Metro Simulation

Generated by Doxygen 1.9.6

Chapter 1

Class Index

1.1 Class List

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

Line . . . . . ??
Logger . . . . . ??
MetroSimulation . . . . . ??
MetroSystem . . . . . ??
MetroXMLParser . . . . . ??
Station . . . . . ??
Track . . . . . ??
Tram . . . . . ??

Chapter 2

File Index

2.1 File List

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

DesignByContract.h . . . . . ??
Line.h . . . . . ??
Logger.h . . . . . ??
MetroSimulation.h . . . . . ??
MetroSystem.h . . . . . ??
MetroXMLParser.h . . . . . ??
Station.h . . . . . ??
Track.h . . . . . ??
Tram.h . . . . . ??

Chapter 3

Class Documentation

3.1 Line Class Reference

`#include <Line.h>`

Public Member Functions

- Line (int lineNumber)
- virtual ~Line ()
- bool **properlyInitialised** () const
- void update (std::ostream &os)
- const std::vector< Track * > & getTracks () const
- const std::vector< Tram * > & getTrams () const
- int getLineNumber () const
- void addTrack (Track *newTrack)
- void addTram (Tram *newTram)

3.1.1 Detailed Description

Line object that contains Tracks and Trams

3.1.2 Constructor & Destructor Documentation

3.1.2.1 Line()

```
Line::Line (
            int lineNumber )  [explicit]
```

Creates a Line Object

@ENSURE properlyInitialised(), "constructor must end in properlyInitialized state"

Parameters

lineNumber	is the lineNumber of the Line

3.1.2.2 ~Line()

```
Line::~Line ( )  [virtual]
```

Destructs a Line Object

Destructs all the Trams and Tracks @REQUIRE properlyInitialised(), "The line was not properly initialised."

3.1.3 Member Function Documentation

3.1.3.1 addTrack()

```
void Line::addTrack (
            Track * newTrack )
```

Adds a track tot the line object.

@REQUIRE properlyInitialised(), "The line was not properly initialised."

Parameters

newTrack	is the new Track that will be added to the Line

3.1.3.2 addTram()

```
void Line::addTram (
            Tram * newTram )
```

Adds a tram tot the line object.

@REQUIRE properlyInitialised(), "The line was not properly initialised."

Parameters

newTram	is the new Tram that will be added to the Line

3.1.3.3 getLineNumber()

```
int Line::getLineNumber ( ) const
```

Returns the LineNumber.

@REQUIRE properlyInitialised(), "The line was not properly initialised."

3.1.3.4 getTracks()

```
const std::vector< Track * > & Line::getTracks ( ) const
```

Returns the vector of tracks.

@REQUIRE properlyInitialised(), "The line was not properly initialised."

3.1.3.5 getTrams()

```
const std::vector< Tram * > & Line::getTrams ( ) const
```

Returns the vector of trams.

@REQUIRE properlyInitialised(), "The line was not properly initialised."

3.1.3.6 update()

```
void Line::update (
            std::ostream & os )
```

Updates/Moves the trams of the line object.

@REQUIRE properlyInitialised(), "The line was not properly initialised."

Parameters

os	is the stream the print statements get send into

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

- Line.h
- Line.cpp

3.2 Logger Class Reference

```
#include <Logger.h>
```

Static Public Member Functions

- static void writeError (std::ostream &stream, const std::string &msg)

3.2.1 Detailed Description

Logger object for simple output formatting

3.2.2 Member Function Documentation

3.2.2.1 writeError()

```
static void Logger::writeError (
            std::ostream & stream,
            const std::string & msg )  [inline], [static]
```

Writes an error message to the given stream, with right formatting

Parameters

stream	is the stream the error message needs to be written to
msg	is the message that needs to be formatted

Attention

newline gets added by default

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

- Logger.h

3.3 MetroSimulation Class Reference

Public Member Functions

- **MetroSimulation** (const std::string &inputFile, std::ostream &errorstream, unsigned int runtime)
- MetroSystem * getSystem () const
- void run (std::ostream &os)

