VehicleModule

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:

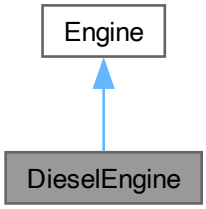
Chapter 4

Class Documentation

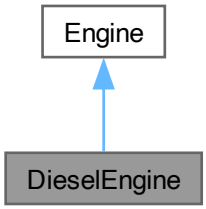
4.1 DieselEngine Class Reference

`#include <Engine.h>`

Inheritance diagram for DieselEngine:



Collaboration diagram for DieselEngine:



Public Member Functions

- DieselEngine (int hp=130)
- std::string getType () const override

  Gauti variklio tipą.
- Engine * clone () const override

  Konstruktorius su variklio tipu.
- int getHorsepower () const override

  Gauti variklio galią.
- void setHorsepower (int hp) override

  Nustatyti variklio galią.
- void save (std::ostream &os) const override

  Išsaugoti variklį į binarinį srautą.
- void load (std::istream &is) override

  Nuskaito variklį iš binarinio srauto.

Public Member Functions inherited from Engine

- virtual ~Engine ()=default

  Konstruktorius be parametrų.

4.1.1 Constructor & Destructor Documentation

4.1.1.1 DieselEngine()

```
DieselEngine::DieselEngine (
            int hp = 130)  [inline]
```

4.1.2 Member Function Documentation

4.1.2.1 clone()

```
Engine * DieselEngine::clone () const  [inline], [override], [virtual]
```

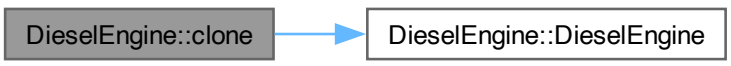
Konstruktorius su variklio tipu.

Parameters

type	Variklio tipas.

Implements Engine.

Here is the call graph for this function:



4.1.2.2 getHorsepower()

```
int DieselEngine::getHorsepower () const  [inline], [override], [virtual]
```

Gauti variklio galią.

Returns

Variklio galia.

Implements Engine.

4.1.2.3 getType()

```
std::string DieselEngine::getType () const  [inline], [override], [virtual]
```

Gauti variklio tipą.

Returns

Variklio tipas.

Implements Engine.

4.1.2.4 load()

```
void DieselEngine::load (
            std::istream & is)  [inline], [override], [virtual]
```

Nuskaito variklį iš binarinio srauto.

Parameters

is	Įvesties srautas.

Implements Engine.

4.1.2.5 save()

```
void DieselEngine::save (
            std::ostream & os) const  [inline], [override], [virtual]
```

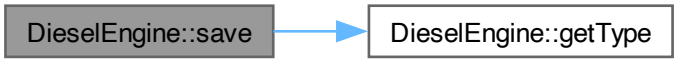
Išsaugoti variklį į binarinį srautą.

Parameters

os	Išvesties srautas.

Implements Engine.

Here is the call graph for this function:



4.1.2.6 setHorsepower()

```
void DieselEngine::setHorsepower (
            int hp)  [inline], [override], [virtual]
```

Nustatyti variklio galią.

Parameters

hp	Variklio galia.

Implements Engine.

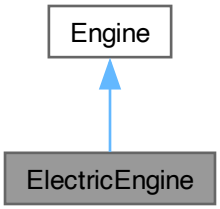
The documentation for this class was generated from the following file:

- Engine.h

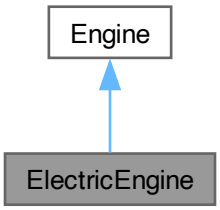
4.2 ElectricEngine Class Reference

```
#include <Engine.h>
```

Inheritance diagram for ElectricEngine:



Collaboration diagram for ElectricEngine:



Public Member Functions

- ElectricEngine (int hp=200)
- std::string getType () const override
  Gauti variklio tipą.
- Engine * clone () const override
  Konstruktorius su variklio tipu.
- int getHorsepower () const override
  Gauti variklio galią.
- void setHorsepower (int hp) override
  Nustatyti variklio galią.
- void save (std::ostream &os) const override
  Išsaugoti variklį į binarinį srautą.
- void load (std::istream &is) override
  Nuskaito variklį iš binarinio srauto.

