here is the call graph for this function:



Here is the caller graph for this function:



4.1.3.2 displayMap()

```
void GameMap::displayMap () const
```

```
//Function: loop to display all node information
```

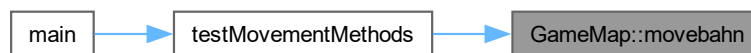
4.1.3.3 movebahn()

```
int GameMap::movebahn () const
```

HKI-11 Map: Implementierung der Bahnhofsfelder.

Returns

Here is the caller graph for this function:



4.1.3.4 moveHub()

```
int GameMap::moveHub () const
```

HKI-12 Map: Implementierung des Hubschrauberlandeplatzes.

Returns

Here is the caller graph for this function:



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

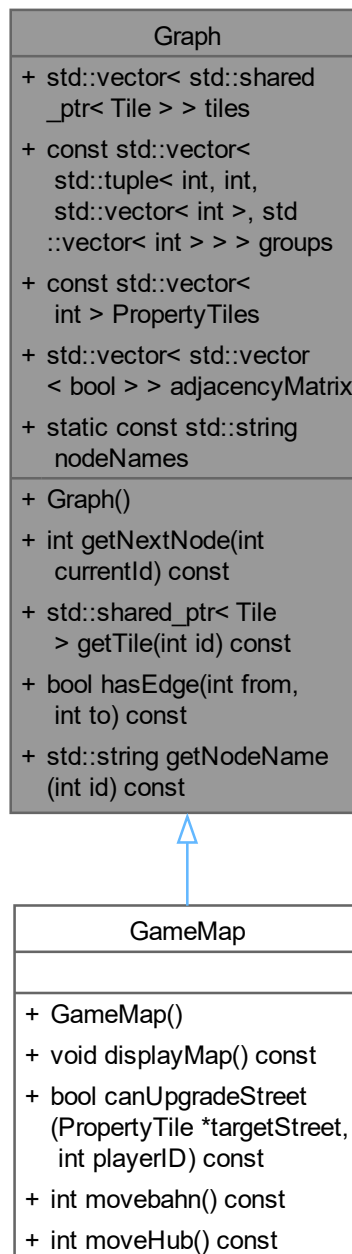
- [Map.hpp](#)
- [Map.cpp](#)

4.2 Graph Class Reference

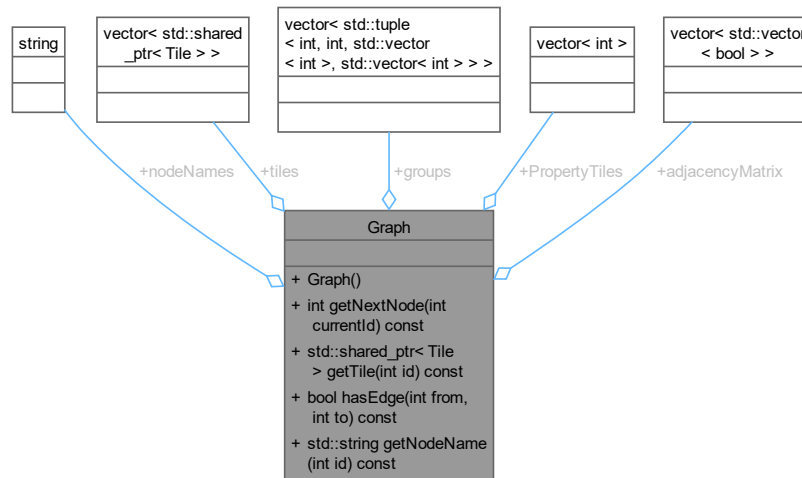
This is the [Graph](#) class.

```
#include <Graph.hpp>
```

Inheritance diagram for Graph:



Collaboration diagram for Graph:



Public Member Functions

- [Graph](#) ()
//Constructor, initialize the adjacency matrix to construct the edges of the graph, and use a loop to insert nodes ([Tile](#)) according to the rent table
- int [getNextNode](#) (int currentId) const
Function: Given the current position ID, returns the ID of the next feasible tile.
- std::shared_ptr< [Tile](#) > [getTile](#) (int id) const
Function: Given a [Tile](#) ID, return a [Tile](#) object.
- bool [hasEdge](#) (int from, int to) const
Function: Given a [Tile](#) ID, return a [Tile](#) name.
- std::string [getNodeName](#) (int id) const
Function: Determine whether two Tiles are connected.

Public Attributes

- std::vector< std::shared_ptr< [Tile](#) > > [tiles](#)
- const std::vector< std::tuple< int, int, std::vector< int >, std::vector< int > > > > [groups](#)
- const std::vector< int > [PropertyTiles](#)
street
- std::vector< std::vector< bool > > [adjacencyMatrix](#)

Static Public Attributes

- static const std::string [nodeNames](#) [[TOTAL_NODES](#)]

4.2.1 Detailed Description

This is the [Graph](#) class.

Implemented the Karlsruhe ring tile graph structure

4.2.2 Constructor & Destructor Documentation

4.2.2.1 Graph()

```
Graph::Graph ()
```

//Constructor, initialize the adjacency matrix to construct the edges of the graph, and use a loop to insert nodes (Tile) according to the rent table

Here is the caller graph for this function:



4.2.3 Member Function Documentation

4.2.3.1 getNextNode()

```
int Graph::getNextNode (  
    int currentId) const
```

Function: Given the current position ID, returns the ID of the next feasible tile.

Parameters

<i>currentId</i>

Returns

4.2.3.2 getNodeName()

```
std::string Graph::getNodeName (  
    int id) const
```

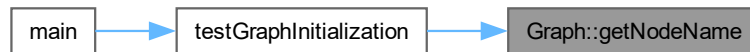
Function: Determine whether two Tiles are connected.

Parameters

<i>id</i>

Returns

Here is the caller graph for this function:



4.2.3.3 getTile()

```
std::shared_ptr< Tile > Graph::getTile (  
    int id) const
```

Function: Given a [Tile](#) ID, return a [Tile](#) object.

Parameters

<i>id</i>

Returns

Here is the caller graph for this function:



4.2.3.4 hasEdge()

```
bool Graph::hasEdge (  
    int from,  
    int to) const
```

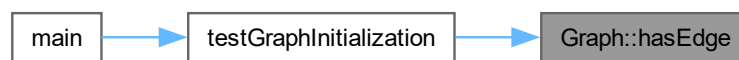
Function: Given a [Tile](#) ID, return a [Tile](#) name.

Parameters

	<i>from</i>
	<i>to</i>

Returns

Here is the caller graph for this function:



4.2.4 Member Data Documentation

4.2.4.1 adjacencyMatrix

```
std::vector<std::vector<bool> > Graph::adjacencyMatrix
```

4.2.4.2 groups

```
const std::vector<std::tuple<int, int, std::vector<int>, std::vector<int> > > Graph::groups
```

Initial value:

```
= {
    std::make_tuple(2, 60, std::vector<int>{2, 10, 30, 90, 160, 250}, std::vector<int>{50, 50}),
    std::make_tuple(3, 100, std::vector<int>{6, 30, 90, 270, 400, 550}, std::vector<int>{50, 50, 50}),
    std::make_tuple(3, 140, std::vector<int>{10, 50, 150, 450, 625, 750}, std::vector<int>{100, 100, 100}),
    std::make_tuple(3, 180, std::vector<int>{14, 70, 200, 550, 750, 950}, std::vector<int>{150, 150, 150}),
    std::make_tuple(3, 220, std::vector<int>{18, 90, 250, 700, 875, 1050}, std::vector<int>{200, 200, 200}),
    std::make_tuple(3, 260, std::vector<int>{22, 110, 330, 800, 975, 1150}, std::vector<int>{250, 250,
    250}),
    std::make_tuple(3, 300, std::vector<int>{22, 110, 330, 800, 975, 1150}, std::vector<int>{250, 250,
    250}),
    std::make_tuple(2, 400, std::vector<int>{50, 200, 600, 1400, 1700, 2000}, std::vector<int>{300, 300})
}
```

