SDP - Uebung 1

Wintersemester 2019/20

Adam Kensy - S1810306018
Philipp Holzer - S1810306028

25. Oktober 2019

Inhaltsverzeichnis

1 Organisatorisches

1.1 Team

- Philipp Holzer, Matr.-Nr.: 1810306028
- Adam Kensy, Matr.-Nr.: 1810306018

1.2 Aufteilung und Verantwortlichkeitsbereiche

- Philipp Holzer
  - Planung
  - Klassendiagramm
  - Implementierung und Testen der Klassen
    - Logbook
    - Vehicle
  - Dokumentation
- Adam Kensy
  - Planung
  - Klassendiagramm
  - Implementierung und Testen der Klassen
    - Carpool
    - Vehicle
    - Car, Truck, Motorcycle
  - Dokumentation

1.3 Aufwand

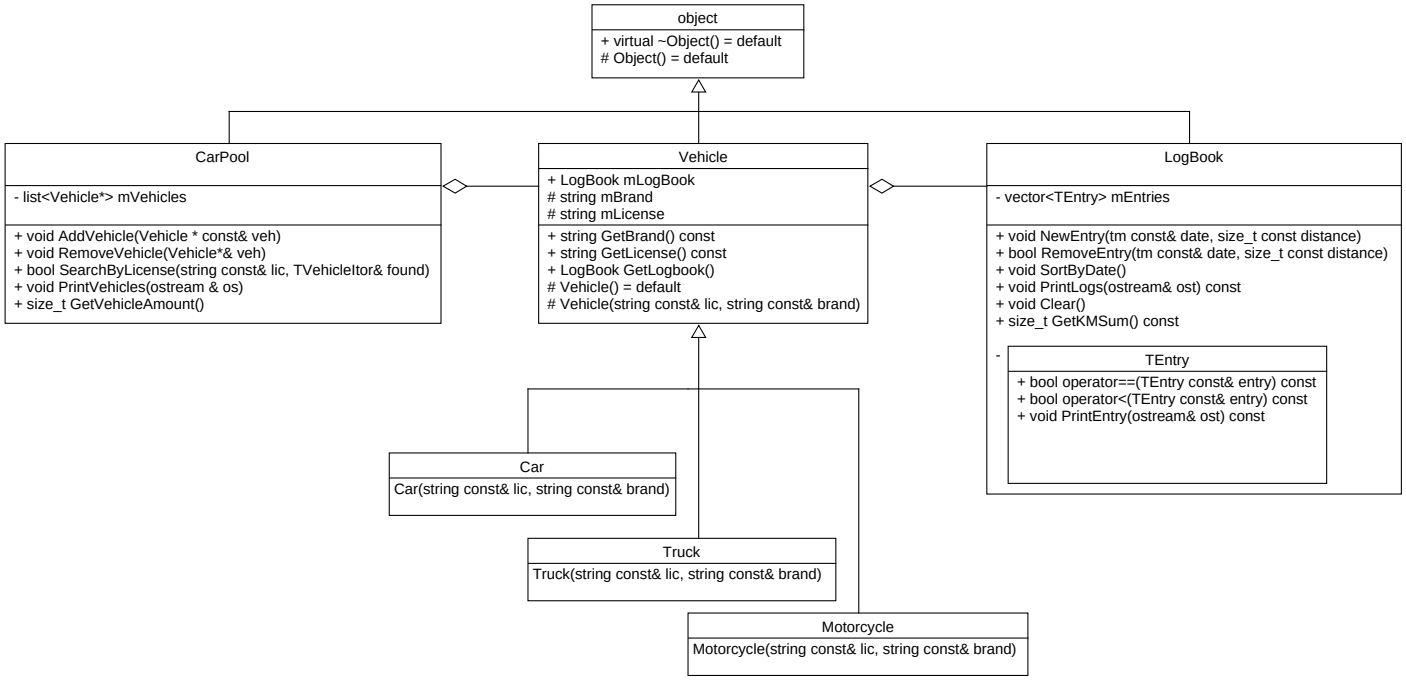
- Philipp Holzer geschätzt: 7 tatsächlich: 8
- Adam Kensy geschätzt: 7 tatsächlich: 5