Public Member Functions inherited from Engine

- virtual ~Engine ()=default
  Konstruktorius be parametrų.

4.2.1 Constructor & Destructor Documentation

4.2.1.1 ElectricEngine()

```
ElectricEngine::ElectricEngine (
            int hp = 200)  [inline]
```

4.2.2 Member Function Documentation

4.2.2.1 clone()

```
Engine * ElectricEngine::clone () const  [inline], [override], [virtual]
```

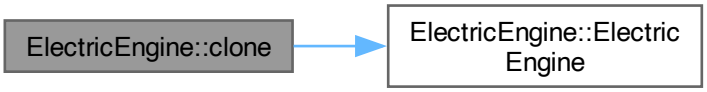
Konstruktorius su variklio tipu.

Parameters

type	Variklio tipas.

Implements Engine.

Here is the call graph for this function:



4.2.2.2 getHorsepower()

```
int ElectricEngine::getHorsepower () const  [inline], [override], [virtual]
```

Gauti variklio galią.

Returns

Variklio galia.

Implements Engine.

4.2.2.3 getType()

```
std::string ElectricEngine::getType () const  [inline], [override], [virtual]
```

Gauti variklio tipą.

Returns

Variklio tipas.

Implements Engine.

4.2.2.4 load()

```
void ElectricEngine::load (
            std::istream & is)  [inline], [override], [virtual]
```

Nuskaito variklį iš binarinio srauto.

Parameters

is	Įvesties srautas.

Implements Engine.

4.2.2.5 save()

```
void ElectricEngine::save (
            std::ostream & os) const  [inline], [override], [virtual]
```

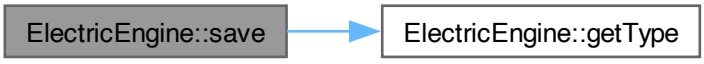
Išsaugoti variklį į binarinį srautą.

Parameters

os	Išvesties srautas.

Implements Engine.

Here is the call graph for this function:



4.2.2.6 setHorsepower()

```
void ElectricEngine::setHorsepower (
            int hp)  [inline], [override], [virtual]
```

Nustatyti variklio galią.

Parameters

hp	Variklio galia.

Implements Engine.

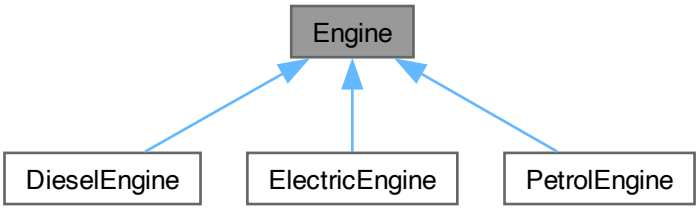
The documentation for this class was generated from the following file:

- Engine.h

4.3 Engine Class Reference

```
#include <Engine.h>
```

Inheritance diagram for Engine:



Public Member Functions

- virtual ~Engine ()=default
  Konstruktorius be parametrų.
- virtual std::string getType () const =0
  Gauti variklio tipą.
- virtual Engine * clone () const =0
  Konstruktorius su variklio tipu.
- virtual int getHorsepower () const =0
  Gauti variklio galią.
- virtual void setHorsepower (int hp)=0
  Nustatyti variklio galią.
- virtual void save (std::ostream &os) const =0
  Išsaugoti variklį į binarinį srautą.
- virtual void load (std::istream &is)=0
  Nuskaito variklį iš binarinio srauto.

4.3.1 Constructor & Destructor Documentation

4.3.1.1 ~Engine()

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

Konstruktorius be parametrų.

4.3.2 Member Function Documentation

4.3.2.1 clone()

```
virtual Engine * Engine::clone () const  [pure virtual]
```

Konstruktorius su variklio tipu.

Parameters

type	Variklio tipas.

Implemented in DieselEngine, ElectricEngine, and PetrolEngine.

4.3.2.2 getHorsepower()

```
virtual int Engine::getHorsepower () const  [pure virtual]
```

Gauti variklio galią.

Returns

Variklio galia.

Implemented in DieselEngine, ElectricEngine, and PetrolEngine.