4.2.4.3 nodeNames

```
const std::string Graph::nodeNames [static]
```

4.2.4.4 PropertyTiles

```
const std::vector<int> Graph::PropertyTiles
```

Initial value:

```
= {  
    Kronenstraße,  
    Adlerstraße,  
    Ebertstraße,  
    Rüppurrerstraße,  
    Ettlingerstraße,  
    Amalienstraße,  
    Hirschstraße,  
    Kriegsstraße,  
    Fastplatz,  
    KaiserAllee,  
    DurlacherAllee,  
    Zirkel,  
    Karlstraße,  
    Brauerstraße,  
    Hildapromenade,  
    Moltkestraße,  
    Karlfriedrichstraße,  
    Herrenstraße,  
    Waldstraße,  
    Erbprinzenstraße,  
    Kaiserstraße,  
    Schlossplatz  
}
```

street

4.2.4.5 tiles

```
std::vector<std::shared_ptr<Tile> > Graph::tiles
```

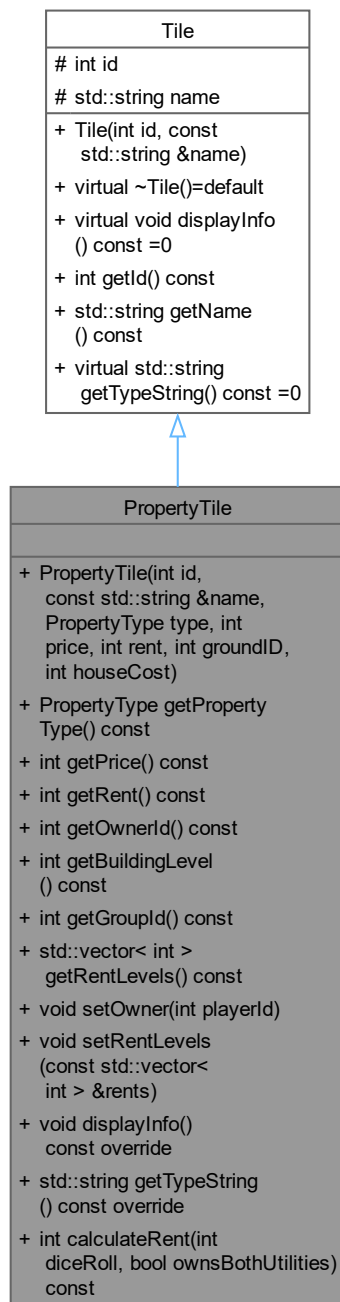
The documentation for this class was generated from the following files:

- [Graph.hpp](#)
- [Graph.cpp](#)

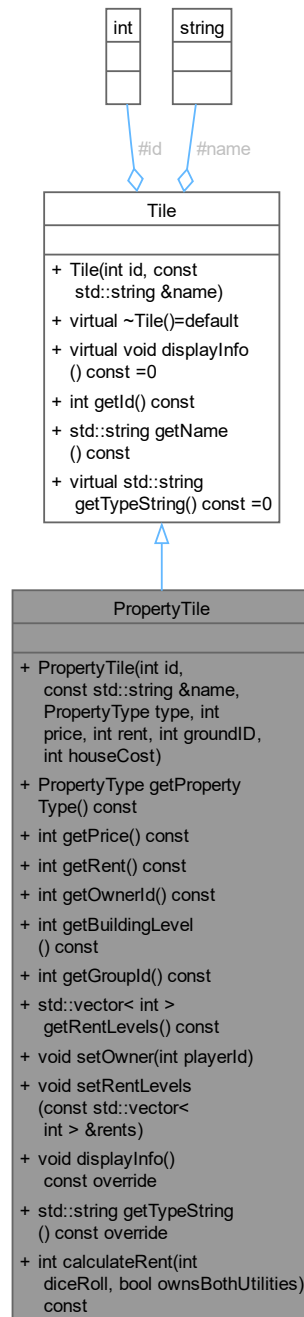
4.3 PropertyTile Class Reference

```
#include <PropertyTile.hpp>
```


Inheritance diagram for PropertyTile:



Collaboration diagram for PropertyTile:



Public Member Functions

- `PropertyTile` (int `id`, const std::string &`name`, `PropertyType` `type`, int `price`, int `rent`, int `groundID`, int `houseCost`)
- `PropertyType` `getPropertyType` () const
- int `getPrice` () const
- int `getRent` () const
- int `getOwnerId` () const

- int `getBuildingLevel` () const
- int `getGroupId` () const
- std::vector< int > `getRentLevels` () const
- void `setOwner` (int playerId)
- void `setRentLevels` (const std::vector< int > &rents)
Function: Set up the street rent table.
- void `displayInfo` () const override
HKI-9 Map: Implementierung der Straßenfelder.
- std::string `getTypeString` () const override
Function: Get street name.
- int `calculateRent` (int diceRoll, bool ownsBothUtilities) const
Function: Calculate the current rent to be paid (context requires additional information).

Public Member Functions inherited from `Tile`

- `Tile` (int id, const std::string &name)
- virtual `~Tile` ()=default
- int `getId` () const
- std::string `getName` () const

Additional Inherited Members

Protected Attributes inherited from `Tile`

- int id
- std::string name

4.3.1 Constructor & Destructor Documentation

4.3.1.1 `PropertyTile()`

```
PropertyTile::PropertyTile (
    int id,
    const std::string & name,
    PropertyType type,
    int price,
    int rent,
    int groundID,
    int houseCost) [inline]
```

Here is the call graph for this function:



4.3.2 Member Function Documentation

4.3.2.1 calculateRent()

```
int PropertyTile::calculateRent (
    int diceRoll,
    bool ownsBothUtilities) const
```

Function: Calculate the current rent to be paid (context requires additional information).

Input parameters: dice random number, whether all public facilities are owned

Parameters

	<i>diceRoll</i>
	<i>ownsBothUtilities</i>

Returns

4.3.2.2 displayInfo()

```
void PropertyTile::displayInfo () const [override], [virtual]
```

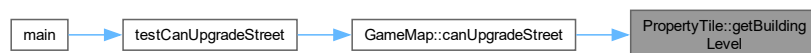
HKI-9 Map: Implementierung der Straßenfelder.

Implements [Tile](#).

4.3.2.3 getBuildingLevel()

```
int PropertyTile::getBuildingLevel () const [inline]
```

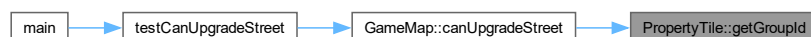
Here is the caller graph for this function:



4.3.2.4 getGroupId()

```
int PropertyTile::getGroupId () const [inline]
```

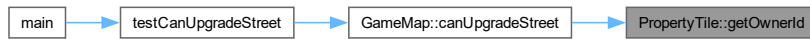
Here is the caller graph for this function:



4.3.2.5 getOwnerId()

```
int PropertyTile::getOwnerId () const [inline]
```

Here is the caller graph for this function:



4.3.2.6 getPrice()

```
int PropertyTile::getPrice () const [inline]
```

4.3.2.7 getPropertyType()

```
PropertyType PropertyTile::getPropertyType () const [inline]
```

Here is the caller graph for this function:



4.3.2.8 getRent()

```
int PropertyTile::getRent () const [inline]
```

4.3.2.9 getRentLevels()

```
std::vector< int > PropertyTile::getRentLevels () const [inline]
```

4.3.2.10 getTypeString()

```
std::string PropertyTile::getTypeString () const [override], [virtual]
```

Function: Get street name.

Returns

Implements [Tile](#).

4.3.2.11 setOwner()

```
void PropertyTile::setOwner (
    int playerId) [inline]
```

Here is the caller graph for this function:



4.3.2.12 setRentLevels()

```
void PropertyTile::setRentLevels (
    const std::vector< int > & rents)
```

Function: Set up the street rent table.

Parameters

	<i>rents</i>
--	--------------

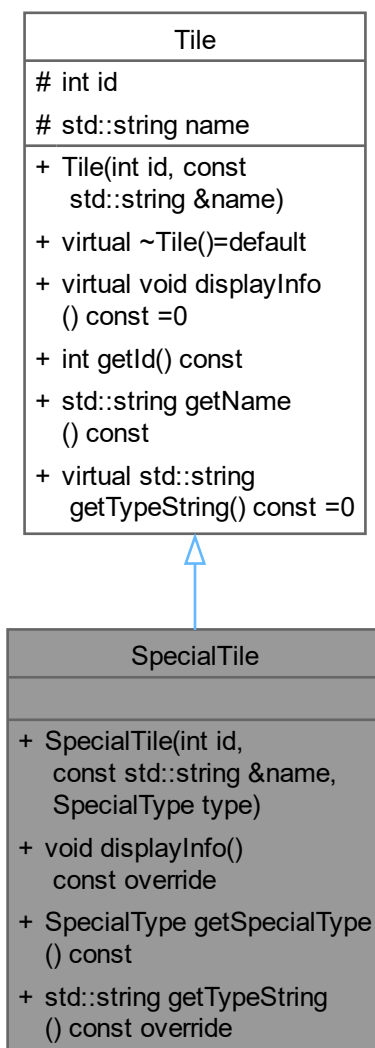
The documentation for this class was generated from the following files:

- [PropertyTile.hpp](#)
- [PropertyTile.cpp](#)