2 Anforderungsdefinition (Systemspezifikation)

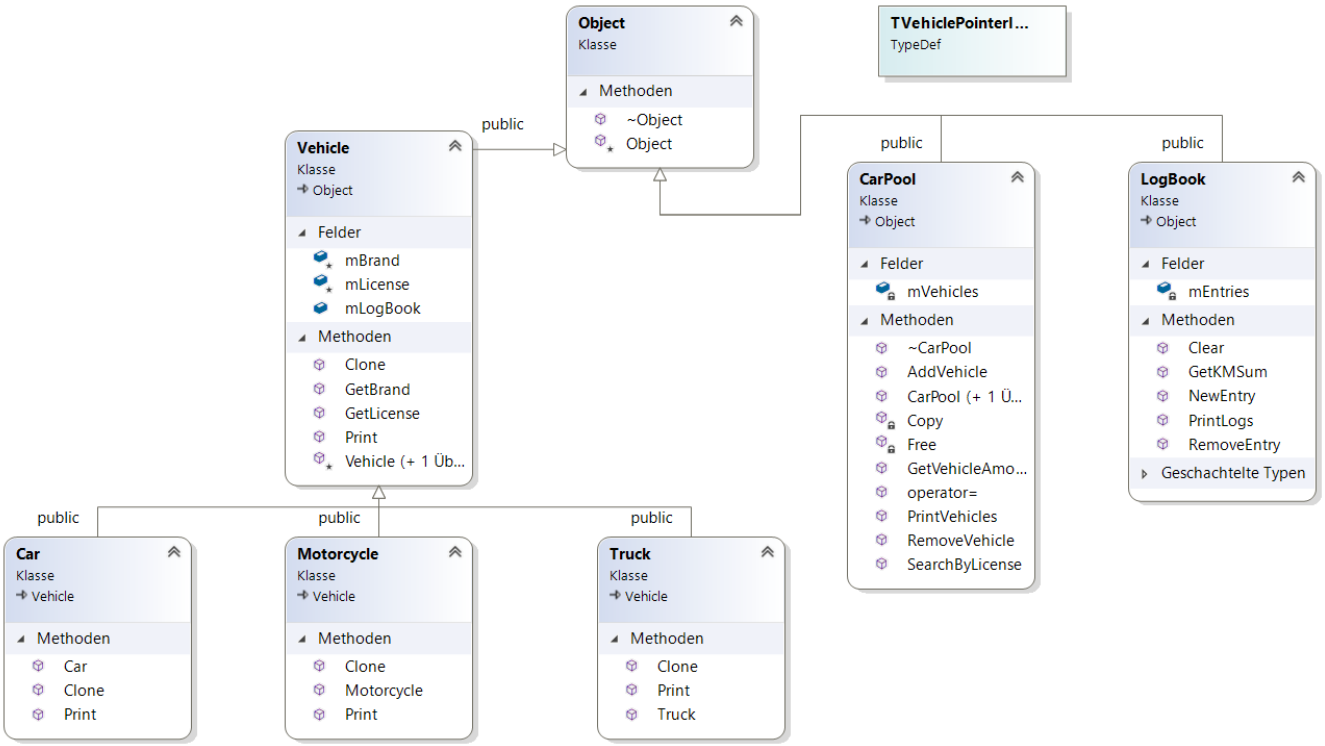
Gesucht ist ein Fuhrpark wo man verschiedene Fahrzeuge verwalten kann. Die Fahrzeuge besitzen jeweils ein Kennzeichen, Marke und ein Fahrtenbuch. Der Fuhrpark hat keine Begrenzung bei der Menge an Fahrzeugen, jedoch kann jedes Kennzeichen nur einmal vorkommen.

3 Systementwurf

3.1 Klassendiagramm



3.2 Klassendiagramm durch Visual Studio



3.3 Komponentenübersicht

- **Klasse "Object"**
  Basis aller Klassen
- **Klasse "Carpool"**
  Verwaltet alle Fahrzeuge
  Besitzt eine Ausgabefunktion um alle enthaltenen Fahrzeuge auszugeben
- **Klasse "Logbook"**
  Das Fahrtenbuch der Fahrzeuge
  Vor der Ausgabe wird das Fahrtenbuch immer nach Datum sortiert
- **Klasse "Vehicle"**
  Stellt die Fahrzeuge dar, dazu gehören: PKW, LKW, Motorräder
- **Klasse "Car", "Truck", "Motorcycle"**
  Konkrete Objekte für die Fahrzeuge
  Besitzen nur eine Ausgabefunktion, wobei der Ausgabeoperator überschrieben ist

3.4 Designentscheidungen

- Es wurde keine EBNF erstellt da es international anwendbar sein soll.
- Der Fuhrpark redet mit uns über die Konsole wenn etwas schiefläuft -¿ wenn kein Fahrzeug hinzugefügt oder entfernt werden konnte bekommen wird eine Fehlermeldung aber ansonsten bleibt der Fuhrpark ruhig.