4.3.2.3 getType()

```
virtual std::string Engine::getType () const  [pure virtual]
```

Gauti variklio tipą.

Returns

Variklio tipas.

Implemented in DieselEngine, ElectricEngine, and PetrolEngine.

4.3.2.4 load()

```
virtual void Engine::load (
            std::istream & is)  [pure virtual]
```

Nuskaito variklį iš binarinio srauto.

Parameters

is	Įvesties srautas.

Implemented in DieselEngine, ElectricEngine, and PetrolEngine.

4.3.2.5 save()

```
virtual void Engine::save (
            std::ostream & os) const  [pure virtual]
```

Išsaugoti variklį į binarinį srautą.

Parameters

os	Išvesties srautas.

Implemented in DieselEngine, ElectricEngine, and PetrolEngine.

4.3.2.6 setHorsepower()

```
virtual void Engine::setHorsepower (
            int hp)  [pure virtual]
```

Nustatyti variklio galią.

Parameters

hp	Variklio galia.

Implemented in DieselEngine, ElectricEngine, and PetrolEngine.

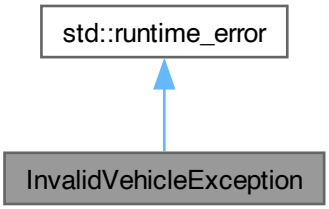
The documentation for this class was generated from the following file:

- Engine.h

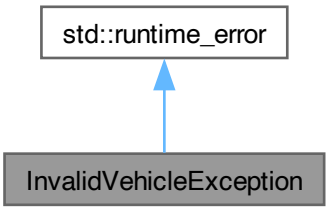
4.4 InvalidVehicleException Class Reference

`#include <Vehicle.h>`

Inheritance diagram for InvalidVehicleException:



Collaboration diagram for InvalidVehicleException:



Public Member Functions

- InvalidVehicleException (const std::string &message)
  Konstruktorius su pranešimu.

4.4.1 Constructor & Destructor Documentation

4.4.1.1 InvalidVehicleException()

```
InvalidVehicleException::InvalidVehicleException (
            const std::string & message)  [inline], [explicit]
```

Konstruktorius su pranešimu.

Parameters

message	Pranešimas apie klaidą.

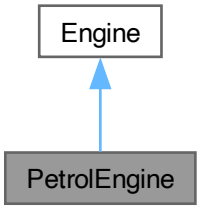
The documentation for this class was generated from the following file:

- Vehicle.h

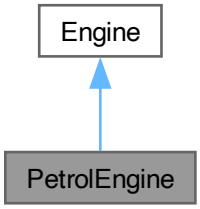
4.5 PetrolEngine Class Reference

`#include <Engine.h>`

Inheritance diagram for PetrolEngine:



Collaboration diagram for PetrolEngine:



Public Member Functions

- PetrolEngine (int hp=150)
- std::string getType () const override
  Gauti variklio tipą.
- Engine * clone () const override
  Konstruktorius su variklio tipu.
- int getHorsepower () const override
  Gauti variklio galią.
- void setHorsepower (int hp) override
  Nustatyti variklio galią.
- void save (std::ostream &os) const override
  Išsaugoti variklį į binarinį srautą.
- void load (std::istream &is) override
  Nuskaito variklį iš binarinio srauto.

Public Member Functions inherited from Engine

- virtual ~Engine ()=default
  Konstruktorius be parametrų.

4.5.1 Constructor & Destructor Documentation

4.5.1.1 PetrolEngine()

```
PetrolEngine::PetrolEngine (
            int hp = 150)  [inline]
```

4.5.2 Member Function Documentation

4.5.2.1 clone()

```
Engine * PetrolEngine::clone () const  [inline], [override], [virtual]
```

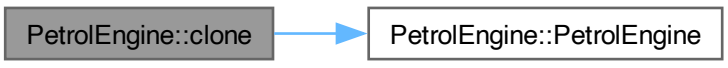
Konstruktorius su variklio tipu.

Parameters

type	Variklio tipas.

Implements Engine.

Here is the call graph for this function:



4.5.2.2 getHorsepower()

```
int PetrolEngine::getHorsepower () const  [inline], [override], [virtual]
```

Gauti variklio galią.

Returns

Variklio galia.

