My Project

Generated by Doxygen 1.13.2

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 files with brief descriptions:

Chapter 4

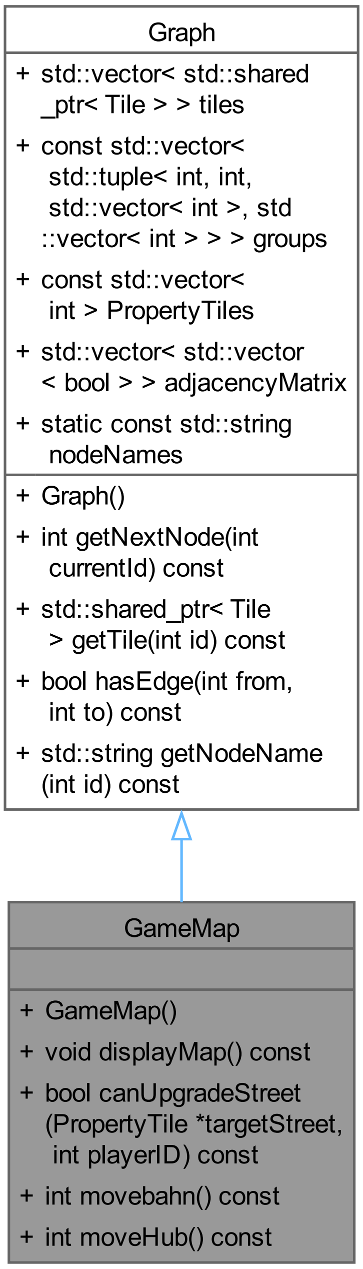
Class Documentation

4.1 GameMap Class Reference

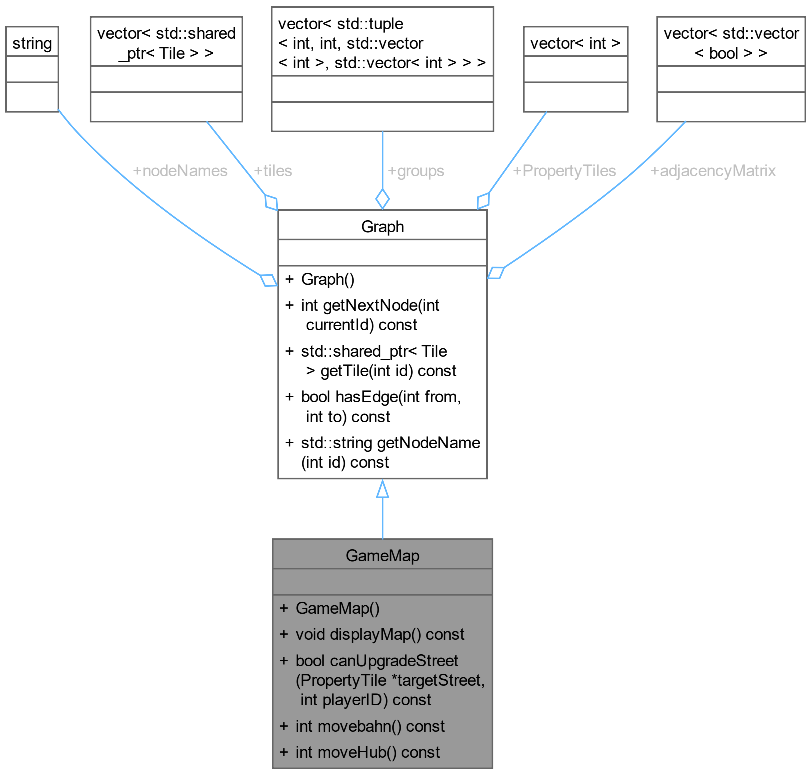
This is the implementation of the Map class.

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

Inheritance diagram for GameMap:


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
+ std::vector< std::vector< bool > > adjacencyMatrix
+ static const std::string nodeNames
+ Graph()
+ int getNextNode(int currentId) const
+ std::shared_ptr< Tile > getTile(int id) const
+ bool hasEdge(int from, int to) const
+ std::string getNodeName(int id) const
GameMap
+ GameMap()
+ void displayMap() const
+ bool canUpgradeStreet(PropertyTile *targetStreet, int playerID) const
+ int movebahn() const
+ int moveHub() const

Collaboration diagram for GameMap:


string
vector< std::shared_ptr< Tile > >
vector< std::tuple < int, int, std::vector < int >, std::vector< int > > >
vector< int >
vector< std::vector < bool > >
+nodeNames
+tiles
+groups
+PropertyTiles
+adjacencyMatrix
Graph
+ Graph()
+ int getNextNode(int currentId) const
+ std::shared_ptr< Tile > getTile(int id) const
+ bool hasEdge(int from, int to) const
+ std::string getNodeName (int id) const
GameMap
+ GameMap()
+ void displayMap() const
+ bool canUpgradeStreet (PropertyTile *targetStreet, int playerID) const
+ int movebahn() const
+ int moveHub() const

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 Srassen-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.

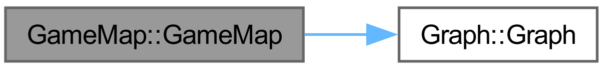
Contains functions for displaying map information, moving train stations and airports, and determining whether streets are upgradeable HKI-8 Map: Erstellung einer Spielfeld-Klasse

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
```

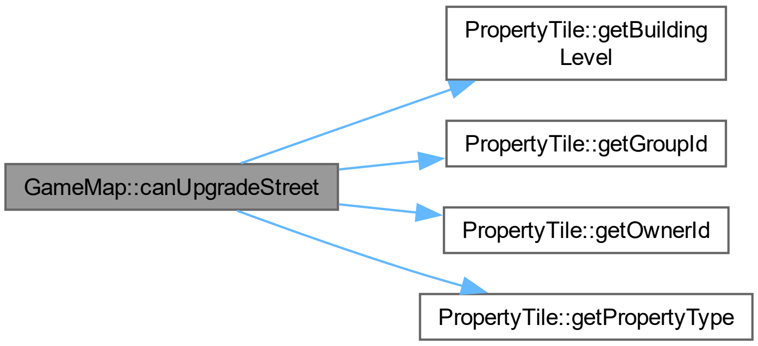
//Function: Verify if the street can be updated //HKI-9 Map: Implementierung der Strassenfelder HKI-10 Map↩
: Entwicklung der Srassen-Eigenschaften

Parameters

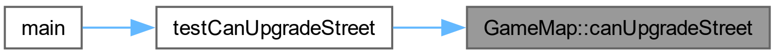
	targetStreet
	playerID

Returns

Here is the call graph for this function:


PropertyTile::getBuilding Level
PropertyTile::getGroupId
GameMap::canUpgradeStreet
PropertyTile::getOwnerId
PropertyTile::getPropertyType

Here is the caller graph for this function:



4.1.3.2 displayMap()

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

//Function: loop to display all node information

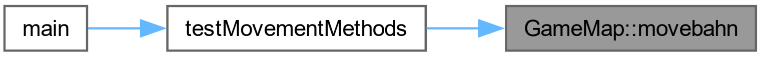
4.1.3.3 movebahn()

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

HKI-11 Map: Implementierung der Bahnhofsfelder.

Returns

Here is the caller graph for this function:



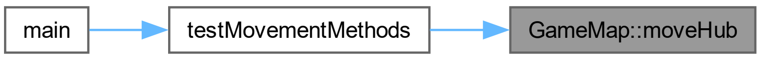
4.1.3.4 moveHub()