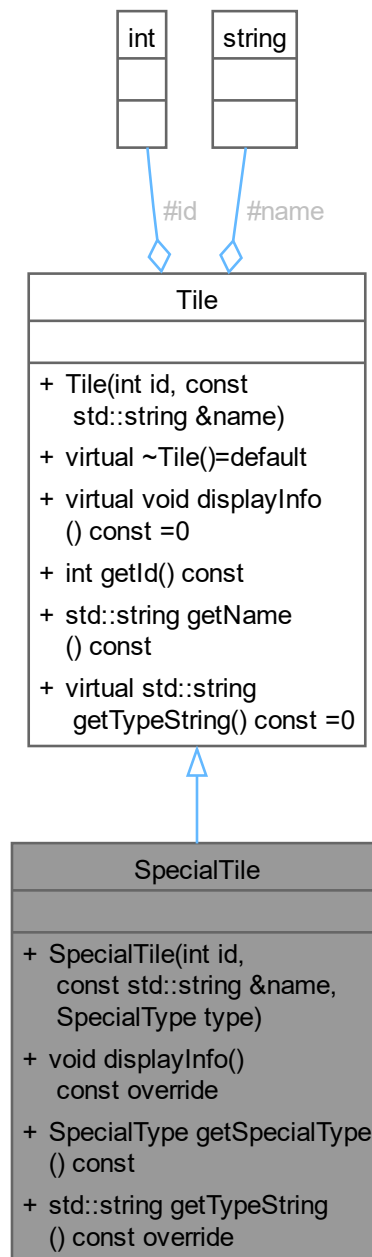
4.4 SpecialTile Class Reference

```
#include <SpecialTile.hpp>
```

Inheritance diagram for SpecialTile:



Collaboration diagram for SpecialTile:



Public Member Functions

- `SpecialTile` (int `id`, const std::string &`name`, `SpecialType` type)
- void `displayInfo` () const override
- `SpecialType` `getSpecialType` () const

//HKI-13 Map: Implementierung der Steuerfelder HKI-14 Map: Implementierung der Ereignisfelder fr Aktionskarten

- std::string `getTypeString` () const override

Public Member Functions inherited from [Tile](#)

- [Tile](#) (int [id](#), const std::string &[name](#))
- virtual [~Tile](#) ()=default
- int [getId](#) () const
- std::string [getName](#) () const

Additional Inherited Members

Protected Attributes inherited from [Tile](#)

- int [id](#)
- std::string [name](#)

4.4.1 Constructor & Destructor Documentation

4.4.1.1 SpecialTile()

```
SpecialTile::SpecialTile (  
    int id,  
    const std::string & name,  
    SpecialType type) [inline]
```

Here is the call graph for this function:



4.4.2 Member Function Documentation

4.4.2.1 displayInfo()

```
void SpecialTile::displayInfo () const [override], [virtual]
```

Implements [Tile](#).

Here is the call graph for this function:



4.4.2.2 getSpecialType()

```
SpecialType SpecialTile::getSpecialType () const [inline]
```

//HKI-13 Map: Implementierung der Steuerfelder HKI-14 Map: Implementierung der Ereignisfelder fr Aktionskarten

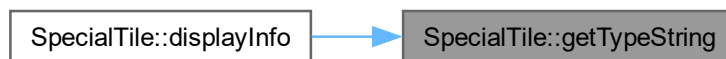
Returns

4.4.2.3 getTypeString()

```
std::string SpecialTile::getTypeString () const [override], [virtual]
```

Implements [Tile](#).

Here is the caller graph for this function:

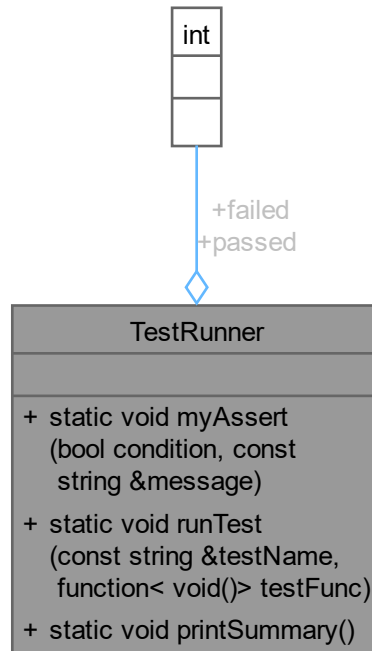


The documentation for this class was generated from the following files:

- [SpecialTile.hpp](#)
- [SpecialTile.cpp](#)

4.5 TestRunner Class Reference

Collaboration diagram for TestRunner:



Static Public Member Functions

- static void `myAssert` (bool condition, const string &message)
- static void `runTest` (const string &testName, function< void()> testFunc)
- static void `printSummary` ()

Static Public Attributes

- static int `passed` = 0
- static int `failed` = 0

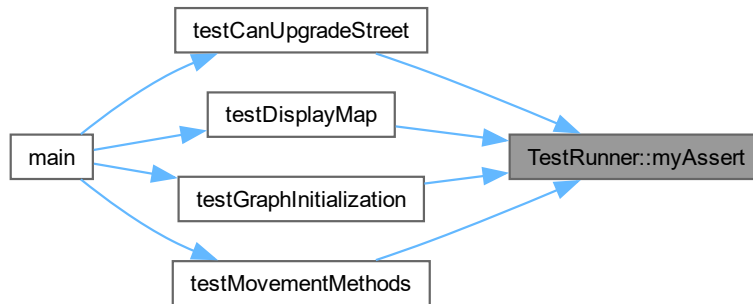
4.5.1 Member Function Documentation

4.5.1.1 `myAssert()`

```

static void TestRunner::myAssert (
    bool condition,
    const string & message) [inline], [static]
  
```

Here is the caller graph for this function:



4.5.1.2 printSummary()

```
static void TestRunner::printSummary () [inline], [static]
```

Here is the caller graph for this function:



4.5.1.3 runTest()

```
static void TestRunner::runTest (  
    const string & testName,  
    function< void()> testFunc) [inline], [static]
```

Here is the caller graph for this function:



4.5.2 Member Data Documentation

4.5.2.1 failed

```
int TestRunner::failed = 0 [static]
```

4.5.2.2 passed

```
int TestRunner::passed = 0 [static]
```

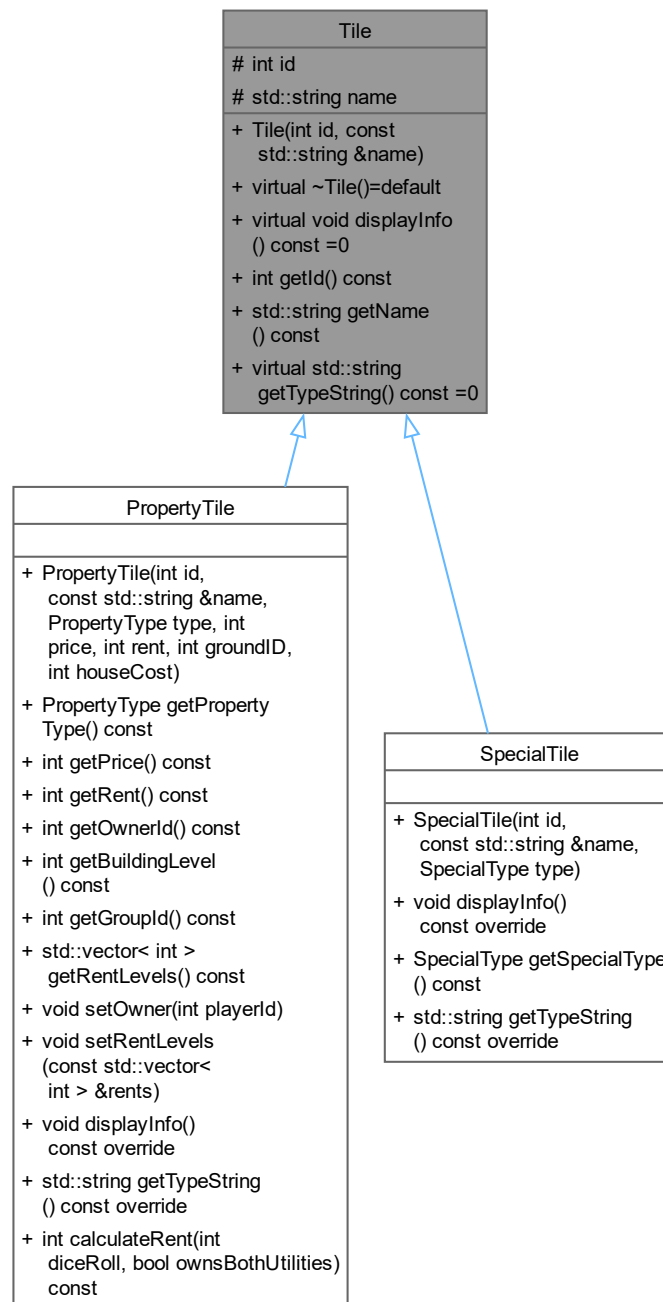
The documentation for this class was generated from the following file:

- [test.cpp](#)

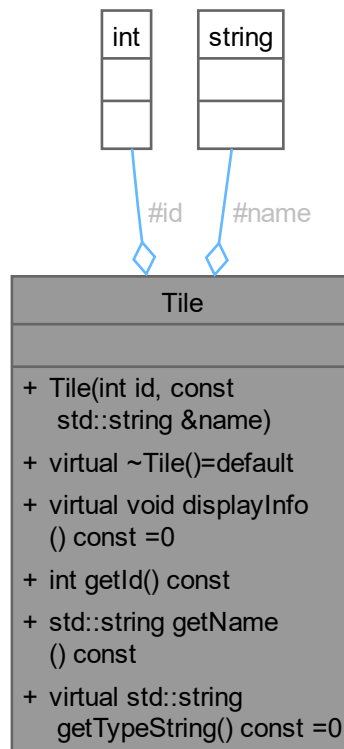
4.6 Tile Class Reference

```
#include <Tile.hpp>
```

Inheritance diagram for Tile:



Collaboration diagram for Tile:



Public Member Functions

- [Tile](#) (int [id](#), const std::string &[name](#))
- virtual [~Tile](#) ()=default
- virtual void [displayInfo](#) () const =0
- int [getId](#) () const
- std::string [getName](#) () const
- virtual std::string [getTypeString](#) () const =0

Protected Attributes

- int [id](#)
- std::string [name](#)

4.6.1 Constructor & Destructor Documentation

4.6.1.1 Tile()

```

Tile::Tile (
    int id,
    const std::string & name) [inline]
  
```

Here is the caller graph for this function:



4.6.1.2 ~Tile()

```
virtual Tile::~~Tile () [virtual], [default]
```

4.6.2 Member Function Documentation

4.6.2.1 displayInfo()

```
virtual void Tile::displayInfo () const [pure virtual]
```

Implemented in [PropertyTile](#), and [SpecialTile](#).