3.3.1 Member Function Documentation

3.3.1.1 getSystem()

```
MetroSystem * MetroSimulation::getSystem ( ) const
```

Gives the metro-system that is used for the simulation

Returns

3.3.1.2 run()

```
void MetroSimulation::run (
            std::ostream & os )
```

Updates the metro-system so all the trams will move to their next location

Parameters

os	std::ostream: Output stream where the movement of the the trams gets written to

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

- MetroSimulation.h
- MetroSimulation.cpp

3.4 MetroSystem Class Reference

Public Member Functions

- **MetroSystem** (const std::string &filename, std::ostream &errorstream)
- void updateSystem (std::ostream &os)
- void outputSystem (std::ostream &os)
- void createDotFile (std::ostream &os)
- bool **properlyInitialized** () const

3.4.1 Member Function Documentation

3.4.1.1 createDotFile()

```
void MetroSystem::createDotFile (
            std::ostream & os )
```

Creates dot file of the current metro-system @REQUIRE(properlyInitialized(), "Metrosimulation is not properly initialised.");

Parameters

os	

3.4.1.2 outputSystem()

```
void MetroSystem::outputSystem (
            std::ostream & os )
```

Outputs the system to the given output stream @REQUIRE(properlyInitialized(), "Metrosimulation is not properly initialised.");

Parameters

os	The outputstream where the current metro-system gets written to

3.4.1.3 updateSystem()

```
void MetroSystem::updateSystem (
            std::ostream & os )
```

Updates the lines of the metro-system @REQUIRE(properlyInitialized(), "Metrosimulation is not properly initialised.");

Parameters

os	The output stream where the updates of the system get written to

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

- MetroSystem.h
- MetroSystem.cpp

3.5 MetroXMLParser Class Reference

```
#include <MetroXMLParser.h>
```

Public Member Functions

- **MetroXMLParser** (const std::string &filename, std::ostream &errorStream)
- bool parse (const std::string &filename)

- void parseStation (TiXmlElement ∗stationElem)
- void parseTram (TiXmlElement ∗tramElem)
- void handleStations ()
- void handleTrams ()
- bool verify ()
- bool **isProperlyParsed** () const
- bool **properlyInitialized** () const
- const std::vector< Tram ∗ > & **getTrams** () const
- const std::vector< Station ∗ > & **getStations** () const
- const std::vector< Line ∗ > & **getLines** () const

3.5.1 Detailed Description

Parser for MetroSimulation

3.5.2 Member Function Documentation

3.5.2.1 handleStations()

```
void MetroXMLParser::handleStations ( )
```

Initialises the stations completely

@REQUIRE THERE ARE STATIONS @REQUIRE STATION MAP NOT EMPTY

Attention

parse() needs to be called before this should be called

3.5.2.2 handleTrams()

```
void MetroXMLParser::handleTrams ( )
```

Initialises the trams completely

@REQUIRE THERE ARE TRAMS @REQUIRE TRAM MAP NOT EMPTY

Attention

parse() needs to be called before this should be called

3.5.2.3 parse()

```
bool MetroXMLParser::parse (
            const std::string & filename )
```

Parses the filename
Creates Objects who are not completely initialised
Creates 2 maps with content to init these objects completely

@REQUIRE properlyInitialized(), "MetroXMLParser was not initialized when calling parse"

Parameters

filename	is Path to the file to be parsed, given as a string

Returns

bool: True if parsed properly, False if failed to parse

3.5.2.4 parseStation()

```
void MetroXMLParser::parseStation (
            TiXmlElement * stationElem )
```

Parses a single station form TiXmlElement @REQUIRE properlyInitialized(), "MetroXMLParser was not initialized when calling parse"

Parameters

stationElem	tinyXML element that contains information about a station

3.5.2.5 parseTram()

```
void MetroXMLParser::parseTram (
            TiXmlElement * tramElem )
```

Parses a single tram form TiXmlElement @REQUIRE properlyInitialized(), "MetroXMLParser was not initialized when calling parse"

Parameters

tramElem	tinyXML element that contains information about a tram

3.5.2.6 verify()

```
bool MetroXMLParser::verify ( )
```

Verifies the content after completely parsing.