Implements Engine.

4.5.2.3 getType()

```
std::string PetrolEngine::getType () const  [inline], [override], [virtual]
```

Gauti variklio tipą.

Returns

Variklio tipas.

Implements Engine.

4.5.2.4 load()

```
void PetrolEngine::load (
            std::istream & is)  [inline], [override], [virtual]
```

Nuskaito variklį iš binarinio srauto.

Parameters

is	Įvesties srautas.

Implements Engine.

4.5.2.5 save()

```
void PetrolEngine::save (
            std::ostream & os) const  [inline], [override], [virtual]
```

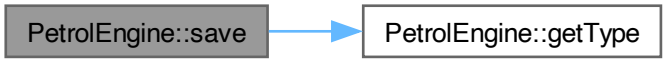
Išsaugoti variklį į binarinį srautą.

Parameters

os	Išvesties srautas.

Implements Engine.

Here is the call graph for this function:



4.5.2.6 setHorsepower()

```
void PetrolEngine::setHorsepower (
            int hp)  [inline], [override], [virtual]
```

Nustatyti variklio galią.

Parameters

hp	Variklio galia.

Implements Engine.

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

- Engine.h

4.6 Vehicle Class Reference

Reprezentuoja transporto priemonę su keičiamu variklio tipu.

```
#include <Vehicle.h>
```

Public Member Functions

- Vehicle ()
  Konstruktorius be parametrų.
- Vehicle (const std::string &brand)
  Konstruktorius su prekės ženklu.
- Vehicle (const std::string &brand, const std::string &color, int year, const std::string &vin)
  Konstruktorius su prekės ženklu, spalva, gamybos metais ir VIN numeriu.
- Vehicle (const Vehicle &other)
  Konstruktorius kopijai.
- Vehicle & operator= (const Vehicle &other)
  Priskyrimo operatorius.
- ~Vehicle ()
  Destruktorius.
- std::string getBrand () const
  Gauti prekės ženklą.
- std::string getColor () const
  Gauti spalvą.
- int getYear () const
  Gauti gamybos metus.
- std::string getVin () const
  Gauti VIN numerį.
- std::string toString () const
  Gauti objekto aprašymą kaip tekstą.
- void setBrand (const std::string &brand)
  Nustatyti prekės ženklą.

- void setColor (const std::string &color)

  Nustatyti spalvą.
- void setYear (int year)

  Nustatyti gamybos metus.
- void setVin (const std::string &vin)

  Nustatyti VIN numerį.
- void setEngine (class Engine *engine)

  Nustatyti variklį.
- void switchEngine (int type)

  Keisti variklį.

Static Public Member Functions

- static int getObjectsCount ()

  Gauti objekto skaičių.

Friends

- std::ostream & operator<< (std::ostream &os, const Vehicle &v)

  Įrašo objektą į binarinį srautą.
- std::istream & operator>> (std::istream &is, Vehicle &v)

  Nuskaito objektą iš binarinio srauto.

4.6.1 Detailed Description

Reprezentuoja transporto priemonę su keičiamu variklio tipu.

Naudoja Pimpl idiomą. Leidžia keisti variklio (Engine) realizaciją vykdymo metu.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 Vehicle() [1/4]

```
Vehicle::Vehicle ()
```

Konstruktorius be parametrų.

4.6.2.2 Vehicle() [2/4]

```
Vehicle::Vehicle (
            const std::string & brand)
```

Konstruktorius su prekės ženklu.

Parameters

brand	Prekės ženklas.

4.6.2.3 Vehicle() [3/4]

```
Vehicle::Vehicle (
            const std::string & brand,
            const std::string & color,
            int year,
            const std::string & vin)
```

Konstruktorius su prekės ženklu, spalva, gamybos metais ir VIN numeriu.

Parameters

brand	Prekės ženklas.
color	Spalva.
year	Gamybos metai.
vin	VIN numeris.

4.6.2.4 Vehicle() [4/4]

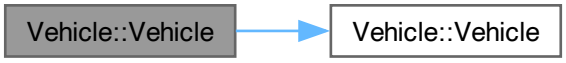
```
Vehicle::Vehicle (
            const Vehicle & other)
```

Konstruktorius kopijai.