- Wir haben uns entschieden, dass das Einfügen und Entfernen über Pointer läuft - wir dachten uns, es macht so am meisten Sinn denn man parkt das Fahrzeug irgendwo und teilt dies dann der Verwaltung mit.
- Dadurch, dass wir mit Pointer arbeiten haben wir beim Carpool die Rule-Of-Three angewandt.
- Wir benutzen die tm-Struktur für Datums-Angaben und erwarten uns eine sinnvolle Eingabe vom Anwender.
- Es wird sortiert ins Fahrtenbuch eingetragen, dies macht am meisten Sinn, da man dies direkt nach einer Fahrt macht.

4 Testprotokollierung

4.1 Testumgebung

Microsoft Visual Studio Enterprise 2019
Version 16.3.5
Microsoft Visual C++ 2019

Windows 10, 64Bit, Build 18362

Testdriver: main.cpp

4.2 Testausgabe

```
Visual Leak Detector read settings from: C:\Program Files (x86)\Visual Leak Detector\vld.ini
Visual Leak Detector Version 2.5.1 installed.
**********************************
Testsection LogBook
**********************************
24.3.1993:       43 km
12.1.2008:       17 km
31.7.2019:       21 km
17.9.2019:       89 km
17.9.2019:       89 km
17.9.2019:      110 km
4.12.2019:       43 km

Entry got removed:
-----------------------------------
24.3.1993:       43 km
12.1.2008:       17 km
31.7.2019:       21 km
17.9.2019:       89 km
17.9.2019:      110 km
4.12.2019:       43 km

 Removing a nonexisting entry:
-----------------------------------
Entry does not exist! Couldn't delete entry.

Print km-sum:
-----------------------------------
323

Log cleared (for the next testcases it got filled again):
-----------------------------------

**********************************
Testsection CarPool
**********************************
All added vehicles:
-----------------------------------
Fahrzeugart:  Motorrad
Marke:        Kawazaki
Kennzeichen:  LL-HAGE1

Fahrzeugart:  LKW
Marke:        Mercedes
Kennzeichen:  LL-HARD3
24.3.1993:       98 km
12.1.2008:       67 km
31.7.2019:       45 km
17.9.2019:       21 km
4.12.2019:       34 km

Fahrzeugart:  PKW
```

```
Marke:         Opel Corsa (nagelneu)
Kennzeichen:   LL-ISSO5
24.3.1993:      927 km
12.1.2008:      231 km
31.7.2019:     7028 km
17.9.2019:     6837 km
4.12.2019:      211 km

Fahrzeugart:   Motorrad
Marke:         Yamaha
Kennzeichen:   LL-BERG2
24.3.1993:       23 km
12.1.2008:      432 km
31.7.2019:       26 km
17.9.2019:       45 km
4.12.2019:      117 km

Fahrzeugart:   LKW
Marke:         Koenigsegg
Kennzeichen:   LL-WARE4
24.3.1993:    82456 km
12.1.2008:     4567 km
31.7.2019:     4332 km
17.9.2019:     4321 km
4.12.2019:     6789 km

Fahrzeugart:   PKW
Marke:         Opel Corsa (verrostet und ohne Klima)
Kennzeichen:   LL-C000L
24.3.1993:     1093 km
12.1.2008:      265 km
31.7.2019:      483 km
17.9.2019:     7392 km
4.12.2019:       46 km

Adams Car (LL-ISSO5) got removed:
-----------------------------------
Fahrzeugart:   Motorrad
Marke:         Kawazaki
Kennzeichen:   LL-HAGE1

Fahrzeugart:   LKW
Marke:         Mercedes
Kennzeichen:   LL-HARD3
24.3.1993:       98 km
12.1.2008:       67 km
31.7.2019:       45 km
17.9.2019:       21 km