@ENSURE STATIONS CONNECTED PROPERLY @ENSURE VALID STARTSTATION OF TRAM @ENSURE CORRESPONDING LINENUMBER BETWEEN TRAM AND STARTSTATION @ENSURE EVERY LINE HAS A TRAM

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

- MetroXMLParser.h
- MetroXMLParser.cpp

3.6 Station Class Reference

Public Member Functions

- **Station** (const std::string &name, Track *nextTrack, Track *prevTrack, int lineNumber)
- operator std::string ()
- const std::string & getName () const
- Track * getNextTrack () const
- Track * getPrevTrack () const
- int getLineNumber () const
- void setName (const std::string &name)
- void setNextTrack (Station *nextStation)
- void setPrevTrack (Station *prevStation)
- void setLineNumber (int lineNumber)

Friends

- std::ostream & **operator**<< (std::ostream &os, const Station &station)

3.6.1 Member Function Documentation

3.6.1.1 getLineNumber()

```
int Station::getLineNumber ( ) const
```

Gives the line number of the station

Returns

3.6.1.2 getName()

```
const std::string & Station::getName ( ) const
```

Gives the name of the station

Returns

string: name of the station

3.6.1.3 getNextTrack()

```
Track * Station::getNextTrack ( ) const
```

Gives a pointer to the next track of the station

Returns

Track* to the next track

3.6.1.4 getPrevTrack()

```
Track * Station::getPrevTrack ( ) const
```

Gives a pointer to the previous track of the station

Returns

Track* to the previous track

3.6.1.5 operator std::string()

```
Station::operator std::string ( )
```

Converts Track into a string @REQUIRE(properlyInitialized(), "Station was not properly initialised.");

3.6.1.6 setLineNumber()

```
void Station::setLineNumber (
            int lineNumber )
```

Sets the line number of the station

Parameters

lineNumber	the line number given to the station

3.6.1.7 setName()

```
void Station::setName (
            const std::string & name )
```

Sets the name of the station

Parameters

name	name to be given to the station

3.6.1.8 setNextTrack()

```
void Station::setNextTrack (
            Station * nextStation )
```

Sets next station of the station

Parameters

nextStation	the next station given to the station

3.6.1.9 setPrevTrack()

```
void Station::setPrevTrack (
            Station * prevStation )
```

Sets the previous station

Parameters

prevStation	the previous station given to the station

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

- Station.h
- Station.cpp

3.7 Track Class Reference

Public Member Functions

- **Track** (Station *begin, Station *anEnd)
- Station * getBegin () const
- Station * getAnEnd () const

3.7.1 Member Function Documentation

3.7.1.1 getAnEnd()

```
Station * Track::getAnEnd ( ) const
```

Gets the station at the end of the track

Returns

Station* to the station at the end of the track

3.7.1.2 getBegin()

```
Station * Track::getBegin ( ) const
```

Gets the station at the beginning of the track

Returns

Station* to the station at the beginning of the track

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

- Track.h
- Track.cpp

3.8 Tram Class Reference

Public Member Functions

- **Tram** (int lineNumber, int tramNumber, int speed, Station *startStation)
- operator std::string ()
- int getLineNumber () const
- int getSpeed () const
- Station * getStartStation () const
- int getTramNumber () const
- Station * getCurrentStation () const
- void setSpeed (int speed)
- void setStartStation (Station *startStation)
- void setTramNumber (int tramNumber)
- void drive (std::ostream &os)

Friends

- std::ostream & **operator**<< (std::ostream &os, const Tram &tram)