Parameters

other	Kitas objektas.

Here is the call graph for this function:



4.6.2.5 ~Vehicle()

```
Vehicle::~Vehicle ()
```

Destruktorius.

4.6.3 Member Function Documentation

4.6.3.1 getBrand()

```
std::string Vehicle::getBrand () const
```

Gauti prekės ženklą.

Returns

Prekės ženklas.

4.6.3.2 getColor()

```
std::string Vehicle::getColor () const
```

Gauti spalvą.

Returns

Spalva.

4.6.3.3 getObjectsCount()

```
int Vehicle::getObjectsCount ()  [static]
```

Gauti objekto skaičių.

Returns

Objekto skaičius.

4.6.3.4 getVin()

```
std::string Vehicle::getVin () const
```

Gauti VIN numerį.

Returns

VIN numeris.

4.6.3.5 getYear()

```
int Vehicle::getYear () const
```

Gauti gamybos metus.

Returns

Gamybos metai.

4.6.3.6 operator=()

```
Vehicle & Vehicle::operator= (
            const Vehicle & other)
```

Priskyrimo operatorius.

Parameters

other	Kitas objektas.

Returns

Nuoroda į šį objektą.

Here is the call graph for this function:

4.6.3.7 setBrand()

```
void Vehicle::setBrand (
            const std::string & brand)
```

Nustatyti prekės ženklą.

Parameters

brand	Prekės ženklas.

4.6.3.8 setColor()

```
void Vehicle::setColor (
            const std::string & color)
```

Nustatyti spalvą.

Parameters

color	Spalva.

4.6.3.9 setEngine()

```
void Vehicle::setEngine (
            class Engine * engine)
```

Nustatyti variklį.

Parameters

engine	Variklis.

4.6.3.10 setVin()

```
void Vehicle::setVin (
            const std::string & vin)
```

Nustatyti VIN numerį.

Parameters

vin	VIN numeris.

4.6.3.11 setYear()

```
void Vehicle::setYear (
            int year)
```

Nustatyti gamybos metus.

Parameters

year	Gamybos metai.

4.6.3.12 switchEngine()

```
void Vehicle::switchEngine (
            int type)
```

Keisti variklį.

Parameters

type	Variklio tipas (0 - benzinas, 1 - dyzelinas, 2 - elektra).

4.6.3.13 toString()

```
std::string Vehicle::toString () const
```

Gauti objekto aprašymą kaip tekstą.

Returns

Objekto aprašymas.

Here is the call graph for this function:

4.6.4 Friends And Related Symbol Documentation

4.6.4.1 operator<<

```
std::ostream & operator<< (
            std::ostream & os,
            const Vehicle & v)  [friend]
```

Įrašo objektą į binarinį srautą.

4.6.4.2 operator>>

```
std::istream & operator>> (
            std::istream & is,
            Vehicle & v)  [friend]
```

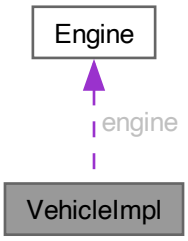
Nuskaito objektą iš binarinio srauto.

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

- Vehicle.h
- Vehicle.cpp

4.7 VehicleImpl Class Reference

Collaboration diagram for VehicleImpl:



Public Member Functions

- VehicleImpl ()
- VehicleImpl (const std::string &b, const std::string &c, int y, const std::string &v, Engine ∗eng=nullptr)
- VehicleImpl (const VehicleImpl &other)
- ∼VehicleImpl ()
- void validate () const
- std::string toString () const
- void setEngine (Engine ∗newEngine)
- void switchEngine (int type)
- void save (std::ostream &os) const
- void load (std::istream &is)

Public Attributes

- std::string brand
- std::string color
- int year
- std::string vin
- int id
- Engine ∗ engine

Static Public Attributes

- static int idCounter = 1
- static int objectsCount = 0

4.7.1 Constructor & Destructor Documentation

4.7.1.1 VehicleImpl() [1/3]

```
VehicleImpl::VehicleImpl ()  [inline]
```

4.7.1.2 VehicleImpl() [2/3]

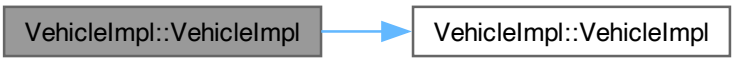
```
VehicleImpl::VehicleImpl (
            const std::string & b,
            const std::string & c,
            int y,
            const std::string & v,
            Engine * eng = nullptr)  [inline]
```