4.6.2.2 getId()

```
int Tile::getId () const [inline]
```

4.6.2.3 getName()

```
std::string Tile::getName () const [inline]
```

4.6.2.4 getTypeString()

```
virtual std::string Tile::getTypeString () const [pure virtual]
```

Implemented in [PropertyTile](#), and [SpecialTile](#).

4.6.3 Member Data Documentation

4.6.3.1 id

```
int Tile::id [protected]
```

4.6.3.2 name

```
std::string Tile::name [protected]
```

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

- [Tile.hpp](#)

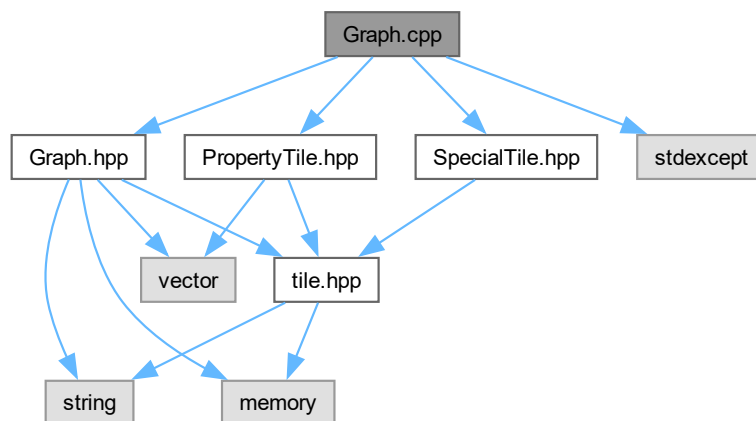
Chapter 5

File Documentation

5.1 Graph.cpp File Reference

```
#include "Graph.hpp"  
#include "PropertyTile.hpp"  
#include "SpecialTile.hpp"  
#include <stdexcept>
```

Include dependency graph for Graph.cpp:

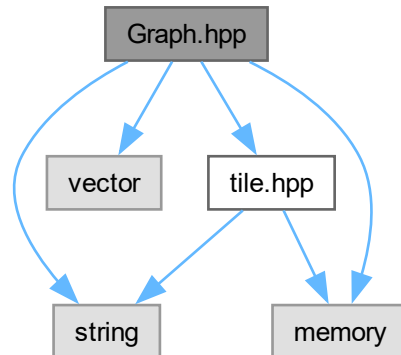


5.2 Graph.hpp File Reference

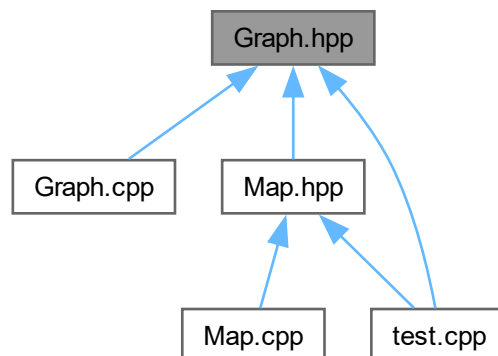
```
#include <string>  
#include <vector>  
#include <memory>
```

```
#include "tile.hpp"
```

Include dependency graph for Graph.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class [Graph](#)

This is the [Graph](#) class.

Macros

- #define [TOTAL_NODES](#) 40
Define node ID using macro.
- #define [LOS](#) 0

- #define [Kronenstraße](#) 1
- #define [Gemeinschaftsfeld](#) 2
- #define [Adlerstraße](#) 3
- #define [Einkommensteuer](#) 4
- #define [Hauptbahnhof](#) 5
- #define [Ebertstraße](#) 6
- #define [Ereignisfeld](#) 7
- #define [Rüppurrerstraße](#) 8
- #define [Ettlingerstraße](#) 9
- #define [Gefängnis](#) 10
- #define [Amalienstraße](#) 11
- #define [Elektrizitätswerk](#) 12
- #define [Hirschstraße](#) 13
- #define [Kriegsstraße](#) 14
- #define [WestBahnhof](#) 15
- #define [Fastplatz](#) 16
- #define [Gemeinschaftsfeld2](#) 17
- #define [KaiserAllee](#) 18
- #define [DurlacherAllee](#) 19
- #define [FreiParken](#) 20
- #define [Zirkel](#) 21
- #define [Gemeinschaftsfeld3](#) 22
- #define [Karlstraße](#) 23
- #define [Brauerstraße](#) 24
- #define [OstBahnhof](#) 25
- #define [Hildapromenade](#) 26
- #define [Moltkestraße](#) 27
- #define [Wasserwerk](#) 28
- #define [Karlfriedrichstraße](#) 29
- #define [GeheinsGefängnis](#) 30
- #define [Herrenstraße](#) 31
- #define [Waldstraße](#) 32
- #define [Gemeinschaftsfeld4](#) 33
- #define [Erbprinzenstraße](#) 34
- #define [Hubschrauberlandeplatze](#) 35
- #define [Ereignisfeld2](#) 36
- #define [Kaiserstraße](#) 37
- #define [Zusatzsteuer](#) 38
- #define [Schlossplatz](#) 39

5.2.1 Macro Definition Documentation

5.2.1.1 Adlerstraße

```
#define Adlerstraße 3
```

5.2.1.2 Amalienstraße

```
#define Amalienstraße 11
```

5.2.1.3 Brauerstraße

```
#define Brauerstraße 24
```

5.2.1.4 DurlacherAllee

```
#define DurlacherAllee 19
```

5.2.1.5 Ebertstraße

```
#define Ebertstraße 6
```

5.2.1.6 Einkommensteuer

```
#define Einkommensteuer 4
```

5.2.1.7 Elektrizitätwerk

```
#define Elektrizitätwerk 12
```

5.2.1.8 Erbprinzenstraße

```
#define Erbprinzenstraße 34
```

5.2.1.9 Ereignisfeld

```
#define Ereignisfeld 7
```

5.2.1.10 Ereignisfeld2

```
#define Ereignisfeld2 36
```

5.2.1.11 Ettlingerstraße

```
#define Ettlingerstraße 9
```

5.2.1.12 Fastplatz

```
#define Fastplatz 16
```

5.2.1.13 FreiParken

```
#define FreiParken 20
```

5.2.1.14 Gefängnis

```
#define Gefängnis 10
```

5.2.1.15 GeheinsGefängnis

```
#define GeheinsGefängnis 30
```

5.2.1.16 Gemeinschaftsfeld

```
#define Gemeinschaftsfeld 2
```

5.2.1.17 Gemeinschaftsfeld2

```
#define Gemeinschaftsfeld2 17
```

5.2.1.18 Gemeinschaftsfeld3

```
#define Gemeinschaftsfeld3 22
```

5.2.1.19 Gemeinschaftsfeld4

```
#define Gemeinschaftsfeld4 33
```

5.2.1.20 Hauptbahnhof

```
#define Hauptbahnhof 5
```

5.2.1.21 Herrenstraße

```
#define Herrenstraße 31
```

5.2.1.22 Hildapromenade

```
#define Hildapromenade 26
```

5.2.1.23 Hirschstraße

```
#define Hirschstraße 13
```

5.2.1.24 Hubschrauberlandeplätze

```
#define Hubschrauberlandeplätze 35
```

5.2.1.25 KaiserAllee

```
#define KaiserAllee 18
```

5.2.1.26 Kaiserstraße

```
#define Kaiserstraße 37
```

5.2.1.27 Karlfriedrichstraße

```
#define Karlfriedrichstraße 29
```

5.2.1.28 Karlstraße

```
#define Karlstraße 23
```

5.2.1.29 Kriegsstraße

```
#define Kriegsstraße 14
```

5.2.1.30 Kronenstraße

```
#define Kronenstraße 1
```

5.2.1.31 LOS

```
#define LOS 0
```

5.2.1.32 Moltkestraße

```
#define Moltkestraße 27
```

5.2.1.33 OstBahnhof

```
#define OstBahnhof 25
```

5.2.1.34 Rüppurrerstraße

```
#define Rüppurrerstraße 8
```

5.2.1.35 Schlossplatz

```
#define Schlossplatz 39
```

5.2.1.36 TOTAL_NODES

```
#define TOTAL_NODES 40
```

Define node ID using macro.