4.12.2019:       34 km

Fahrzeugart:   Motorrad
Marke:         Yamaha
Kennzeichen:   LL-BERG2
24.3.1993:       23 km
12.1.2008:      432 km
31.7.2019:       26 km
17.9.2019:       45 km
4.12.2019:      117 km

Fahrzeugart:   LKW
Marke:         Koenigsegg
Kennzeichen:   LL-WARE4
24.3.1993:    82456 km
12.1.2008:     4567 km
31.7.2019:     4332 km
17.9.2019:     4321 km
4.12.2019:     6789 km

Fahrzeugart:   PKW
Marke:         Opel Corsa (verrostet und ohne Klima)
Kennzeichen:   LL-C000L
24.3.1993:     1093 km
12.1.2008:      265 km
```

```
31.7.2019:     483 km
17.9.2019:    7392 km
4.12.2019:      46 km

Print LKW with license 'LL-HARD3' (SearchByLicense):
-----------------------------------
Fahrzeugart:  LKW
Marke:        Mercedes
Kennzeichen:  LL-HARD3
24.3.1993:      98 km
12.1.2008:      67 km
31.7.2019:      45 km
17.9.2019:      21 km
4.12.2019:      34 km

Number of vehicles in the car pool:
5

Print vehicles of copied object (Copy CTOR):
-----------------------------------
Fahrzeugart:  Motorrad
Marke:        Kawazaki
Kennzeichen:  LL-HAGE1

Fahrzeugart:  LKW
Marke:        Mercedes
Kennzeichen:  LL-HARD3
24.3.1993:      98 km
12.1.2008:      67 km
31.7.2019:      45 km
17.9.2019:      21 km
4.12.2019:      34 km

Fahrzeugart:  Motorrad
Marke:        Yamaha
Kennzeichen:  LL-BERG2
24.3.1993:      23 km
12.1.2008:     432 km
31.7.2019:      26 km
17.9.2019:      45 km
4.12.2019:     117 km

Fahrzeugart:  LKW
Marke:        Koenigsegg
Kennzeichen:  LL-WARE4
24.3.1993:   82456 km
12.1.2008:    4567 km
31.7.2019:    4332 km
17.9.2019:    4321 km
4.12.2019:    6789 km

Fahrzeugart:  PKW
Marke:        Opel Corsa (verrostet und ohne Klima)
Kennzeichen:  LL-C000L
24.3.1993:    1093 km
12.1.2008:     265 km
31.7.2019:     483 km
17.9.2019:    7392 km
4.12.2019:      46 km

Remove vehicle with license 'LL-HARD3' and assign PandA_Solutions to carpool1 (assignment operator):
-----------------------------------
Fahrzeugart:  Motorrad
Marke:        Kawazaki
Kennzeichen:  LL-HAGE1

Fahrzeugart:  Motorrad
Marke:        Yamaha
Kennzeichen:  LL-BERG2
24.3.1993:      23 km
12.1.2008:     432 km
31.7.2019:      26 km
17.9.2019:      45 km
```

```
4.12.2019:     117 km

Fahrzeugart:  LKW
Marke:        Koenigsegg
Kennzeichen:  LL-WARE4
24.3.1993:   82456 km
12.1.2008:    4567 km
31.7.2019:    4332 km
17.9.2019:    4321 km
4.12.2019:    6789 km

Fahrzeugart:  PKW
Marke:        Opel Corsa (verrostet und ohne Klima)
Kennzeichen:  LL-C000L
24.3.1993:    1093 km
12.1.2008:     265 km
31.7.2019:     483 km
17.9.2019:    7392 km
4.12.2019:      46 km

No memory leaks detected.
Visual Leak Detector is now exiting.

C:\Users\kensy\Google Drive\Hardware-Software-Design\3-Semester\SDP3\Uebung\Fuhrpark\Fuhrpark\
    CarPool\x64\Debug\CarPool.exe (Prozess "18932") wurde mit Code "0" beendet.
Um die Konsole beim Beenden des Debuggens automatisch zu schließen, aktivieren Sie "Extras" > "
    Optionen" > "Debuggen" > "Konsole␣beim␣Beenden␣des␣Debuggings␣automatisch␣schließen".
Drücken Sie eine beliebige Taste, um dieses Fenster zu schließen.
```