Here is the call graph for this function:

4.7.1.3 VehicleImpl() [3/3]

```
VehicleImpl::VehicleImpl (
            const VehicleImpl & other)  [inline]
```

Here is the call graph for this function:



4.7.1.4 ~VehicleImpl()

```
VehicleImpl::~VehicleImpl ()  [inline]
```

4.7.2 Member Function Documentation

4.7.2.1 load()

```
void VehicleImpl::load (
            std::istream & is)  [inline]
```

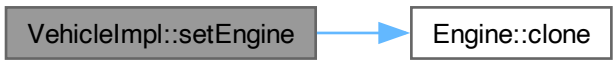
4.7.2.2 save()

```
void VehicleImpl::save (
            std::ostream & os) const  [inline]
```

4.7.2.3 setEngine()

```
void VehicleImpl::setEngine (
            Engine * newEngine)  [inline]
```

Here is the call graph for this function:



4.7.2.4 switchEngine()

```
void VehicleImpl::switchEngine (
            int type)  [inline]
```

4.7.2.5 toString()

```
std::string VehicleImpl::toString () const  [inline]
```

4.7.2.6 validate()

```
void VehicleImpl::validate () const  [inline]
```

4.7.3 Member Data Documentation

4.7.3.1 brand

```
std::string VehicleImpl::brand
```

4.7.3.2 color

```
std::string VehicleImpl::color
```

4.7.3.3 engine

```
Engine* VehicleImpl::engine
```

4.7.3.4 id

```
int VehicleImpl::id
```

4.7.3.5 idCounter

```
int VehicleImpl::idCounter = 1  [static]
```

4.7.3.6 objectsCount

```
int VehicleImpl::objectsCount = 0  [static]
```

4.7.3.7 vin

```
std::string VehicleImpl::vin
```

4.7.3.8 year

```
int VehicleImpl::year
```

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

- Vehicle.cpp

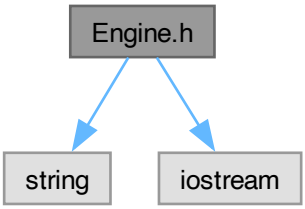
Chapter 5

File Documentation

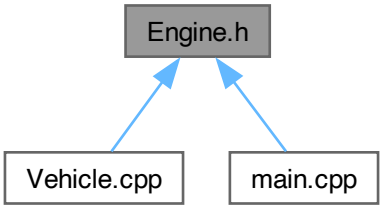
5.1 Engine.h File Reference

```
#include <string>
#include <iostream>
```

Include dependency graph for Engine.h:



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



Classes

- class Engine
- class PetrolEngine
- class DieselEngine
- class ElectricEngine

5.2 Engine.h

Go to the documentation of this file.