```
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:

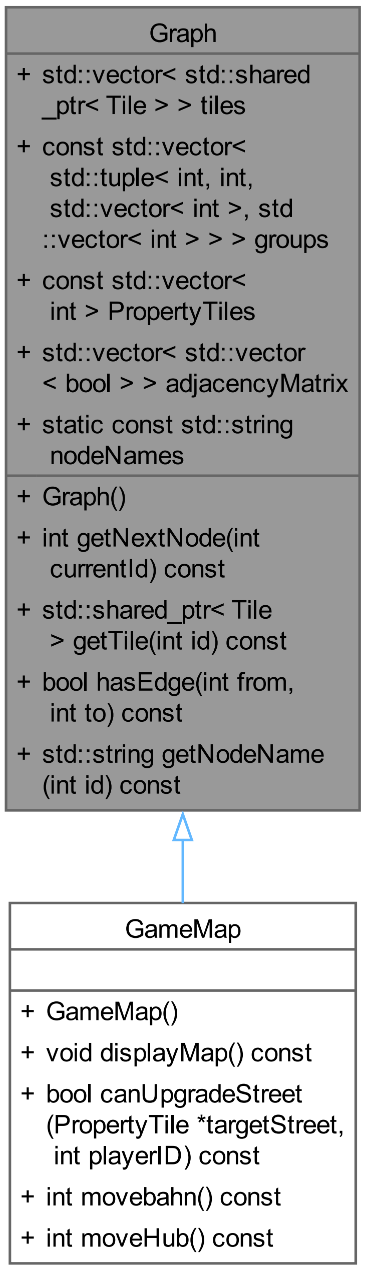
- Map.hpp
- Map.cpp

4.2 Graph Class Reference

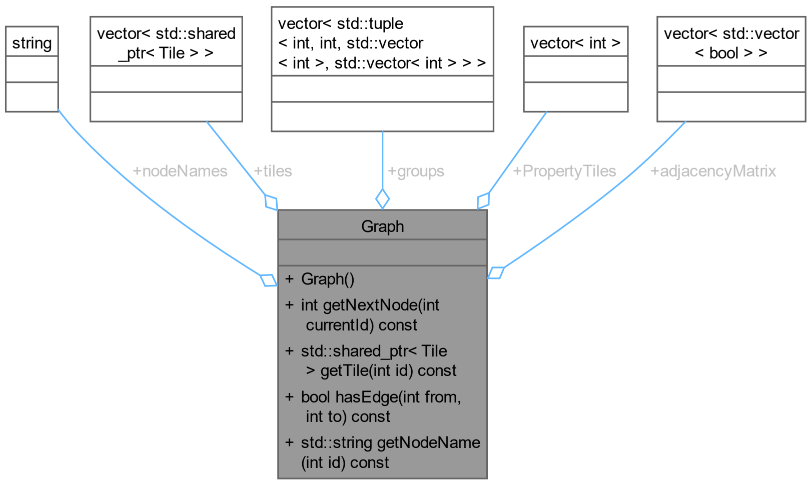
This is the Graph class.

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

Inheritance diagram for Graph:


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
+ std::vector< std::vector< bool > > adjacencyMatrix
+ static const std::string nodeNames
+ Graph()
+ int getNextNode(int currentId) const
+ std::shared_ptr< Tile > getTile(int id) const
+ bool hasEdge(int from, int to) const
+ std::string getNodeName(int id) const
GameMap
+ GameMap()
+ void displayMap() const
+ bool canUpgradeStreet(PropertyTile *targetStreet, int playerID) const
+ int movebahn() const
+ int moveHub() const

Collaboration diagram for Graph:


string
vector< std::shared_ptr< Tile > >
vector< std::tuple < int, int, std::vector < int >, std::vector< int > > >
vector< int >
vector< std::vector < bool > >
+nodeNames
+tiles
+groups
+PropertyTiles
+adjacencyMatrix
Graph
+ Graph()
+ int getNextNode(int currentId) const
+ std::shared_ptr< Tile > getTile(int id) const
+ bool hasEdge(int from, int to) const
+ std::string getNodeName (int id) const

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

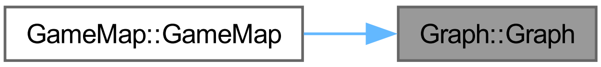
4.2.2 Constructor & Destructor Documentation

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

	currentId

Returns

4.2.3.2 getNodeName()

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

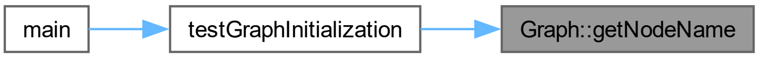
Function: Determine whether two Tiles are connected.

Parameters

	id

Returns

Here is the caller graph for this function:



4.2.3.3 getTile()

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

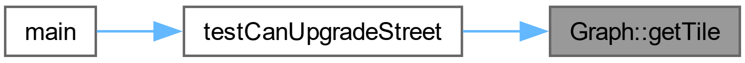
Function: Given a Tile ID, return a Tile object.

Parameters

	id

Returns

Here is the caller graph for this function:



4.2.3.4 hasEdge()

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

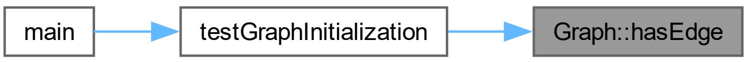
Function:Given a Tile ID, return a Tile name.