5 Quellcode

5.1 Object

Listing 1: CarPool/Object.h

```
1  ////////////////////////////////////////////////////////////////////////////
2  // Workfile : Object.cpp
3  // Author : Philipp Holzer / Adam Kensy
4  // Date : 15.10.2019
5  // Description : common base class
6  // Remarks : -
7  // Revision : 0
8  ////////////////////////////////////////////////////////////////////////////
9
10 #ifndef OBJECT_H
11 #define OBJECT_H
12
13 class Object
14 {
15 protected:
16   Object() = default;
17
18 public:
19
20   virtual ~Object() = default;
21 };
22
23 #endif
```

Listing 2: CarPool/Object.cpp

5.2 CarPool

Listing 3: ./CarPool/CarPool/CarPool.h

```
1  ////////////////////////////////////////////////////////////////////////////
```

```
2  // Workfile : LogBook.cpp
3  // Author : Philipp Holzer / Adam Kensy
4  // Date : 15.10.2019
5  // Description : car pool system which includes different vehicle types
6  // Remarks : -
7  // Revision : 0
8  ////////////////////////////////////////////////////////////////////////
9
10 #ifndef CARPOOL_H
11 #define CARPOOL_H
12
13 #include "../Object.h"
14 #include "../Vehicles/Vehicle.h"
15 #include "../Vehicles/Car.h"
16 #include "../Vehicles/Truck.h"
17 #include "../Vehicles/Motorcycle.h"
18 #include <list>
19
20 typedef std::list<Vehicle*>::iterator TVehiclePointerItor;
21
22 class CarPool : public Object
23 {
24 public:
25   //DTor for CarPool
26   ~CarPool() override;
27
28   //Default CTOR
29   CarPool() = default;
30
31   //Copy-CTOR
32   CarPool(CarPool const& cp);
33
34   //assignmentoperator
35   void operator=(CarPool const& cp);
36
37   //Adds a new car to the carpool
38   //param c: an existing car object
39   void AddVehicle(Vehicle * v);
40
41   //Removes an existing vehicle out of the carpool
42   //param veh: a vehicle that should be in the carpool
43   void RemoveVehicle(std::string const& license);
44
45   //searches through the CarPool for an existing vehicle
46   //param lic: license plate number of the vehicle
47   //param found: iterator which points on the found vehicle
48   //return: true if a vehicle was found else false
49   bool SearchByLicense(std::string const& lic, TVehiclePointerItor& found);
50
51   //prints the info of all vehicles in the carpool
52   void PrintVehicles(std::ostream & os) const;
53
54   //get function for amount of vehicles
55   size_t GetVehicleAmount();
56 private:
57
58   //container which includes all vehicles
59   std::list<Vehicle*> mVehicles;
60
61   //Helpfunction for DTOR and assignment operator
62   //param cp: the copied/assigned CarPool
63   void Copy(CarPool const& cp);
64
65   //Helpfunction for DTOR and assignment operator
66   //Frees all allocated memory
67   void Free();
68
69 };
70
71 #endif
```

Listing 4: ./CarPool/CarPool/CarPool.cpp

```
1  ////////////////////////////////////////////////////////////////////////
2  // Workfile : LogBook.cpp
3  // Author : Philipp Holzer / Adam Kensy
4  // Date : 15.10.2019
5  // Description : car pool system which includes different vehicle types
6  // Remarks : -
7  // Revision : 0
8  ////////////////////////////////////////////////////////////////////////
9  #include "../CarPool/CarPool.h"
10 #include <algorithm>
11 #include <iostream>
12
13 static const std::string cErrLicenseAlreadyExists = "License already exists! Couldn't add
       vehicle.";
14 static const std::string cErrVehicleDoesNotExist = "Vehicle does not exist! Couldn't remove
       vehicle.";
15
16 CarPool::~CarPool()
17 {
18   Free();
19 }
20
21 CarPool::CarPool(CarPool const& cp)
22 {
23   Copy(cp);
24 }
25
26 void CarPool::operator=(CarPool const& cp)
27 {
28   if (this != &cp)
29   {
30     Free();
31     Copy(cp);
32   }
33 }
34
35 void CarPool::AddVehicle(Vehicle * v)
36 {
37   TVehiclePointerItor it;
38   if (!SearchByLicense(v->GetLicense(), it))
39   {
40     mVehicles.emplace_back(v);
41   }
42   else
43   {
44     std::cerr << cErrLicenseAlreadyExists << std::endl;
45   }
46 }
47
48 void CarPool::RemoveVehicle(std::string const& license)
49 {
50   TVehiclePointerItor it;
51   if (SearchByLicense(license, it))
52   {
53     delete* it;
54     *it = nullptr;
55     mVehicles.erase(it);
56   }
57   else
58   {
59     std::cerr << cErrVehicleDoesNotExist << std::endl;
60   }
61 }
62
63 bool CarPool::SearchByLicense(std::string const& lic, TVehiclePointerItor & found)
64 {
65   auto compByLicense = [&](auto * v) { return v->GetLicense() == lic; };
66   found = std::find_if(mVehicles.begin(), mVehicles.end(), compByLicense);
67   if (found != mVehicles.cend())
68   {
69     return true;
```

```
  }
  else
  {
    return false;
  }
}

void CarPool::PrintVehicles(std::ostream & os) const
{
  if (!os.good())
  {
    std::cerr << "error write stream" << std::endl;
  }
  for (auto it = mVehicles.cbegin(); it != mVehicles.cend(); ++it)
  {
    (*it)->Print(os);
  }
}

size_t CarPool::GetVehicleAmount()
{
  return mVehicles.size();
}

void CarPool::Copy(CarPool const& cp)
{
  for (auto it = cp.mVehicles.cbegin(); it != cp.mVehicles.cend(); ++it)
  {
    AddVehicle((*it)->Clone());
  }
}

void CarPool::Free()
{
  for (auto it = mVehicles.begin(); it != mVehicles.end(); ++it)
  {
    delete* it;
    *it = nullptr;
  }

  mVehicles.clear();
}
```