3.8.1 Member Function Documentation

3.8.1.1 drive()

```
void Tram::drive (
            std::ostream & os )
```

Moves the tram to the next station

Parameters

os	std::ostream: where the output of moving the train is written to

3.8.1.2 getCurrentStation()

```
Station * Tram::getCurrentStation ( ) const
```

Gives the station where the tram currently is

Returns

Station∗ to the station where the tram currently is

3.8.1.3 getLineNumber()

```
int Tram::getLineNumber ( ) const
```

Gives the line number of the track on which the tram is driving

Returns

int: line number of the track on which the tram is driving

3.8.1.4 getSpeed()

```
int Tram::getSpeed ( ) const
```

Gives the speed of which the tram is capable

Returns

int: the speed of which the tram is capable

3.8.1.5 getStartStation()

```
Station * Tram::getStartStation ( ) const
```

Gives the station where the track starts

Returns

Station∗ to the station where the track starts

3.8.1.6 getTramNumber()

```
int Tram::getTramNumber ( ) const
```

Gives the tram number of the tram

Returns

int: the tram number of the tram

3.8.1.7 operator std::string()

```
Tram::operator std::string ( )
```

Converts Tram into a string @REQUIRE(properlyInitialized(), "Station was not properly initialised.");

3.8.1.8 setSpeed()

```
void Tram::setSpeed (
            int speed )
```

Sets the speed of the tram

Parameters

speed	int: speed of the tram

3.8.1.9 setStartStation()

```
void Tram::setStartStation (
            Station * startStation )
```

Sets the station where the train starts at the start of the system

Parameters

startStation	Station *: station where the tram starts form

3.8.1.10 setTramNumber()

```
void Tram::setTramNumber (
            int tramNumber )
```

Sets the tram number of the tram

Parameters

tramNumber	int: the tram number given to the tram

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

- Tram.h
- Tram.cpp

Chapter 4

File Documentation

4.1 DesignByContract.h

```
00001 //=============================================================================
00002 // Name        : DesignByContract.h
00003 // Author      : Serge Demeyer, modified by Kasper Engelen
00004 // Version     :
00005 // Copyright   : Project Software Engineering - BA1 Informatica - Serge Demeyer - University of
      Antwerp
00006 // Description : Declarations for design by contract in C++
00007 //=============================================================================
00008
00009 #include <assert.h>
00010
00011 #if defined(__assert)
00012 #define REQUIRE(assertion, what) \
00013     if (!(assertion)) __assert (what, __FILE__, __LINE__)
00014
00015 #define ENSURE(assertion, what) \
00016     if (!(assertion)) __assert (what, __FILE__, __LINE__)
00017 #else
00018 #define REQUIRE(assertion, what) \
00019     if (!(assertion)) _assert (what, __FILE__, __LINE__)
00020
00021 #define ENSURE(assertion, what) \
00022     if (!(assertion)) _assert (what, __FILE__, __LINE__)
00023 #endif
```

4.2 Line.h

```
00001 #ifndef PSE_METRO_SIMULATIE_LINE_H
00002 #define PSE_METRO_SIMULATIE_LINE_H
00003 #include "vector"
00004 #include "iostream"
00005
00006 class Track;
00007 class Tram;
00008
00012 class Line {
00013 public:
00020     explicit Line(int lineNumber);
00027     virtual ~Line();
00028
00029     bool properlyInitialised() const;
00030
00037     void update(std::ostream &os);
00038
00044     const std::vector<Track *> &getTracks() const;
00045
00051     const std::vector<Tram *> &getTrams() const;
00052
00058     int getLineNumber() const;
00059
00066     void addTrack(Track* newTrack);
00067
00074     void addTram(Tram* newTram);
00075 private:
```

```
00076     int lineNumber;
00077     std::vector<Track*> tracks;
00078     std::vector<Tram*> trams;
00079     Line* _initCheck;
00080 };
00081
00082
00083 #endif //PSE_METRO_SIMULATIE_LINE_H
```