Parameters

	from
	to

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
```

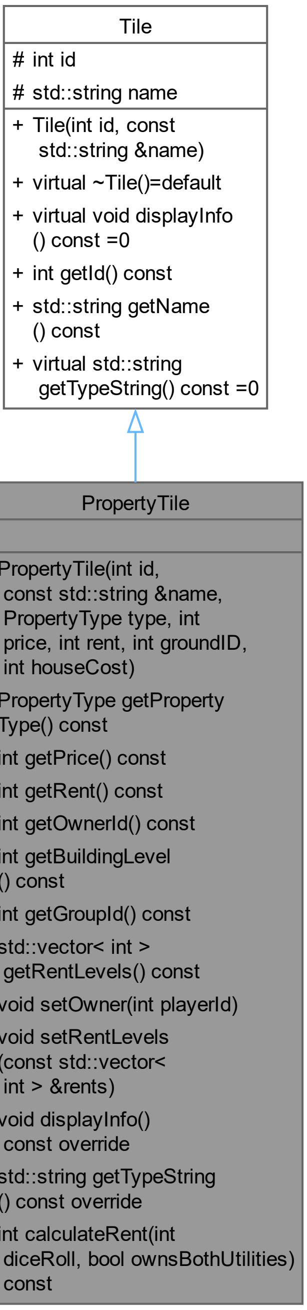
The documentation for this class was generated from the following files:

- Graph.hpp
- Graph.cpp

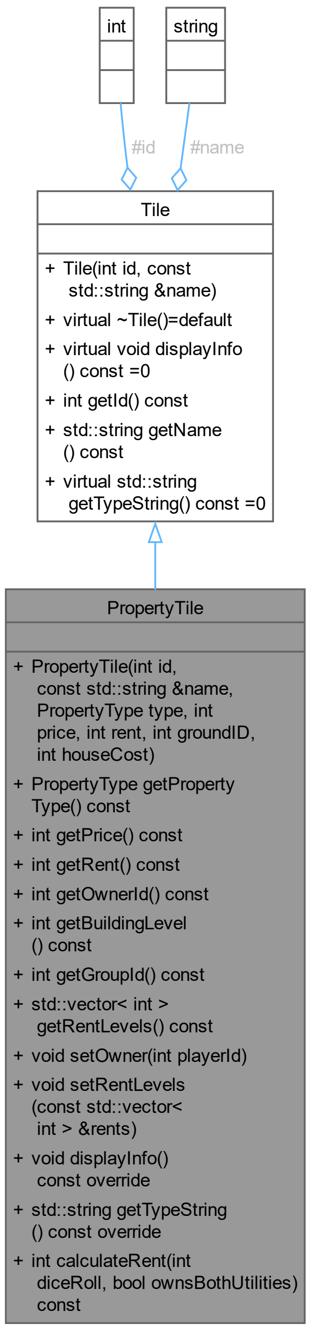
4.3 PropertyTile Class Reference

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

Inheritance diagram for PropertyTile:


Tile
# int id
# std::string name
+ 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
PropertyTile
+ PropertyTile(int id, const std::string &name, PropertyType type, int price, int rent, int groundID, int houseCost)
+ PropertyType getProperty Type() 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)
+ void displayInfo() const override
+ std::string getTypeString () const override
+ int calculateRent(int diceRoll, bool ownsBothUtilities) const

Collaboration diagram for PropertyTile:


int
string
#id
#name
Tile
+ 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
PropertyTile
+ 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)
+ void displayInfo() const override
+ std::string getTypeString() const override
+ int calculateRent(int diceRoll, bool ownsBothUtilities) const

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

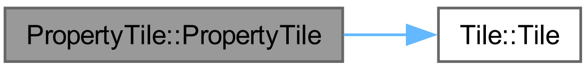
- int id
- std::string name

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

	diceRoll
	ownsBothUtilities

Returns

4.3.2.2 displayInfo()

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

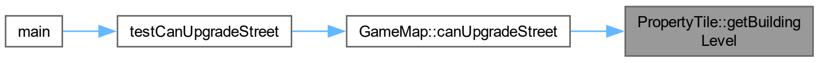
HKI-9 Map: Implementierung der Straßenfelder.

Implements Tile.

4.3.2.3 getBuildingLevel()

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

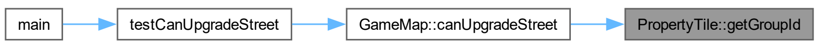
Here is the caller graph for this function:



4.3.2.4 getGroupId()

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

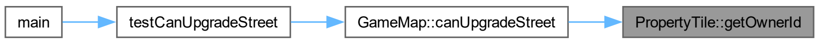
Here is the caller graph for this function:



4.3.2.5 getOwnerId()

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

Here is the caller graph for this function:



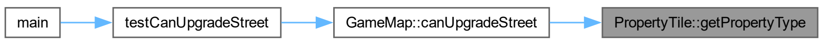
4.3.2.6 getPrice()

```
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.

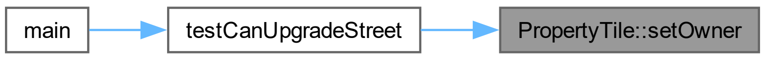
Returns

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

	rents

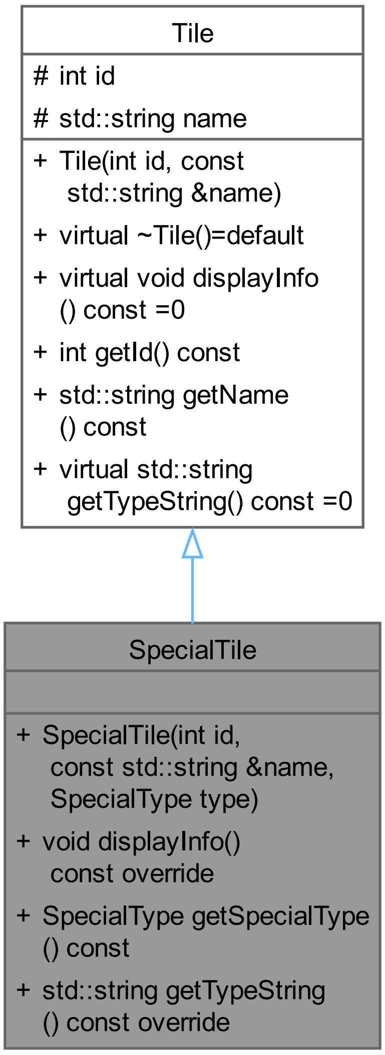
The documentation for this class was generated from the following files:

- PropertyTile.hpp
- PropertyTile.cpp

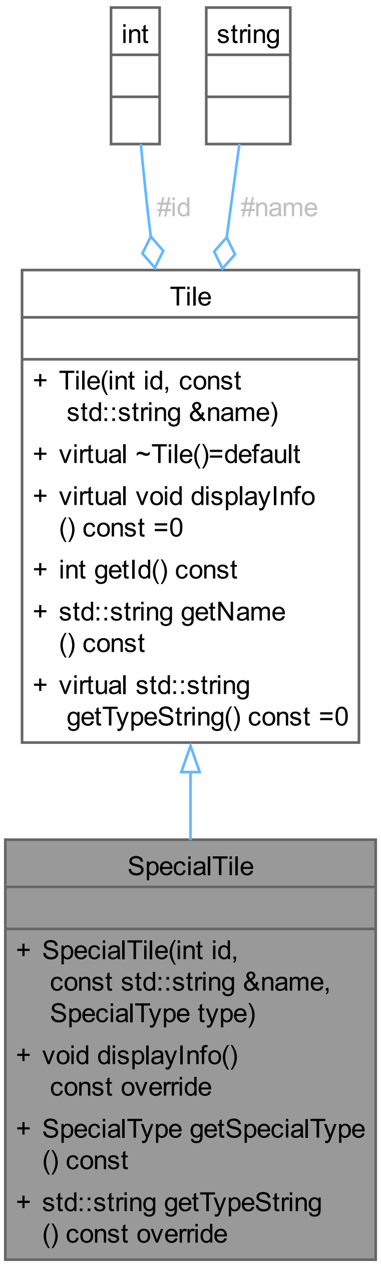
4.4 SpecialTile Class Reference

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

Inheritance diagram for SpecialTile:


Tile
# int id
# std::string name
+ 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
SpecialTile
+ SpecialTile(int id, const std::string &name, SpecialType type)
+ void displayInfo() const override
+ SpecialType getSpecialType() const
+ std::string getTypeString() const override

Collaboration diagram for SpecialTile:


int
string
#id
#name
Tile
+ 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
SpecialTile
+ SpecialTile(int id, const std::string &name, SpecialType type)
+ void displayInfo() const override
+ SpecialType getSpecialType () const
+ std::string getTypeString () const override

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

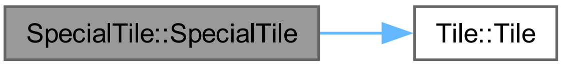
- int id
- std::string name

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:



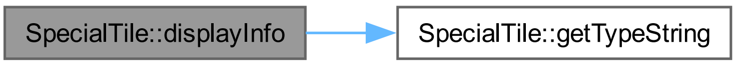
4.4.2 Member Function Documentation

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

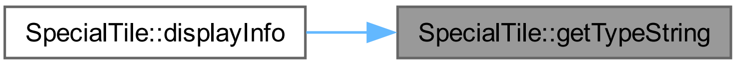
Returns

4.4.2.3 getTypeString()

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

Implements Tile.

Here is the caller graph for this function:

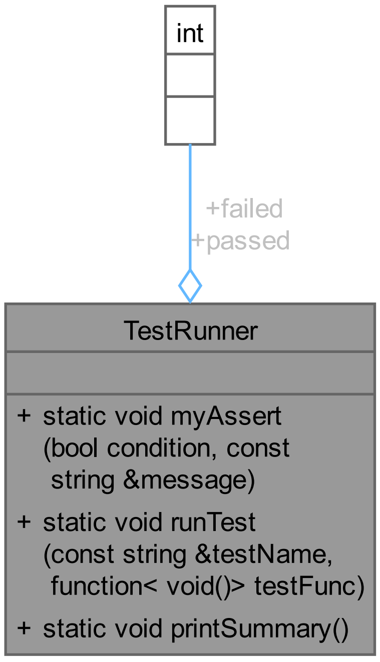


The documentation for this class was generated from the following files:

- SpecialTile.hpp
- SpecialTile.cpp

4.5 TestRunner Class Reference

Collaboration diagram for TestRunner:


int
+failed
+passed
TestRunner
+ static void myAssert (bool condition, const string &message)
+ static void runTest (const string &testName, function< void()> testFunc)
+ static void printSummary()

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

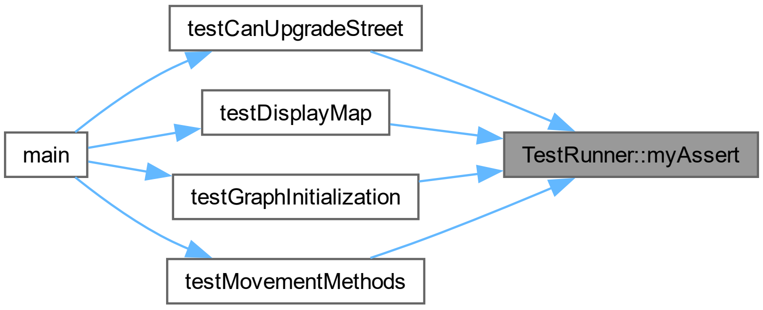
- static int passed = 0
- static int failed = 0

4.5.1 Member Function Documentation

4.5.1.1 myAssert()

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

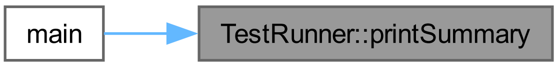
Here is the caller graph for this function:



4.5.1.2 printSummary()

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

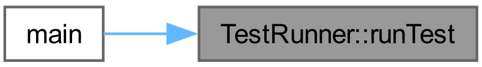
Here is the caller graph for this function:



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]
```

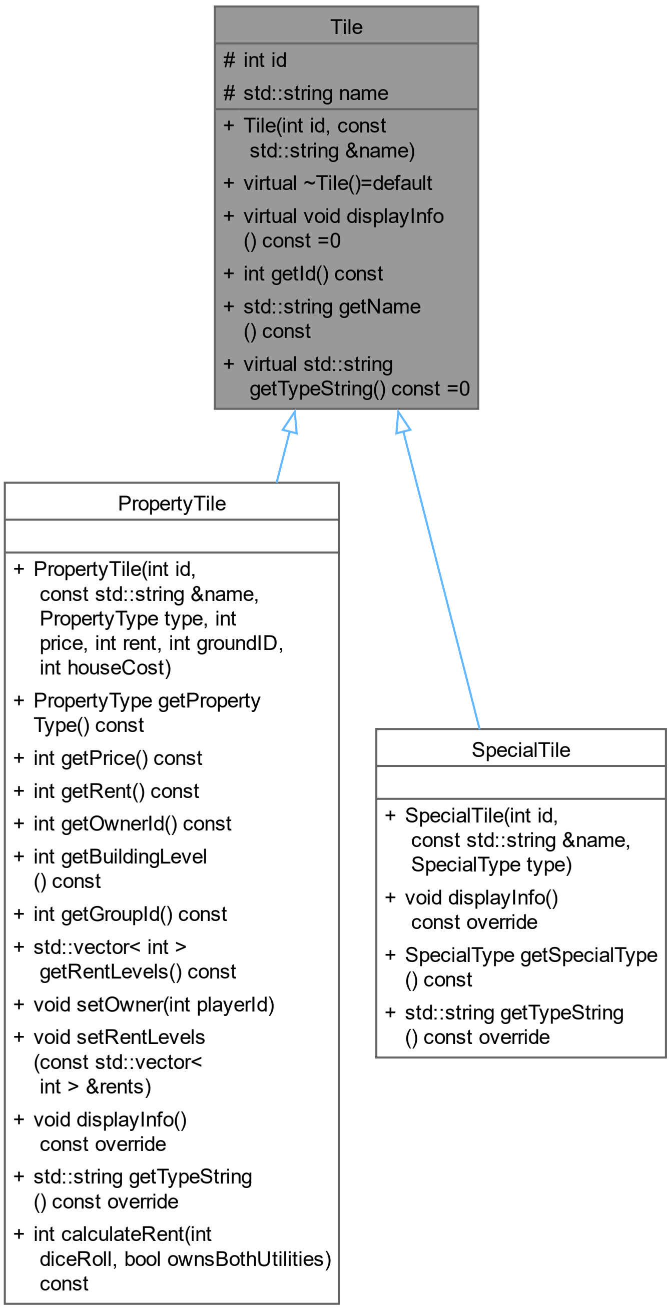
The documentation for this class was generated from the following file:

- test.cpp

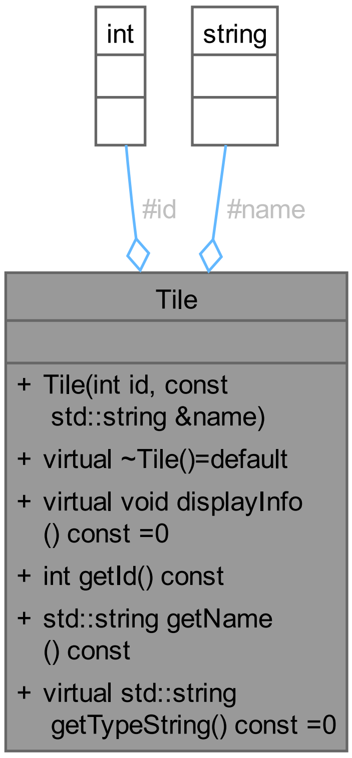
4.6 Tile Class Reference

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

Inheritance diagram for Tile:


Tile
# int id
# std::string name
+ 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
PropertyTile
+ 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)
+ void displayInfo() const override
+ std::string getTypeString() const override
+ int calculateRent(int diceRoll, bool ownsBothUtilities) const
SpecialTile
+ SpecialTile(int id, const std::string &name, SpecialType type)
+ void displayInfo() const override
+ SpecialType getSpecialType() const
+ std::string getTypeString() const override

Collaboration diagram for Tile:


int
string
#id
#name
Tile
+ 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

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

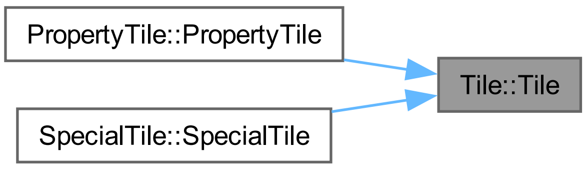
- int id
- std::string name

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

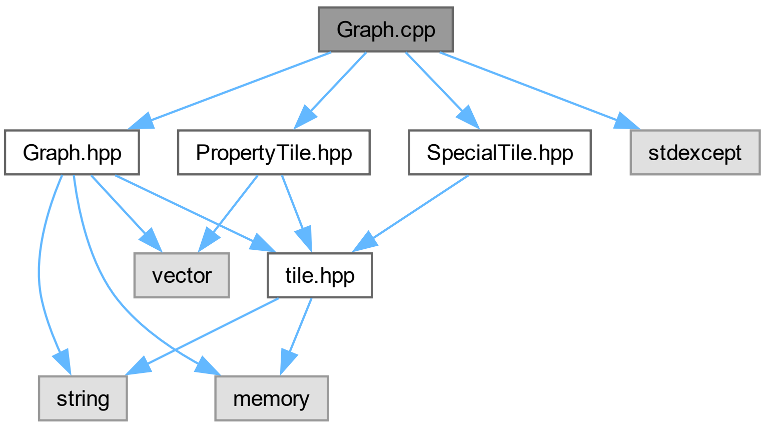
Chapter 5

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:

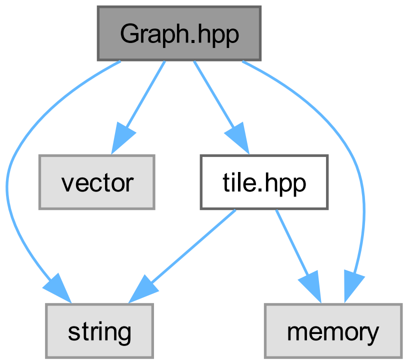


5.2 Graph.hpp File Reference

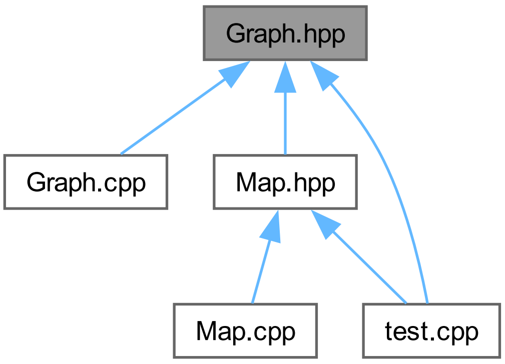
```
#include <string>
#include <vector>
#include <memory>
```