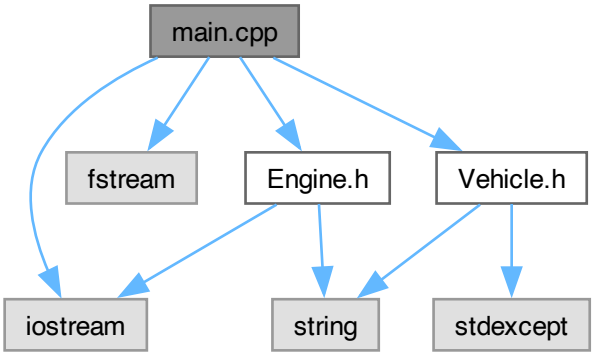
```
00001 #ifndef ENGINE_H
00002 #define ENGINE_H
00003
00004 #include <string>
00005 #include <iostream>
00006
00007 class Engine {
00008 public:
00012     virtual ~Engine() = default;
00013
00018     virtual std::string getType() const = 0;
00019
00024     virtual Engine* clone() const = 0;
00025
00030     virtual int getHorsepower() const = 0;
00031
00036     virtual void setHorsepower(int hp) = 0;
00037
00042     virtual void save(std::ostream& os) const = 0;
00043
00048     virtual void load(std::istream& is) = 0;
00049 };
00050
00051 class PetrolEngine : public Engine {
00052     int horsepower;
00053 public:
00054     PetrolEngine(int hp = 150) : horsepower(hp) {}
00055     std::string getType() const override { return "Petrol"; }
00056     Engine* clone() const override { return new PetrolEngine(*this); }
00057
00058     int getHorsepower() const override { return horsepower; }
00059     void setHorsepower(int hp) override { horsepower = hp; }
00060
00061     void save(std::ostream& os) const override {
00062         int typeLength = getType().length();
00063         os.write((char*)&typeLength, sizeof(typeLength));
00064         os.write(getType().c_str(), typeLength);
00065         os.write((char*)&horsepower, sizeof(horsepower));
00066     }
00067
00068     void load(std::istream& is) override {
00069         is.read((char*)&horsepower, sizeof(horsepower));
00070     }
00071
00072 };
00073
00074 class DieselEngine : public Engine {
00075     int horsepower;
00076 public:
00077     DieselEngine(int hp = 130) : horsepower(hp) {}
00078     std::string getType() const override { return "Diesel"; }
00079     Engine* clone() const override { return new DieselEngine(*this); }
00080
00081     int getHorsepower() const override { return horsepower; }
00082     void setHorsepower(int hp) override { horsepower = hp; }
00083
00084     void save(std::ostream& os) const override {
00085         int typeLength = getType().length();
00086         os.write((char*)&typeLength, sizeof(typeLength));
00087         os.write(getType().c_str(), typeLength);
00088         os.write((char*)&horsepower, sizeof(horsepower));
00089     }
00090
00091     void load(std::istream& is) override {
00092         is.read((char*)&horsepower, sizeof(horsepower));
00093     }
00094
00095 };
```

```
00096
00097 class ElectricEngine : public Engine {
00098     int horsepower;
00099 public:
00100     ElectricEngine(int hp = 200) : horsepower(hp) {}
00101     std::string getType() const override { return "Electric"; }
00102     Engine* clone() const override { return new ElectricEngine(*this); }
00103
00104     int getHorsepower() const override { return horsepower; }
00105     void setHorsepower(int hp) override { horsepower = hp; }
00106
00107     void save(std::ostream& os) const override {
00108         int typeLength = getType().length();
00109         os.write((char*)&typeLength, sizeof(typeLength));
00110         os.write(getType().c_str(), typeLength);
00111         os.write((char*)&horsepower, sizeof(horsepower));
00112     }
00113
00114     void load(std::istream& is) override {
00115         is.read((char*)&horsepower, sizeof(horsepower));
00116     }
00117
00118 };
00119
00120 #endif // ENGINE_H
```

5.3 main.cpp File Reference

```
#include <iostream>
#include <fstream>
#include "Vehicle.h"
#include "Engine.h"
```

Include dependency graph for main.cpp:



Functions

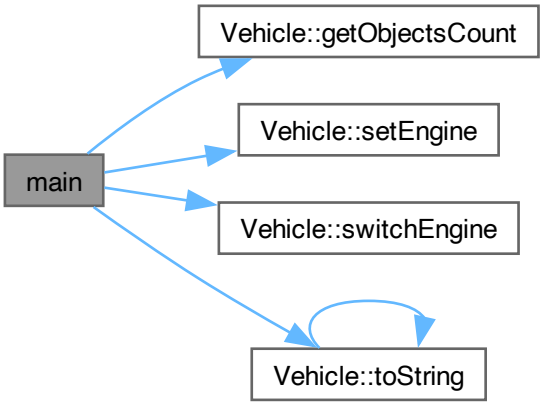
- int main ()