5.2.1.37 Waldstraße

```
#define Waldstraße 32
```

5.2.1.38 Wasserwerk

```
#define Wasserwerk 28
```

5.2.1.39 WestBahnhof

```
#define WestBahnhof 15
```

5.2.1.40 Zirkel

```
#define Zirkel 21
```

5.2.1.41 Zusatzsteuer

```
#define Zusatzsteuer 38
```

5.3 Graph.hpp

[Go to the documentation of this file.](#)

```

00001
00004 #ifndef GRAPH_HPP
00005 #define GRAPH_HPP
00006 #include <string>
00007 #include <vector>
00008 #include <memory>
00009 #include "tile.hpp"
00010 class Graph {
00011 public:
00015     #define TOTAL_NODES 40
00016     #define LOS 0
00017     #define Kronenstraße 1
00018     #define Gemeinschaftsfeld 2
00019     #define Adlerstraße 3
00020     #define Einkommensteuer 4
00021     #define Hauptbahnhof 5
00022     #define Ebertstraße 6
00023     #define Ereignisfeld 7
00024     #define Rüppurrerstraße 8
00025     #define Ettlingerstraße 9
00026     #define Gefängnis 10
00027     #define Amalienstraße 11
00028     #define Elektrizitätswerk 12
00029     #define Hirschstraße 13
00030     #define Kriegsstraße 14
00031     #define WestBahnhof 15
00032     #define Fastplatz 16
00033     #define Gemeinschaftsfeld2 17
00034     #define KaiserAllee 18
00035     #define DurlacherAllee 19
00036     #define FreiParken 20
00037     #define Zirkel 21
00038     #define Gemeinschaftsfeld3 22
00039     #define Karlstraße 23
00040     #define Brauerstraße 24
00041     #define OstBahnhof 25
00042     #define Hildapromenade 26
00043     #define Moltkestraße 27
00044     #define Wasserwerk 28
00045     #define Karlfriedrichstraße 29
00046     #define GeheinsGefängnis 30
00047     #define Herrenstraße 31
00048     #define Waldstraße 32
00049     #define Gemeinschaftsfeld4 33
00050     #define Erbprinzenstraße 34
00051     #define Hubschrauberlandeplatze 35
00052     #define Ereignisfeld2 36
00053     #define Kaiserstraße 37
00054     #define Zusatzsteuer 38
00055     #define Schlossplatz 39
00056
00057
00058     static const std::string nodeNames[TOTAL_NODES];
00059     std::vector<std::shared_ptr<Tile> > tiles;
00060     const std::vector<std::tuple<int, int, std::vector<int>, std::vector<int>> > groups= {
00061         std::make_tuple(2, 60, std::vector<int>{2, 10, 30, 90, 160, 250}, std::vector<int>{50, 50}),
00062         std::make_tuple(3, 100, std::vector<int>{6, 30, 90, 270, 400, 550}, std::vector<int>{50, 50, 50}),
00063         std::make_tuple(3, 140, std::vector<int>{10, 50, 150, 450, 625, 750}, std::vector<int>{100, 100,
00064             100}),
00065         std::make_tuple(3, 180, std::vector<int>{14, 70, 200, 550, 750, 950}, std::vector<int>{150, 150,
00066             150}),
00067         std::make_tuple(3, 220, std::vector<int>{18, 90, 250, 700, 875, 1050}, std::vector<int>{200, 200,
00068             200}),
00069         std::make_tuple(3, 260, std::vector<int>{22, 110, 330, 800, 975, 1150}, std::vector<int>{250, 250,
00070             250}),
00071         std::make_tuple(3, 300, std::vector<int>{22, 110, 330, 800, 975, 1150}, std::vector<int>{250, 250,
00072             250}),
00073         std::make_tuple(2, 400, std::vector<int>{50, 200, 600, 1400, 1700, 2000}, std::vector<int>{300,
00074             300})
00075     };
00076     const std::vector<int> PropertyTiles = {
00077         Kronenstraße,
00078         Adlerstraße,
00079         Ebertstraße,
00080         Rüppurrerstraße,
00081         Ettlingerstraße,
00082         Amalienstraße,
00083         Hirschstraße,
00084         Kriegsstraße,
00085         Fastplatz,
00086         KaiserAllee,
00087         DurlacherAllee,

```



```

00085     Zirkel,
00086     Karlstraße,
00087     Brauerstraße,
00088     Hildapromenade,
00089     Moltkestraße,
00090     Karlfriedrichstraße,
00091     Herrenstraße,
00092     Waldstraße,
00093     Erbprinzenstraße,
00094     Kaiserstraße,
00095     Schlossplatz
00096 };
00097
00098
00099     std::vector<std::vector<bool> > adjacencyMatrix;
00100
00101     Graph();
00102
00103     int getNextNode(int currentId) const;
00104     std::shared_ptr<Tile> getTile(int id) const;
00105
00106     bool hasEdge(int from, int to) const;
00107
00108     std::string getNodeName(int id) const;
00109 };
00110 #endif // GRAPH_HPP

```

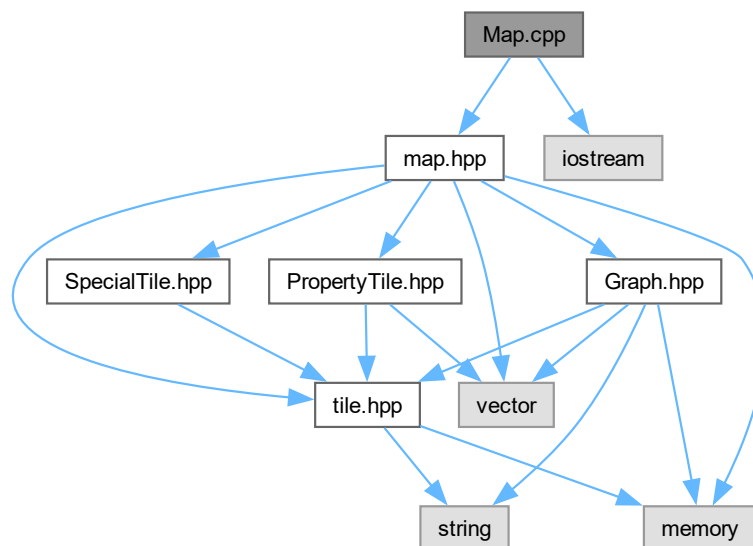
5.4 Map.cpp File Reference

```

#include "map.hpp"
#include <iostream>

```

Include dependency graph for Map.cpp:



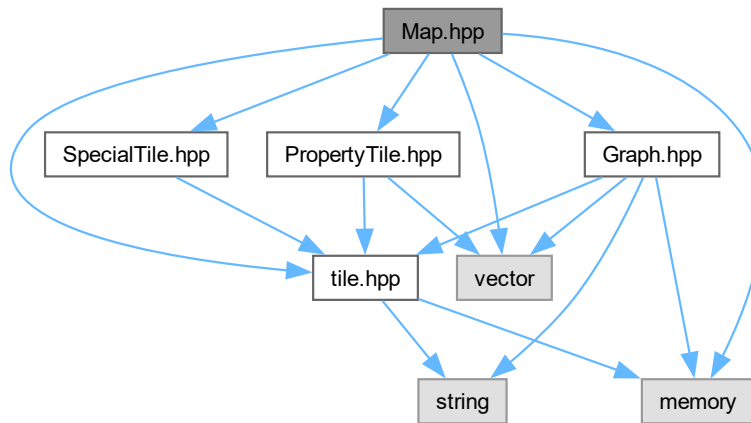
5.5 Map.hpp File Reference

```

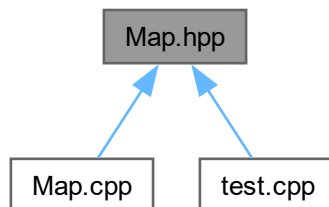
#include "tile.hpp"
#include "PropertyTile.hpp"

```

```
#include "SpecialTile.hpp"
#include <vector>
#include <memory>
#include "Graph.hpp"
Include dependency graph for Map.hpp:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [GameMap](#)

This is the implementation of the Map class.

5.6 Map.hpp

[Go to the documentation of this file.](#)