4.3 Logger.h

```
00001 #ifndef PSE_METRO_SIMULATIE_LOGGER_H
00002 #define PSE_METRO_SIMULATIE_LOGGER_H
00003
00004 #include <iostream>
00005
00009 class Logger {
00010 public:
00017     static void writeError(std::ostream &stream, const std::string &msg) {
00018         stream « "ERROR: " « msg « std::endl;
00019     }
00020 };
00021
00022
00023 #endif //PSE_METRO_SIMULATIE_LOGGER_H
```

4.4 MetroSimulation.h

```
00001 #ifndef PSE_METRO_SIMULATIE_METROSIMULATION_H
00002 #define PSE_METRO_SIMULATIE_METROSIMULATION_H
00003 #include "MetroSystem.h"
00004
00005 class MetroSimulation  {
00006 public:
00007     // Constructor
00008     MetroSimulation(const std::string &inputFile, std::ostream &errorstream, unsigned int runtime);
00009     // Destructor
00010     virtual ~MetroSimulation();
00011
00016     MetroSystem *getSystem() const;
00017
00022     void run(std::ostream &os);
00023
00024 private:
00025     bool properlyInitialized() const;
00026
00027     MetroSystem *system;
00028     unsigned int runtime;
00029     unsigned int time;
00030
00031     MetroSimulation* _initCheck;
00032 };
00033
00034
00035 #endif //PSE_METRO_SIMULATIE_METROSIMULATION_H
```

4.5 MetroSystem.h

```
00001 #ifndef PSE_METRO_SIMULATIE_METROSYSTEM_H
00002 #define PSE_METRO_SIMULATIE_METROSYSTEM_H
00003
00004 #include "vector"
00005 #include "Line.h"
00006 #include "Station.h"
00007 #include "DesignByContract.h"
00008 #include "MetroXMLParser.h"
00009 #include "Logger.h"
00010
00011 class MetroSystem {
00012 public:
00013     // Constructor
00014     explicit MetroSystem(const std::string& filename, std::ostream &errorstream);
00015
00016     // Other
00022     void updateSystem(std::ostream &os);
```

```
00023
00029     void outputSystem(std::ostream &os);
00030
00036     void createDotFile(std::ostream &os);
00037
00038     bool properlyInitialized() const;
00039
00040 private:
00041     std::vector<Line*> lines;
00042     std::vector<Station*> stations;
00043     std::vector<Tram*> trams;
00044     MetroSystem* _initCheck;
00045 //    std::ostream &errorstream;
00046 };
00047
00048
00049 #endif //PSE_METRO_SIMULATIE_METROSYSTEM_H
```

4.6 MetroXMLParser.h

```
00001 #ifndef PSE_METRO_SIMULATIE_XMLPARSER_H
00002 #define PSE_METRO_SIMULATIE_XMLPARSER_H
00003 #include "string"
00004 #include "vector"
00005 #include "map"
00006 #include "ostream"
00007 #include "../tinyxml/tinyxml.h"
00008 #include "Tram.h"
00009 #include "Station.h"
00010 #include "Line.h"
00011 #include "fstream"
00012
00016 class MetroXMLParser {
00017 public:
00018     // Constructor
00019     explicit MetroXMLParser(const std::string &filename, std::ostream &errorStream);
00020     // Destructor
00021     virtual ~MetroXMLParser();
00022
00032     bool parse(const std::string& filename);
00033
00039     void parseStation(TiXmlElement* stationElem);
00040
00046     void parseTram(TiXmlElement* tramElem);
00047
00056     void handleStations();
00057
00066     void handleTrams();
00067
00076     bool verify();
00077
00078     bool isProperlyParsed() const;
00079     bool properlyInitialized() const;
00080
00081     const std::vector<Tram *> &getTrams() const;
00082     const std::vector<Station *> &getStations() const;
00083     const std::vector<Line *> &getLines() const;
00084 private:
00085     std::pair<std::string, bool> readKey(TiXmlElement* elem, const std::string &key);
00086
00087     bool properlyParsed;
00088     MetroXMLParser *_initCheck;
00089     std::ostream &errorstream;
00090     std::vector<Tram*> trams;
00091     std::vector<Station*> stations;
00092     std::vector<Line*> lines;
00093     std::map<Station*, std::pair<std::string, std::string> > stationMap;
00094     std::map<Tram*, std::string> tramMap;
00095 };
00096
00097 #endif //PSE_METRO_SIMULATIE_XMLPARSER_H
```