```
#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 Elektrizitätwerk 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 Hubschrauberlandeplatze

```
#define Hubschrauberlandeplatze 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

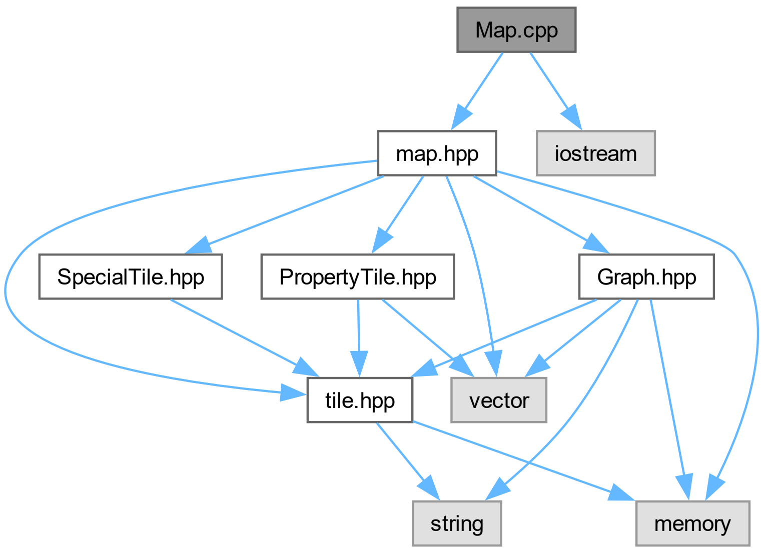
Go to the documentation of this file.

```

#ifndef GRAPH_HPP
#define GRAPH_HPP
#include <string>
#include <vector>
#include <memory>
#include "tile.hpp"
class Graph {
public:
    #define TOTAL_NODES 40
    #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 Elektrizitätwerk 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


    static const std::string nodeNames[TOTAL_NODES];
    std::vector<std::shared_ptr<Tile» tiles;
    const std::vector<std::tuple<int, int, std::vector<int>, std::vector<int»> groups= {
    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})
    };
    const std::vector<int> PropertyTiles = {
    Kronenstraße,
    Adlerstraße,
    Ebertstraße,
    Rüppurrerstraße,
    Ettlingerstraße,
    Amalienstraße,
    Hirschstraße,
    Kriegsstraße,
    Fastplatz,
    KaiserAllee,
    DurlacherAllee,
    Zirkel,
    Karlstraße,
    Brauerstraße,
    Hildapromenade,
    Moltkestraße,
    Karlfriedrichstraße,
    Herrenstraße,
    Waldstraße,
    Erbprinzenstraße,
    Kaiserstraße,
    Schlossplatz
    };


    std::vector<std::vector<bool» adjacencyMatrix;

    Graph();

    int getNextNode(int currentId) const;
    std::shared_ptr<Tile> getTile(int id) const;

    bool hasEdge(int from, int to) const;

    std::string getNodeName(int id) const;
};
#endif // GRAPH_HPP
```

5.4 Map.cpp File Reference

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

Include dependency graph for Map.cpp:

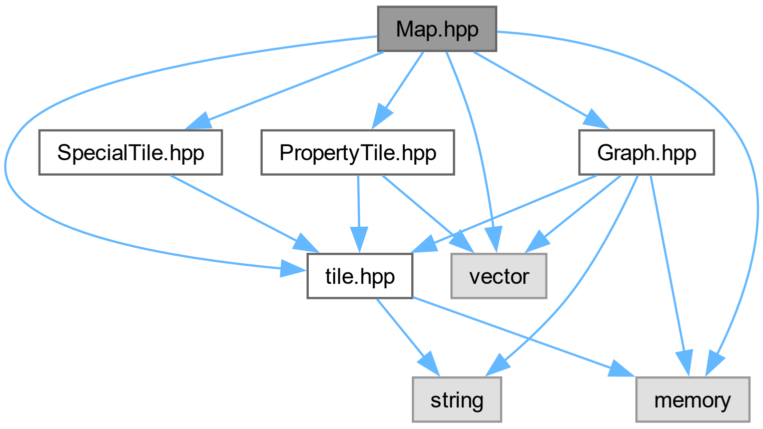


5.5 Map.hpp File Reference

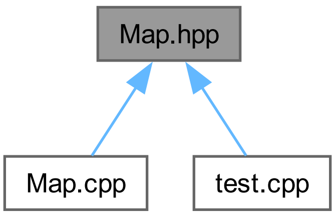
```
#include "tile.hpp"
#include "PropertyTile.hpp"
```

```
#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.

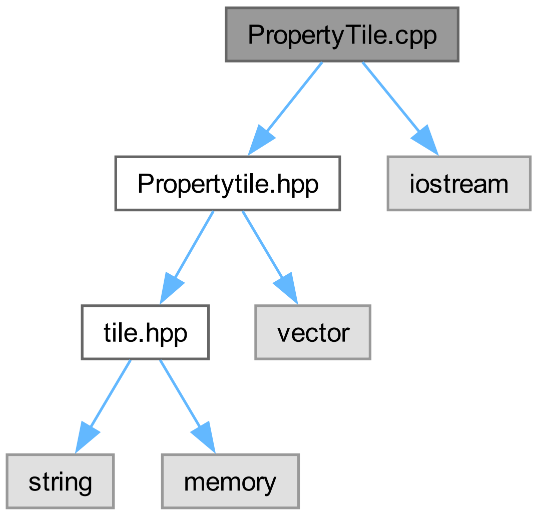
```
00001
00005 #ifndef MAP_HPP
```

```
#define MAP_HPP

#include "tile.hpp"
#include "PropertyTile.hpp"
#include "SpecialTile.hpp"
#include <vector>
#include <memory>
#include "Graph.hpp"

class GameMap:public Graph {
public:

    GameMap();
    void displayMap() const;
    bool canUpgradeStreet(PropertyTile* targetStreet, int playerID) const;
    int movebahn() const;
    int moveHub() const;
private:
    void printTileInfo(int nodeId) const;
};

#endif
```

5.7 PropertyTile.cpp File Reference

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

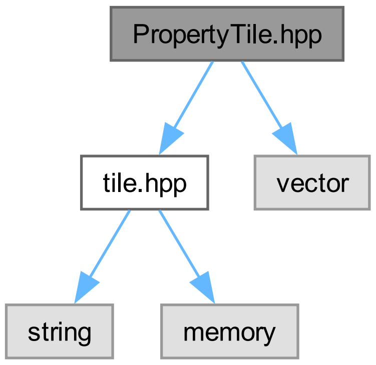
Include dependency graph for PropertyTile.cpp:



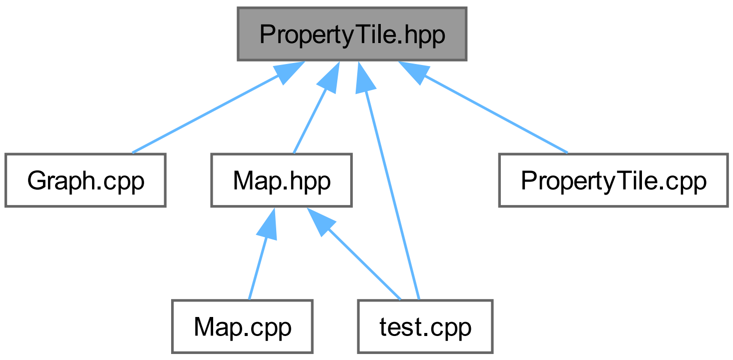
5.8 PropertyTile.hpp File Reference

```
#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

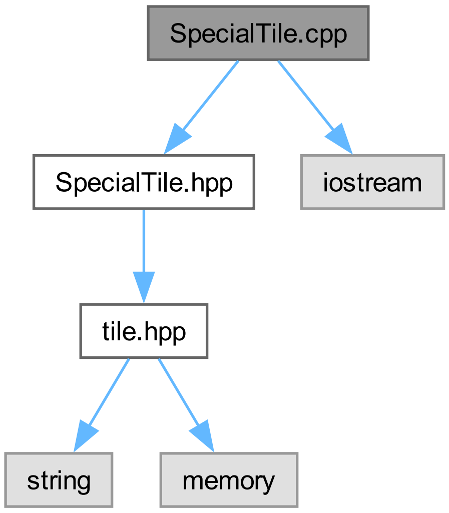
Go to the documentation of this file.

```
#ifndef PROPERTY_TILE_HPP
#define PROPERTY_TILE_HPP

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

enum class PropertyType { Street, Utility };

class PropertyTile : public Tile {
private:
    PropertyType propertyType;
    int price;
    int rent;
    int ownerId;

    int groudID;
    int buildingLevel;
    int houseCost;
    std::vector<int> rentLevels;
public:
    //PropertyTile(int id, const std::string& name, PropertyType type, int price): Tile(id, name),
propertyType(type), price(price){}
    PropertyTile(int id, const std::string& name, PropertyType type, int price, int rent,int groundID,
int houseCost)
        : Tile(id, name), propertyType(type), price(price), rent(rent), ownerId(-1),rentLevels({10,
50, 150, 450, 625, 750}),houseCost(100),
buildingLevel(0),groudID(-1) {}
// Getter
    PropertyType getPropertyType() const { return propertyType; }
    int getPrice() const { return price; }
    int getRent() const { return rent; }
    int getOwnerId() const { return ownerId; }
    int getBuildingLevel() const { return buildingLevel; }
    int getGroupId() const { return groudID; }
    std::vector<int> getRentLevels() const {return rentLevels;}
    // Setter
    void setOwner(int playerId) { ownerId = playerId; }
    void setRentLevels(const std::vector<int>& rents);
    void displayInfo() const override;
    std::string getTypeString() const override;
     int calculateRent(int diceRoll, bool ownsBothUtilities) const;
};

#endif
```

5.10 SpecialTile.cpp File Reference

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

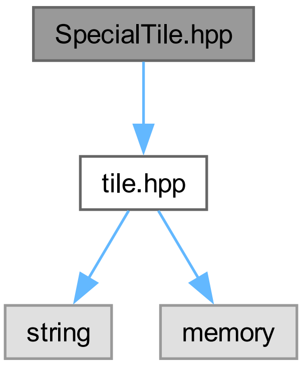
Include dependency graph for SpecialTile.cpp:



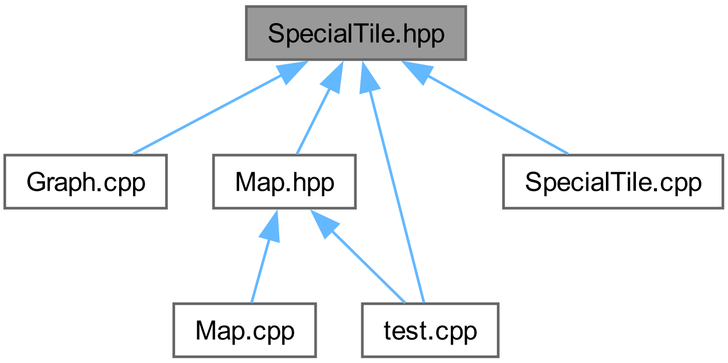
5.11 SpecialTile.hpp File Reference

```
#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

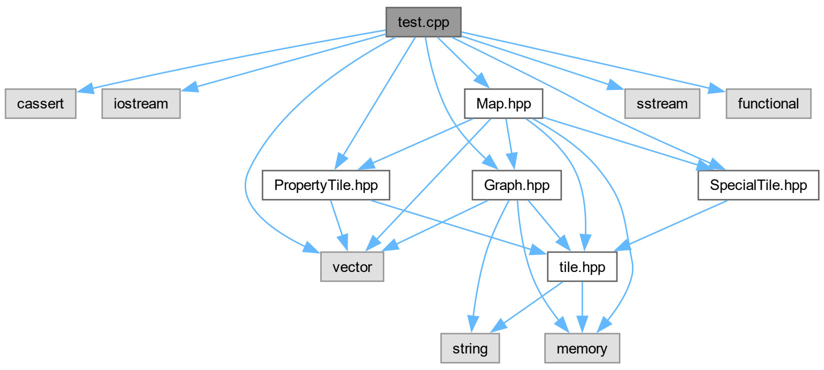
Go to the documentation of this file.

```
#ifndef SPECIAL_TILE_HPP
#define SPECIAL_TILE_HPP

#include "tile.hpp"

enum class SpecialType { Start, Event,Community, Bahnhof, Tax,LuxuryTax, Jail, GoToJail, FreeParking,
Hubschrauberlandeplatz};

class SpecialTile : public Tile {
private:
    SpecialType specialType;

public:
    SpecialTile(int id, const std::string& name, SpecialType type)
        : Tile(id, name), specialType(type) {}

    void displayInfo() const override;
    SpecialType getSpecialType() const { return specialType; }
    std::string getTypeString() const override;
};

#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

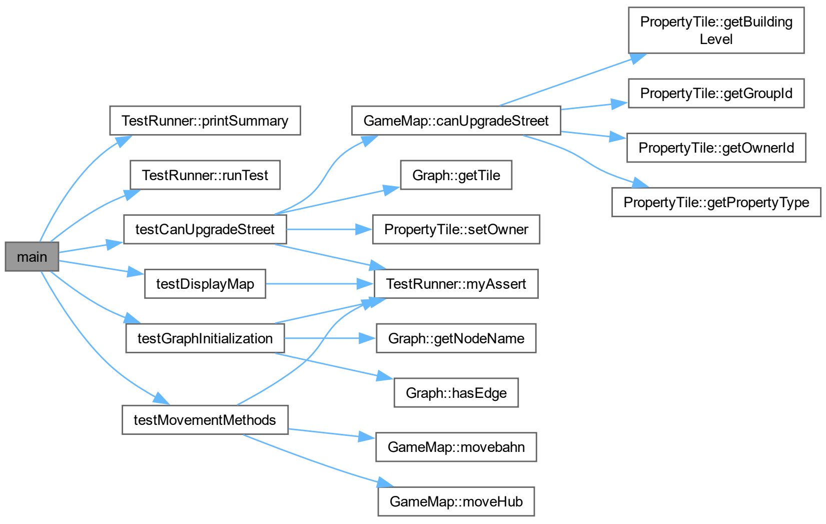
- void testGraphInitialization ()
- void testDisplayMap ()
- void testCanUpgradeStreet ()
- void testMovementMethods ()
- int main ()

5.13.1 Function Documentation

5.13.1.1 main()