5.3 Vehicles

Listing 5: ./CarPool/Vehicles/Vehicle.h

```
////////////////////////////////////////////////////////////////////////
// Workfile : Vehicle.h
// Author : Philipp Holzer / Adam Kensy
// Date : 15.10.2019
// Description : vehicle class
// Remarks : -
// Revision : 0
////////////////////////////////////////////////////////////////////////

#ifndef VEHICLE_H
#define VEHICLE_H
#include "../LogBook/LogBook.h"
#include "../Object.h"

class Vehicle : public Object
{
public:

  LogBook mLogBook;

  //Get-function for brand
  std::string GetBrand() const;

```

```
24   //Get-function for license plate
25   std::string GetLicense() const;
26
27   //Prints the vehicle to the given ostream
28   virtual void Print(std::ostream& os) const = 0;
29
30   //creates a clone of itself on the heap
31   //return: pointer to the instance of itself on the heap
32   virtual Vehicle* Clone() const = 0;
33
34 protected:
35
36   Vehicle() = default;
37
38   Vehicle(std::string const& lic, std::string const& brand);
39
40   std::string mBrand;
41
42   std::string mLicense;
43 };
44
45 #endif
```

Listing 6: ./CarPool/Vehicles/Vehicle.cpp

```
1  ////////////////////////////////////////////////////////////////////////
2  // Workfile : Vehicle.cpp
3  // Author : Philipp Holzer / Adam Kensy
4  // Date : 15.10.2019
5  // Description : vehicle class
6  // Remarks : -
7  // Revision : 0
8  ////////////////////////////////////////////////////////////////////////
9  #include "Vehicle.h"
10
11
12 std::string Vehicle::GetBrand() const
13 {
14   return mBrand;
15 }
16
17 std::string Vehicle::GetLicense() const
18 {
19   return mLicense;
20 }
21
22 Vehicle::Vehicle(std::string const& lic, std::string const& brand) : mLicense{lic}, mBrand{
       brand}
23 {
24 }
```

Listing 7: ./CarPool/Vehicles/Car.h

```
1  ////////////////////////////////////////////////////////////////////////
2  // Workfile : Car.h
3  // Author : Philipp Holzer / Adam Kensy
4  // Date : 15.10.2019
5  // Description : vehicle class - car
6  // Remarks : -
7  // Revision : 0
8  ////////////////////////////////////////////////////////////////////////
9
10 #ifndef CAR_H
11 #define CAR_H
12 #include "Vehicle.h"
13
14 class Car : public Vehicle
15 {
16 public:
17   Car(std::string const& lic, std::string const& brand);
```

```
18
19    void Print(std::ostream& os) const override;
20
21    Vehicle* Clone() const override;
22  };
23
24  #endif
```

Listing 8: ./CarPool/Vehicles/Car.cpp

```
1  ////////////////////////////////////////////////////////////////////////
2  // Workfile : Car.cpp
3  // Author : Philipp Holzer / Adam Kensy
4  // Date : 15.10.2019
5  // Description : vehicle class - car
6  // Remarks : -
7  // Revision : 0
8  ////////////////////////////////////////////////////////////////////////
9  #include "Car.h"
10 #include "../PrintParameters.h"
11 #include <iomanip>
12
13
14 Car::Car(std::string const& lic, std::string const& brand) : Vehicle {lic, brand}
15 {
16 }
17
18 void Car::Print(std::ostream& os) const
19 {
20   if (!os.good())
21   {
22     std::cerr << "error write stream" << std::endl;
23   }
24   os << std::setw(14) << std::left << "Fahrzeugart:" << std::right << "PKW" << std::endl;
25   os << std::setw(14) << std::left << "Marke: " << std::right << mBrand << std::endl;
26   os << std::setw(14) << std::left << "Kennzeichen: "  << std::right << mLicense << std::endl;
27   mLogBook.PrintLogs(os);
28   os << std::endl;
29 }
30
31 Vehicle* Car::Clone() const
32 {
33   return new Car{ *this };
34 }
```

Listing 9: ./CarPool/Vehicles/Truck.h

```
1  ////////////////////////////////////////////////////////////////////////
2  // Workfile : Truck.h
3  // Author : Philipp Holzer / Adam Kensy
4  // Date : 15.10.2019
5  // Description : vehicle class - truck
6  // Remarks : -
7  // Revision : 0
8  ////////////////////////////////////////////////////////////////////////
9
10 #ifndef TRUCK_H
11 #define TRUCK_H
12 #include "Vehicle.h"
13
14
15 class Truck : public Vehicle
16 {
17 public:
18   Truck(std::string const& lic, std::string const& brand);
19
20   void Print(std::ostream& os) const override;
21
22   Vehicle* Clone() const override;
23 };
24
25 #endif
```