4.7 Station.h

```
00001 #ifndef PSE_METRO_SIMULATIE_STATION_H
00002 #define PSE_METRO_SIMULATIE_STATION_H
00003 #include <string>
00004 #include <ostream>
00005
```

```
00006 class Track;
00007
00008 class Station {
00009 public:
00010     // Constructor
00011     Station();
00012     Station(const std::string &name, Track *nextTrack, Track *prevTrack, int lineNumber);
00013     // Destructor
00014     ~Station();
00015
00016     // Operators
00021     operator std::string();
00022
00023     // Getters
00028     const std::string &getName() const;
00029
00034     Track *getNextTrack() const;
00035
00040     Track *getPrevTrack() const;
00041
00046     int getLineNumber() const;
00047
00048     // Setters
00053     void setName(const std::string &name);
00054
00059     void setNextTrack(Station *nextStation);
00060
00065     void setPrevTrack(Station *prevStation);
00066
00071     void setLineNumber(int lineNumber);
00072
00073 private:
00074     // other
00075     bool properlyInitialized() const;
00076
00077     // Data
00078     std::string name;
00079     Track *nextTrack;
00080     Track *prevTrack;
00081     int lineNumber;
00082
00083     Station* _initCheck;
00084
00085     // OS
00086     friend std::ostream &operator<<(std::ostream &os, const Station &station);
00087 };
00088
00089
00090 #endif //PSE_METRO_SIMULATIE_STATION_H
```

4.8 Track.h

```
00001 #ifndef PSE_METRO_SIMULATIE_TRACK_H
00002 #define PSE_METRO_SIMULATIE_TRACK_H
00003
00004 class Station;
00005
00006 class Track {
00007 public:
00008     // Constructor
00009     Track(Station *begin, Station *anEnd);
00010     // Destructor
00011     virtual ~Track();
00012
00013     //Getters
00018     Station *getBegin() const;
00019
00024     Station *getAnEnd() const;
00025
00026 private:
00027     // Other
00028     bool properlyInitialized() const;
00029
00030     Station *begin;
00031     Station *end;
00032
00033     Track* _initCheck;
00034 };
00035
00036
00037 #endif //PSE_METRO_SIMULATIE_TRACK_H
```

4.9 Tram.h

```
00001 #ifndef PSE_METRO_SIMULATIE_TRAM_H
00002 #define PSE_METRO_SIMULATIE_TRAM_H
00003 #include <ostream>
00004 #include "Station.h"
00005 #include "Track.h"
00006
00007 class Tram {
00008 public:
00009     // Constructor
00010     Tram(int lineNumber, int tramNumber, int speed, Station *startStation);
00011
00012     // Destructor
00013     virtual ~Tram();
00014
00015     // Operators
00020     operator std::string();
00021
00022     // Getters
00027     int getLineNumber() const;
00028
00033     int getSpeed() const;
00034
00039     Station *getStartStation() const;
00040
00045     int getTramNumber() const;
00046
00051     Station *getCurrentStation() const;
00052
00057     void setSpeed(int speed);
00058
00063     void setStartStation(Station *startStation);
00064
00069     void setTramNumber(int tramNumber);
00070
00075     void drive(std::ostream &os);
00076
00077 private:
00078     int lineNumber;
00079     int tramNumber;
00080     int speed;
00081     Station *startStation;
00082     Station *currentStation;
00083
00084     // OS
00085     friend std::ostream &operator<<(std::ostream &os, const Tram &tram);
00086
00087     Tram* _initCheck;
00088
00089     bool properlyInitialized() const;
00090 };
00091
00092
00093 #endif //PSE_METRO_SIMULATIE_TRAM_H
```