```
00001
00005 #ifndef MAP_HPP
```

```

00006 #define MAP_HPP
00007
00008 #include "tile.hpp"
00009 #include "PropertyTile.hpp"
00010 #include "SpecialTile.hpp"
00011 #include <vector>
00012 #include <memory>
00013 #include "Graph.hpp"
00014
00015 class GameMap:public Graph {
00016 public:
00017
00018     GameMap();
00022     void displayMap() const;
00029     bool canUpgradeStreet(PropertyTile* targetStreet, int playerId) const;
00034     int movebahn() const;
00039     int moveHub() const;
00040 private:
00045     void printTileInfo(int nodeId) const;
00046 };
00047
00048 #endif

```

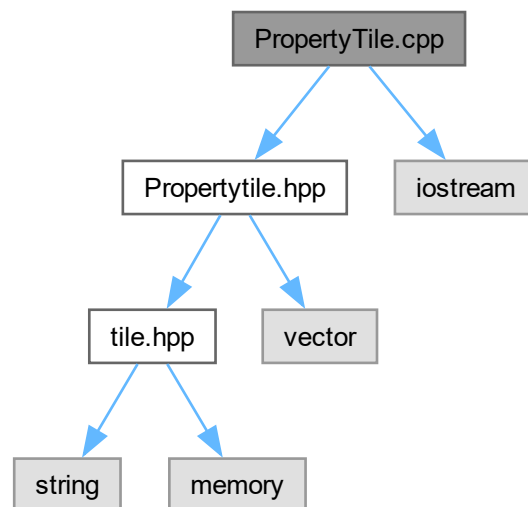
5.7 PropertyTile.cpp File Reference

```

#include "Propertytile.hpp"
#include <iostream>

```

Include dependency graph for PropertyTile.cpp:



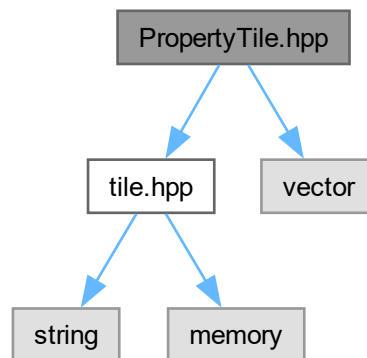
5.8 PropertyTile.hpp File Reference

```

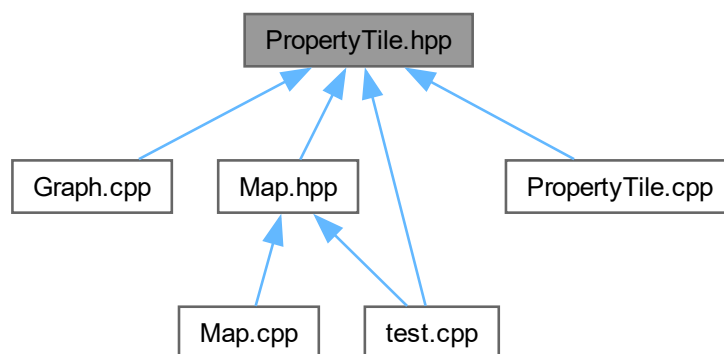
#include "tile.hpp"
#include "vector"

```

Include dependency graph for PropertyTile.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class [PropertyTile](#)

Enumerations

- enum class [PropertyType](#) { [Street](#) , [Utility](#) }

This is the implementation of the [PropertyTile](#) class.

5.8.1 Enumeration Type Documentation

5.8.1.1 PropertyType

```
enum class PropertyType [strong]
```

This is the implementation of the [PropertyTile](#) class.

Contains street information display, rent setting and calculation

Enumerator

	Street
	Utility

5.9 PropertyTile.hpp

[Go to the documentation of this file.](#)

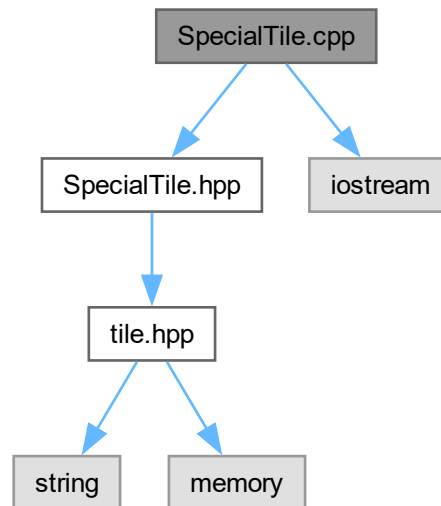
```
00001
00004 #ifndef PROPERTY_TILE_HPP
00005 #define PROPERTY_TILE_HPP
00006
00007 #include "tile.hpp"
00008 #include "vector"
00009
00010 enum class PropertyType { Street, Utility };
00011
00012 class PropertyTile : public Tile {
00013 private:
00014     PropertyType propertyType;
00015     int price;
00016     int rent;
00017     int ownerId;
00018
00019     int groudID;
00020     int buildingLevel;
00021     int houseCost;
00022     std::vector<int> rentLevels;
00023 public:
00024     //PropertyTile(int id, const std::string& name, PropertyType type, int price): Tile(id, name),
00025     //propertyType(type), price(price){}
00026     PropertyTile(int id, const std::string& name, PropertyType type, int price, int rent, int groundID,
00027                 int houseCost)
00028         : Tile(id, name), propertyType(type), price(price), rent(rent), ownerId(-1), rentLevels({10,
00029         50, 150, 450, 625, 750}), houseCost(100),
00030         buildingLevel(0), groudID(-1) {}
00031 // Getter
00032     PropertyType getPropertyType() const { return propertyType; }
00033     int getPrice() const { return price; }
00034     int getRent() const { return rent; }
00035     int getOwnerId() const { return ownerId; }
00036     int getBuildingLevel() const { return buildingLevel; }
00037     int getGroupID() const { return groudID; }
00038     std::vector<int> getRentLevels() const { return rentLevels; }
00039 // Setter
00040     void setOwner(int playerId) { ownerId = playerId; }
00041     void setRentLevels(const std::vector<int>& rents);
00042     void displayInfo() const override;
00043     std::string getTypeString() const override;
00044     int calculateRent(int diceRoll, bool ownsBothUtilities) const;
00045 };
00046
00047 #endif
```

5.10 SpecialTile.cpp File Reference

```
#include "SpecialTile.hpp"
```

```
#include <iostream>
```

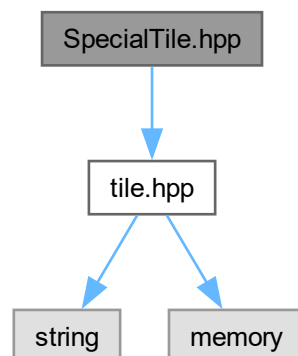
Include dependency graph for SpecialTile.cpp:



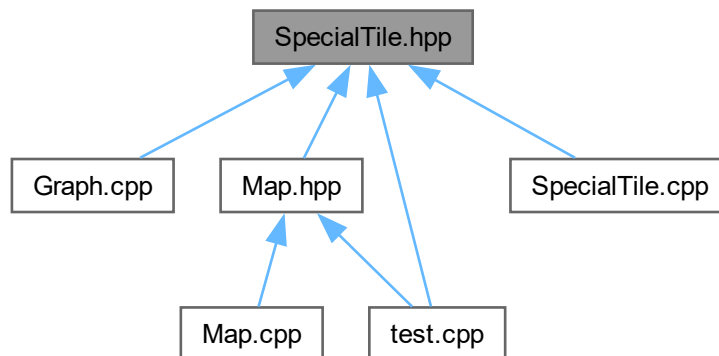
5.11 SpecialTile.hpp File Reference

```
#include "tile.hpp"
```

Include dependency graph for SpecialTile.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class [SpecialTile](#)

Enumerations

- enum class [SpecialType](#) {
[Start](#) , [Event](#) , [Community](#) , [Bahnhof](#) ,
[Tax](#) , [LuxuryTax](#) , [Jail](#) , [GoToJail](#) ,
[FreeParking](#) , [Hubschrauberlandeplatz](#) }

This is the implementation of the [SpecialTile](#) class. Provide special tile types.

5.11.1 Enumeration Type Documentation

5.11.1.1 SpecialType

```
enum class SpecialType [strong]
```

This is the implementation of the [SpecialTile](#) class. Provide special tile types.

Enumerator

	Start
	Event
	Community
	Bahnhof
	Tax
	LuxuryTax
	Jail
	GoToJail
	FreeParking
	Hubschrauberlandeplatz

5.12 SpecialTile.hpp

[Go to the documentation of this file.](#)