Listing 10: ./CarPool/Vehicles/Truck.cpp

```cpp
////////////////////////////////////////////////////////////////////////
// Workfile : Truck.cpp
// Author : Philipp Holzer / Adam Kensy
// Date : 15.10.2019
// Description : vehicle class - truck
// Remarks : -
// Revision : 0
////////////////////////////////////////////////////////////////////////
#include "Truck.h"
#include "../PrintParameters.h"
#include <iomanip>

Truck::Truck(std::string const& lic, std::string const& brand) : Vehicle{lic, brand}
{
}

void Truck::Print(std::ostream& os) const
{
  if (!os.good())
  {
    std::cerr << "error write stream" << std::endl;
  }
  os << std::setw(14) << std::left << "Fahrzeugart:" << std::right << "LKW" << std::endl;
  os << std::setw(14) << std::left << "Marke: " << std::right << mBrand << std::endl;
  os << std::setw(14) << std::left << "Kennzeichen: " << std::right << mLicense << std::endl;
  mLogBook.PrintLogs(os);
  os << std::endl;
}

Vehicle* Truck::Clone() const
{
  return new Truck{ *this };
}
```

Listing 11: ./CarPool/Vehicles/Motorcycle.h

```cpp
////////////////////////////////////////////////////////////////////////
// Workfile : Motorcycle.h
// Author : Philipp Holzer / Adam Kensy
// Date : 15.10.2019
// Description : vehicle class - motorcycle
// Remarks : -
// Revision : 0
////////////////////////////////////////////////////////////////////////

#ifndef MOTORCYCLE_H
#define MOTORCYCLE_H
#include "Vehicle.h"

class Motorcycle : public Vehicle
{
public:

  Motorcycle(std::string& lic, std::string& brand);

  void Print(std::ostream& os) const override;

  Vehicle* Clone() const override;
};

#endif
```

Listing 12: ./CarPool/Vehicles/Motorcycle.cpp

```cpp
////////////////////////////////////////////////////////////////////////
// Workfile : Motorcycle.cpp
// Author : Philipp Holzer / Adam Kensy
// Date : 15.10.2019
// Description : vehicle class - motorcycle
```

```cpp
// Remarks : -
// Revision : 0
//////////////////////////////////////////////////////////////////////////
#include "Motorcycle.h"
#include "../PrintParameters.h"
#include <iomanip>

Motorcycle::Motorcycle(std::string& lic, std::string& brand) : Vehicle{lic, brand}
{
}

void Motorcycle::Print(std::ostream& os) const
{
  if (!os.good())
  {
    std::cerr << "error write stream" << std::endl;
  }
  os << std::setw(14) << std::left << "Fahrzeugart:" << std::right << "Motorrad" << std::endl;
  os << std::setw(14) << std::left << "Marke: " << std::right << mBrand << std::endl;
  os << std::setw(14) << std::left << "Kennzeichen: " << std::right << mLicense << std::endl;
  mLogBook.PrintLogs(os);
  os << std::endl;
}

Vehicle* Motorcycle::Clone() const
{
  return new Motorcycle{ *this };
}
```

5.4 LogBook

Listing 13: ./CarPool/LogBook/LogBook.h

```cpp
//////////////////////////////////////////////////////////////////////////
// Workfile : LogBook.cpp
// Author : Philipp Holzer / Adam Kensy
// Date : 15.10.2019
// Description : drivers log book for a vehicle
// Remarks : -
// Revision : 0
//////////////////////////////////////////////////////////////////////////

#ifndef LOGBOOK_H
#define LOGBOOK_H

#include "../Object.h"
#include <vector>
#include <ctime>
#include <iostream>

//Class which represents a log book for vehicles
class LogBook : public Object
{
public:

  //Creates a new entry and adds it to the log book
  //param date: Struct tm from ctime
  //param distance: the driven distance in km
  void NewEntry(tm const& date, size_t const distance);

  //Removes one single entry and which contains exactly the given date and distance
  //param date: Struct tm from ctime
  //param distance: the driven distance in km
  void RemoveEntry(tm const& date, size_t const distance);

  //Prints the whole log book to the given ostream
  //param ost: ostream to write
  void PrintLogs(std::ostream& ost) const;

  //Deletes all entries
```

```
38   void Clear();
39
40   //Calculates the total distance in km
41   //return: total distance in km
42   size_t GetKMSum() const;
43
44 private:
45
46   //This class represents an entry in the log book
47   class TEntry
48   {
49   public:
50     tm mDate;
51     size_t mDistance;
52
53     bool operator==(TEntry const& entry) const;
54     bool operator<(TEntry const& entry) const;
55
56     //Prints a single entry to the given ostream
57     //param ost: ostream to print at
58     void PrintEntry(std::ostream& ost) const;
59   };
60
61   std::vector<TEntry> mEntries;
62 };
63
64 #endif
```