5.3.1 Function Documentation

5.3.1.1 main()

```
int main ()
```

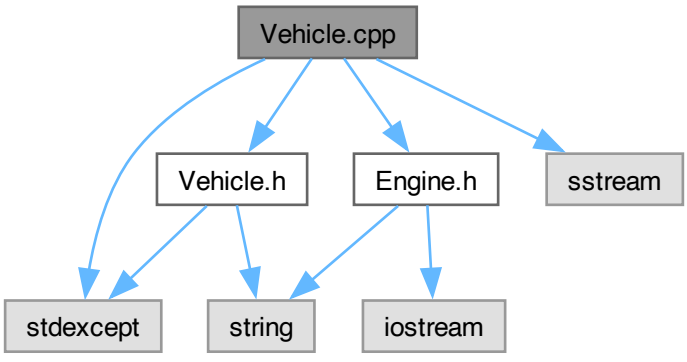
Here is the call graph for this function:



5.4 Vehicle.cpp File Reference

```
#include "Vehicle.h"
#include "Engine.h"
#include <sstream>
#include <stdexcept>
```

Include dependency graph for Vehicle.cpp:



Classes

- class VehicleImpl

Functions

- std::ostream & operator<< (std::ostream &os, const Vehicle &v)
- std::istream & operator>> (std::istream &is, Vehicle &v)

5.4.1 Function Documentation

5.4.1.1 operator<<()

```
std::ostream & operator<< (
            std::ostream & os,
            const Vehicle & v)
```

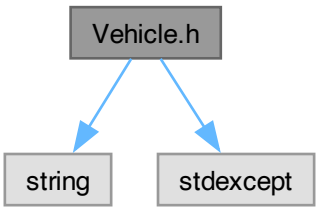
5.4.1.2 operator>>()

```
std::istream & operator>> (
            std::istream & is,
            Vehicle & v)
```

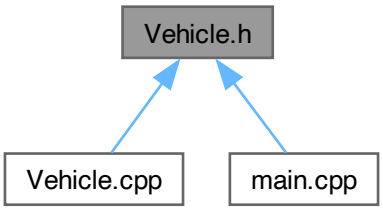
5.5 Vehicle.h File Reference

```
#include <string>
#include <stdexcept>
```

Include dependency graph for Vehicle.h:



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



Classes

- class Vehicle

  Reprezentuoja transporto priemonę su keičiamu variklio tipu.
- class InvalidVehicleException

Macros

- #define INVALID_VEHICLE_EXCEPTION_H

5.5.1 Macro Definition Documentation

5.5.1.1 INVALID_VEHICLE_EXCEPTION_H

```
#define INVALID_VEHICLE_EXCEPTION_H
```

5.6 Vehicle.h

Go to the documentation of this file.

```
00001 #ifndef VEHICLE_H
00002 #define VEHICLE_H
00003
00004 #include <string>
00005
00006 class VehicleImpl;
00007
00014
00015 class Vehicle {
00016 public:
00020     Vehicle();
00021
00026     Vehicle(const std::string& brand);
00027
00035     Vehicle(const std::string& brand, const std::string& color, int year, const std::string& vin);
00036
00041     Vehicle(const Vehicle& other);
00042
00048     Vehicle& operator=(const Vehicle& other);
00049
00053     ~Vehicle();
00054
00059     std::string getBrand() const;
00060
00065     std::string getColor() const;
00066
00071
00072     int getYear() const;
00073
00078     std::string getVin() const;
00079
00084     std::string toString() const;
00085
00090     void setBrand(const std::string& brand);
00091
00096     void setColor(const std::string& color);
00097
00102     void setYear(int year);
00103
00108     void setVin(const std::string& vin);
00109
00114     static int getObjectsCount();
00115
00120     void setEngine(class Engine* engine);
00121
00126     void switchEngine(int type);
00127
00128
00132     friend std::ostream& operator<<(std::ostream& os, const Vehicle& v);
```

```
00133
00137     friend std::istream& operator»(std::istream& is, Vehicle& v);
00138
00139 private:
00140     VehicleImpl* impl; // Pointer to implementation
00141 };
00142
00143 #endif // VEHICLE_H
00144
00145
00146 #ifndef INVALID_VEHICLE_EXCEPTION_H
00147 #define INVALID_VEHICLE_EXCEPTION_H
00148
00149 #include <stdexcept>
00150 #include <string>
00151
00152
00153 class InvalidVehicleException : public std::runtime_error {
00154 public:
00159     explicit InvalidVehicleException(const std::string& message)
00160         : std::runtime_error("InvalidVehicleException: " + message) {}
00161 };
00162
00163 #endif
00164
```

Index