```

00001
00004 #ifndef SPECIAL_TILE_HPP
00005 #define SPECIAL_TILE_HPP
00006
00007 #include "tile.hpp"
00008
00009 enum class SpecialType { Start, Event, Community, Bahnhof, Tax, LuxuryTax, Jail, GoToJail, FreeParking,
00010                           Hubschrauberlandeplatz};
00011
00011 class SpecialTile : public Tile {
00012 private:
00013     SpecialType specialType;
00014
00015 public:
00016     SpecialTile(int id, const std::string& name, SpecialType type)
00017         : Tile(id, name), specialType(type) {}
00018
00019     void displayInfo() const override;
00024     SpecialType getSpecialType() const { return specialType; }
00025     std::string getTypeString() const override;
00026 };
00027
00028 #endif

```

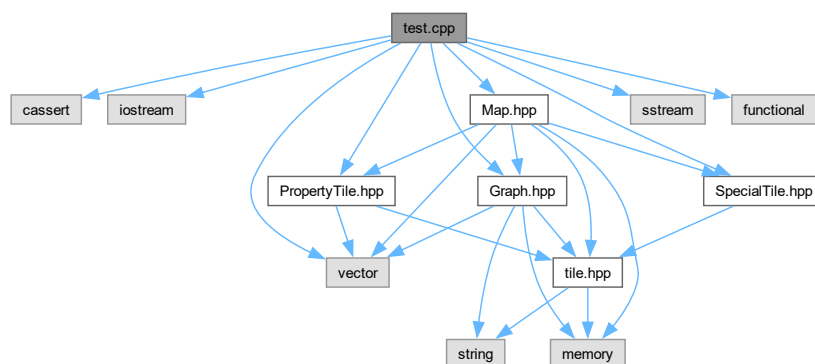
5.13 test.cpp File Reference

```

#include <cassert>
#include <iostream>
#include <vector>
#include <sstream>
#include "PropertyTile.hpp"
#include "SpecialTile.hpp"
#include "Map.hpp"
#include "Graph.hpp"
#include <functional>

```

Include dependency graph for test.cpp:



Classes

- class [TestRunner](#)

Functions

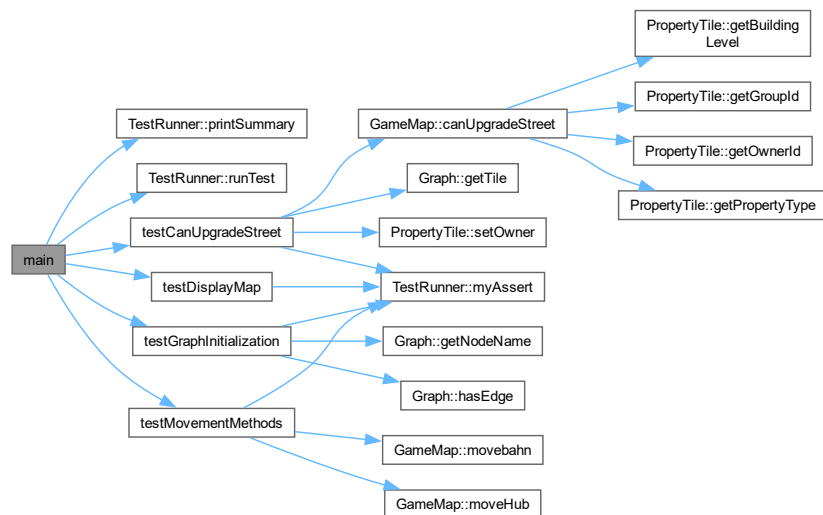
- void [testGraphInitialization](#) ()
- void [testDisplayMap](#) ()
- void [testCanUpgradeStreet](#) ()
- void [testMovementMethods](#) ()
- int [main](#) ()

5.13.1 Function Documentation

5.13.1.1 main()

```
int main ()
```

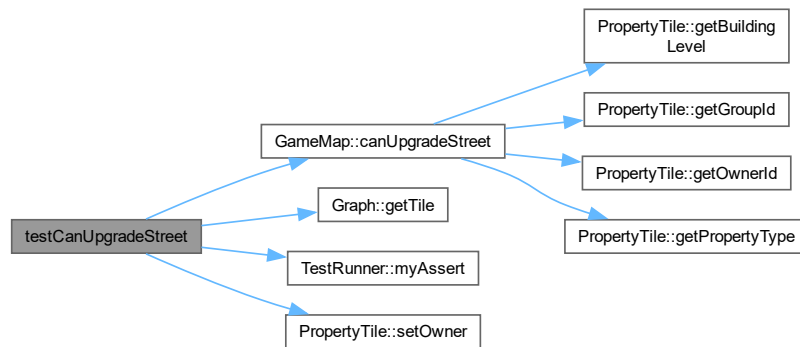
Here is the call graph for this function:



5.13.1.2 testCanUpgradeStreet()

```
void testCanUpgradeStreet ()
```

Here is the call graph for this function:



Here is the caller graph for this function:



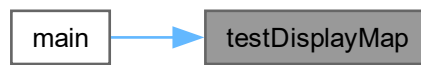
5.13.1.3 testDisplayMap()

```
void testDisplayMap ()
```

Here is the call graph for this function:



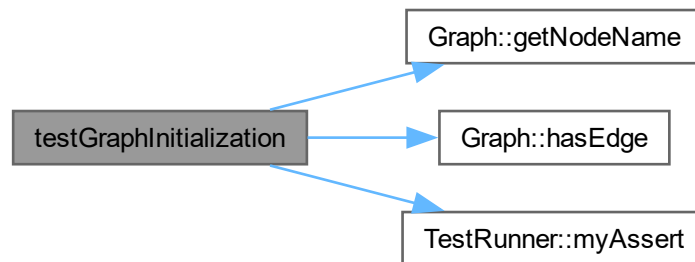
Here is the caller graph for this function:



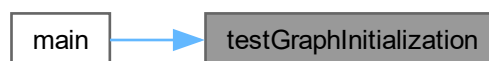
5.13.1.4 testGraphInitialization()

```
void testGraphInitialization ()
```

Here is the call graph for this function:



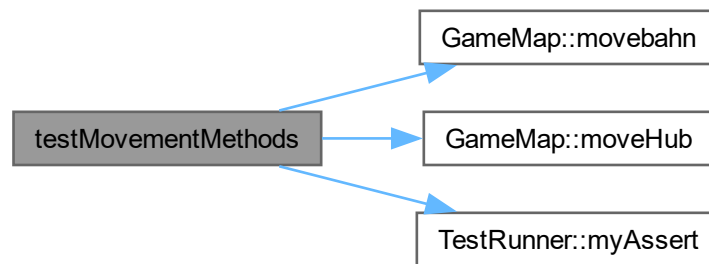
Here is the caller graph for this function:



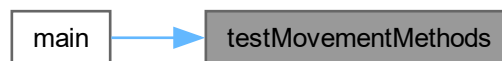
5.13.1.5 testMovementMethods()

```
void testMovementMethods ()
```

Here is the call graph for this function:



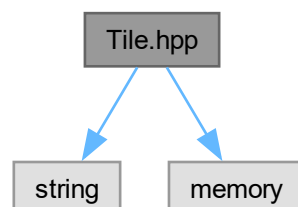
Here is the caller graph for this function:



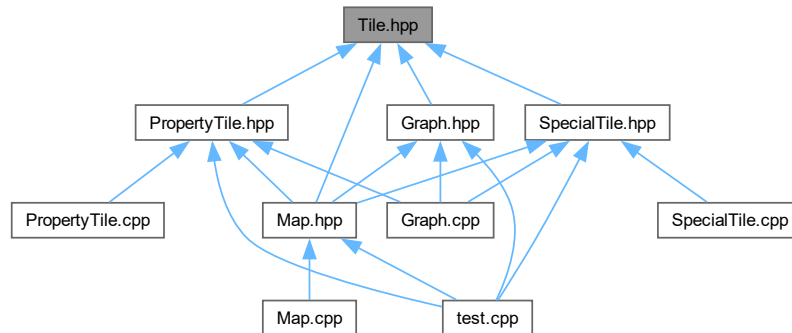
5.14 Tile.hpp File Reference

```
#include <string>
#include <memory>
```

Include dependency graph for `Tile.hpp`:



This graph shows which files directly or indirectly include this file:



Classes

- class [Tile](#)

5.15 Tile.hpp

[Go to the documentation of this file.](#)