```
int main ()
```

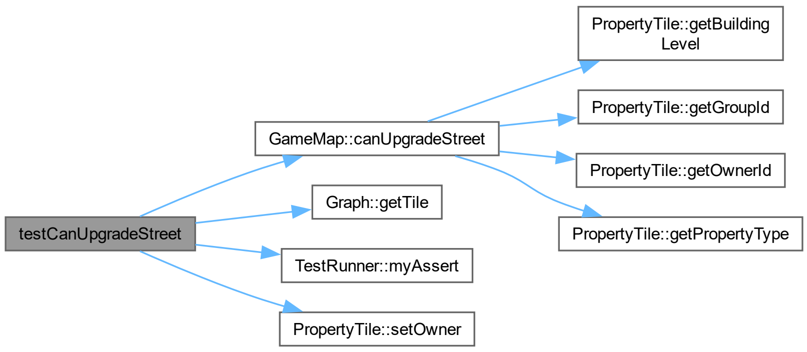
Here is the call graph for this function:



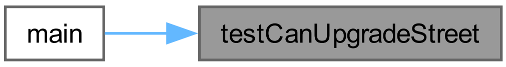
5.13.1.2 testCanUpgradeStreet()

```
void testCanUpgradeStreet ()
```

Here is the call graph for this function:



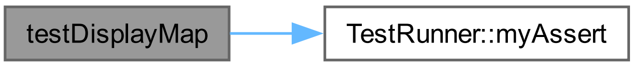
Here is the caller graph for this function:



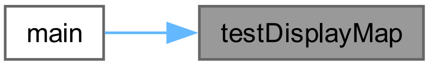
5.13.1.3 testDisplayMap()

```
void testDisplayMap ()
```

Here is the call graph for this function:



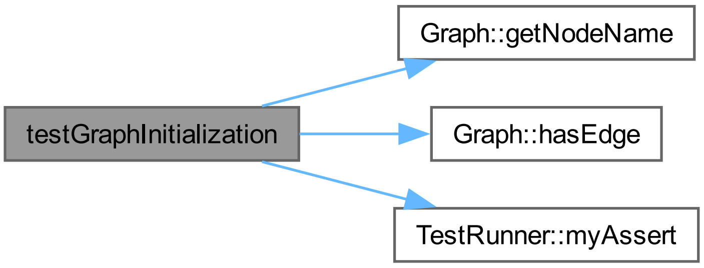
Here is the caller graph for this function:



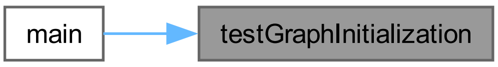
5.13.1.4 testGraphInitialization()

```
void testGraphInitialization ()
```

Here is the call graph for this function:



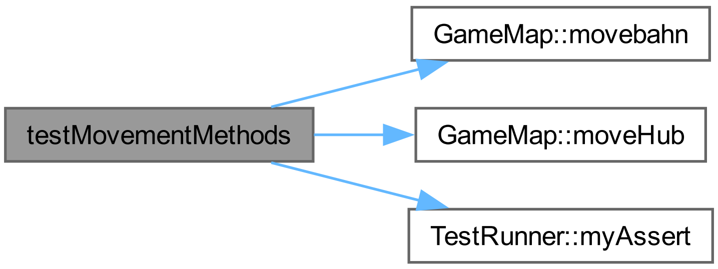
Here is the caller graph for this function:



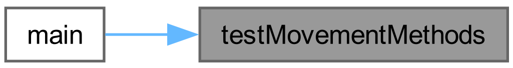
5.13.1.5 testMovementMethods()

```
void testMovementMethods ()
```

Here is the call graph for this function:



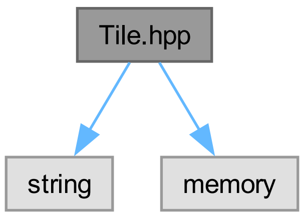
Here is the caller graph for this function:



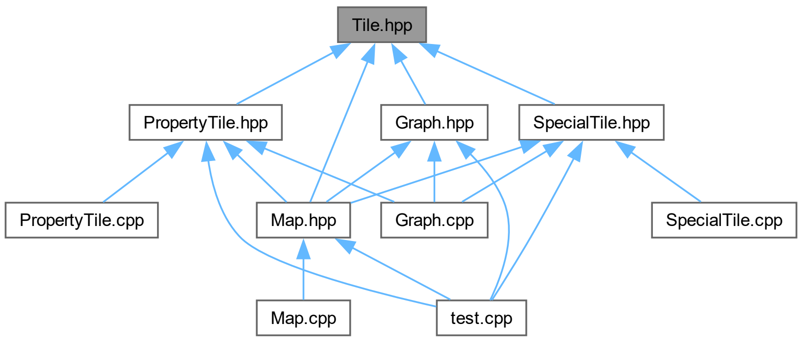
5.14 Tile.hpp File Reference

```
#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.

```
//Abstract class, pure virtual function definition Tile
#ifndef TILE_HPP
#define TILE_HPP

#include <string>
#include <memory>

class Tile {
protected:
    int id;
    std::string name;

public:
    Tile(int id, const std::string& name) : id(id), name(name) {}
    virtual ~Tile() = default;

    virtual void displayInfo() const = 0;

    int getId() const { return id; }
    std::string getName() const { return name; }
    virtual std::string getTypeString() const=0;
};

#endif
```

Index