```

00001 //Abstract class, pure virtual function definition Tile
00002 #ifndef TILE_HPP
00003 #define TILE_HPP
00004
00005 #include <string>
00006 #include <memory>
00007
00008 class Tile {
00009 protected:
00010     int id;
00011     std::string name;
00012
00013 public:
00014     Tile(int id, const std::string& name) : id(id), name(name) {}
00015     virtual ~Tile() = default;
00016
00017     virtual void displayInfo() const = 0;
00018
00019     int getId() const { return id; }
00020     std::string getName() const { return name; }
00021     virtual std::string getTypeString() const=0;
00022 };
00023
00024 #endif
  
```


Index

- ~Tile
 - Tile, [34](#)
- adjacencyMatrix
 - Graph, [17](#)
- Adlerstraße
 - Graph.hpp, [37](#)
- Amalienstraße
 - Graph.hpp, [37](#)
- Bahnhof
 - SpecialTile.hpp, [49](#)
- Brauerstraße
 - Graph.hpp, [37](#)
- calculateRent
 - PropertyTile, [22](#)
- canUpgradeStreet
 - GameMap, [10](#)
- Community
 - SpecialTile.hpp, [49](#)
- displayInfo
 - PropertyTile, [22](#)
 - SpecialTile, [27](#)
 - Tile, [34](#)
- displayMap
 - GameMap, [11](#)
- DurlacherAllee
 - Graph.hpp, [38](#)
- Ebertstraße
 - Graph.hpp, [38](#)
- Einkommensteuer
 - Graph.hpp, [38](#)
- Elektrizitätswerk
 - Graph.hpp, [38](#)
- Erbprinzenstraße
 - Graph.hpp, [38](#)
- Ereignisfeld
 - Graph.hpp, [38](#)
- Ereignisfeld2
 - Graph.hpp, [38](#)
- Ettlingerstraße
 - Graph.hpp, [38](#)
- Event
 - SpecialTile.hpp, [49](#)
- failed
 - TestRunner, [31](#)
- Fastplatz
 - Graph.hpp, [38](#)
- FreeParking
 - SpecialTile.hpp, [49](#)
- FreiParken
 - Graph.hpp, [38](#)
- GameMap, [7](#)
 - canUpgradeStreet, [10](#)
 - displayMap, [11](#)
 - GameMap, [10](#)
 - movebahn, [11](#)
 - moveHub, [12](#)
- Gefängnis
 - Graph.hpp, [39](#)
- GeheinsGefängnis
 - Graph.hpp, [39](#)
- Gemeinschaftsfeld
 - Graph.hpp, [39](#)
- Gemeinschaftsfeld2
 - Graph.hpp, [39](#)
- Gemeinschaftsfeld3
 - Graph.hpp, [39](#)
- Gemeinschaftsfeld4
 - Graph.hpp, [39](#)
- getBuildingLevel
 - PropertyTile, [22](#)
- getGroupId
 - PropertyTile, [22](#)
- getId
 - Tile, [34](#)
- getName
 - Tile, [34](#)
- getNextNode
 - Graph, [15](#)
- getNodeName
 - Graph, [15](#)
- getOwnerId
 - PropertyTile, [22](#)
- getPrice
 - PropertyTile, [23](#)
- getPropertyType
 - PropertyTile, [23](#)
- getRent
 - PropertyTile, [23](#)
- getRentLevels
 - PropertyTile, [23](#)
- getSpecialType
 - SpecialTile, [27](#)
- getTile
 - Graph, [16](#)

- getTypeString
 - PropertyTile, 23
 - SpecialTile, 28
 - Tile, 34
- GoToJail
 - SpecialTile.hpp, 49
- Graph, 13
 - adjacencyMatrix, 17
 - getNextNode, 15
 - getNodeName, 15
 - getTile, 16
 - Graph, 15
 - groups, 17
 - hasEdge, 16
 - nodeNames, 17
 - PropertyTiles, 17
 - tiles, 18
- Graph.cpp, 35
- Graph.hpp, 35
 - Adlerstraße, 37
 - Amalienstraße, 37
 - Brauerstraße, 37
 - DurlacherAllee, 38
 - Ebertstraße, 38
 - Einkommensteuer, 38
 - Elektrizitätswerk, 38
 - Erbprinzenstraße, 38
 - Ereignisfeld, 38
 - Ereignisfeld2, 38
 - Ettlingerstraße, 38
 - Fastplatz, 38
 - FreiParken, 38
 - Gefängnis, 39
 - GeheinsGefängnis, 39
 - Gemeinschaftsfeld, 39
 - Gemeinschaftsfeld2, 39
 - Gemeinschaftsfeld3, 39
 - Gemeinschaftsfeld4, 39
 - Hauptbahnhof, 39
 - Herrenstraße, 39
 - Hildapromenade, 39
 - Hirschstraße, 39
 - Hubschrauberlandeplätze, 40
 - KaiserAllee, 40
 - Kaiserstraße, 40
 - Karlfriedrichstraße, 40
 - Karlstraße, 40
 - Kriegsstraße, 40
 - Kronenstraße, 40
 - LOS, 40
 - Moltkestraße, 40
 - OstBahnhof, 40
 - Rüppurrerstraße, 41
 - Schlossplatz, 41
 - TOTAL_NODES, 41
 - Waldstraße, 41
 - Wasserwerk, 41
 - WestBahnhof, 41
 - Zirkel, 41
 - Zusatzsteuer, 41
- groups
 - Graph, 17
- hasEdge
 - Graph, 16
- Hauptbahnhof
 - Graph.hpp, 39
- Herrenstraße
 - Graph.hpp, 39
- Hildapromenade
 - Graph.hpp, 39
- Hirschstraße
 - Graph.hpp, 39
- Hubschrauberlandeplatz
 - SpecialTile.hpp, 49
- Hubschrauberlandeplätze
 - Graph.hpp, 40
- id
 - Tile, 34
- Jail
 - SpecialTile.hpp, 49
- KaiserAllee
 - Graph.hpp, 40
- Kaiserstraße
 - Graph.hpp, 40
- Karlfriedrichstraße
 - Graph.hpp, 40
- Karlstraße
 - Graph.hpp, 40
- Kriegsstraße
 - Graph.hpp, 40
- Kronenstraße
 - Graph.hpp, 40
- LOS
 - Graph.hpp, 40
- LuxuryTax
 - SpecialTile.hpp, 49
- main
 - test.cpp, 51
- Map.cpp, 43
- Map.hpp, 43
- Moltkestraße
 - Graph.hpp, 40
- movebahn
 - GameMap, 11
- moveHub
 - GameMap, 12
- myAssert
 - TestRunner, 29
- name
 - Tile, 34
- nodeNames

- Graph, [17](#)
- OstBahnhof
 - Graph.hpp, [40](#)
- passed
 - TestRunner, [31](#)
- printSummary
 - TestRunner, [30](#)
- PropertyTile, [18](#)
 - calculateRent, [22](#)
 - displayInfo, [22](#)
 - getBuildingLevel, [22](#)
 - getGroupId, [22](#)
 - getOwnerId, [22](#)
 - getPrice, [23](#)
 - getPropertyType, [23](#)
 - getRent, [23](#)
 - getRentLevels, [23](#)
 - getTypeString, [23](#)
 - PropertyTile, [21](#)
 - setOwner, [23](#)
 - setRentLevels, [24](#)
- PropertyTile.cpp, [45](#)
- PropertyTile.hpp, [45](#)
 - PropertyType, [47](#)
 - Street, [47](#)
 - Utility, [47](#)
- PropertyTiles
 - Graph, [17](#)
- PropertyType
 - PropertyTile.hpp, [47](#)
- runTest
 - TestRunner, [30](#)
- Rüppurrerstraße
 - Graph.hpp, [41](#)
- Schlossplatz
 - Graph.hpp, [41](#)
- setOwner
 - PropertyTile, [23](#)
- setRentLevels
 - PropertyTile, [24](#)
- SpecialTile, [24](#)
 - displayInfo, [27](#)
 - getSpecialType, [27](#)
 - getTypeString, [28](#)
 - SpecialTile, [27](#)
- SpecialTile.cpp, [48](#)
- SpecialTile.hpp, [48](#)
 - Bahnhof, [49](#)
 - Community, [49](#)
 - Event, [49](#)
 - FreeParking, [49](#)
 - GoToJail, [49](#)
 - Hubschrauberlandeplatz, [49](#)
 - Jail, [49](#)
 - LuxuryTax, [49](#)
 - SpecialType, [49](#)
 - Start, [49](#)
 - Tax, [49](#)
- SpecialType
 - SpecialTile.hpp, [49](#)
- Start
 - SpecialTile.hpp, [49](#)
- Street
 - PropertyTile.hpp, [47](#)
- Tax
 - SpecialTile.hpp, [49](#)
- test.cpp, [50](#)
 - main, [51](#)
 - testCanUpgradeStreet, [51](#)
 - testDisplayMap, [52](#)
 - testGraphInitialization, [53](#)
 - testMovementMethods, [53](#)
- testCanUpgradeStreet
 - test.cpp, [51](#)
- testDisplayMap
 - test.cpp, [52](#)
- testGraphInitialization
 - test.cpp, [53](#)
- testMovementMethods
 - test.cpp, [53](#)
- TestRunner, [29](#)
 - failed, [31](#)
 - myAssert, [29](#)
 - passed, [31](#)
 - printSummary, [30](#)
 - runTest, [30](#)
- Tile, [31](#)
 - ~Tile, [34](#)
 - displayInfo, [34](#)
 - getId, [34](#)
 - getName, [34](#)
 - getTypeString, [34](#)
 - id, [34](#)
 - name, [34](#)
 - Tile, [33](#)
- Tile.hpp, [54](#)
- tiles
 - Graph, [18](#)
- TOTAL_NODES
 - Graph.hpp, [41](#)
- Utility
 - PropertyTile.hpp, [47](#)
- Waldstraße
 - Graph.hpp, [41](#)
- Wasserwerk
 - Graph.hpp, [41](#)
- WestBahnhof
 - Graph.hpp, [41](#)
- Zirkel
 - Graph.hpp, [41](#)

Zusatzsteuer
Graph.hpp, [41](#)