Listing 14: ./CarPool/LogBook/LogBook.cpp

```
1  ////////////////////////////////////////////////////////////////////////////
2  // Workfile : LogBook.cpp
3  // Author : Philipp Holzer / Adam Kensy
4  // Date : 15.10.2019
5  // Description : drivers log book for a vehicle
6  // Remarks : -
7  // Revision : 0
8  ////////////////////////////////////////////////////////////////////////////
9
10 #include "LogBook.h"
11 #include <algorithm>
12 #include <iomanip>
13 #include "../PrintParameters.h"
14
15 static const std::string cDistanceUnit = "km";
16
17 static const size_t cTmOffsetYears = 1900;
18 static const size_t cTmMonthOffset = 1;
19
20 static const std::string cErrEntryDoesNotExist = "Entry does not exist! Couldn't delete entry.
       ";
21
22 void LogBook::NewEntry(tm const& date, size_t const distance)
23 {
24   TEntry newEntry{ date, distance };
25   //find the position where to insert the new entry
26   auto it = std::find_if(mEntries.cbegin(), mEntries.cend(), [newEntry](TEntry const& e) {
         return newEntry < e; });
27   mEntries.insert(it, newEntry);
28 }
29
30 void LogBook::RemoveEntry(tm const& date, size_t const distance)
31 {
32   auto foundIt = std::find(mEntries.cbegin(), mEntries.cend(), TEntry{ date, distance });
33   if (foundIt != mEntries.cend())
34   {
35     mEntries.erase(foundIt);
36   }
37   else
38   {
39     std::cerr << cErrEntryDoesNotExist << std::endl;
40   }
```

```
41 }
42
43 void LogBook::PrintLogs(std::ostream& ost) const
44 {
45   if (!ost.good())
46   {
47     std::cerr << "error write stream" << std::endl;
48   }
49   for (auto it = mEntries.cbegin(); it != mEntries.cend(); ++it)
50   {
51     it->PrintEntry(ost);
52   }
53 }
54
55 void LogBook::Clear()
56 {
57   mEntries.clear();
58 }
59
60 size_t LogBook::GetKMSum() const
61 {
62   size_t sum = 0;
63   for (auto it = mEntries.cbegin(); it != mEntries.cend(); ++it)
64   {
65     sum += it->mDistance;
66   }
67
68   return sum;
69 }
70
71 bool LogBook::TEntry::operator==(TEntry const& entry) const
72 {
73   return mDistance == entry.mDistance && mDate.tm_year == entry.mDate.tm_year && mDate.tm_mon
       == entry.mDate.tm_mon && mDate.tm_mday == entry.mDate.tm_mday;
74 }
75
76 bool LogBook::TEntry::operator<(TEntry const& entry) const
77 {
78   if (mDate.tm_year <= entry.mDate.tm_year)
79   {
80     if (mDate.tm_year < entry.mDate.tm_year)
81     {
82       return true;
83     }
84     else
85     {
86       if (mDate.tm_mon <= entry.mDate.tm_mon)
87       {
88         if (mDate.tm_mon < entry.mDate.tm_mon)
89         {
90           return true;
91         }
92         else
93         {
94           if (mDate.tm_mday < entry.mDate.tm_mday)
95           {
96             return true;
97           }
98           else
99           {
100            if (mDistance < entry.mDistance)
101            {
102              return true;
103            }
104            else
105            {
106              return false;
107            }
108          }
109        }
110      }
111      else
112      {
```

```
113         return false;
114       }
115     }
116   }
117   else
118   {
119     return false;
120   }
121 }
122
123 void LogBook::TEntry::PrintEntry(std::ostream& ost) const
124 {
125   if (!ost.good())
126   {
127     std::cerr << "error write stream" << std::endl;
128   }
129   ost << mDate.tm_mday << "." << mDate.tm_mon + cTmMonthOffset << "."
130     << mDate.tm_year + cTmOffsetYears << ":" << std::setw(8)<< std::right
131     << mDistance << " " << cDistanceUnit << std::endl;
132 }
```

5.5 main

Listing 15: "./CarPool/main.cpp"

```
1  ////////////////////////////////////////////////////////////////////////
2  // Workfile : main.cpp
3  // Author : Philipp Holzer / Adam Kensy
4  // Date : 15.10.2019
5  // Description : testdriver for the carpool
6  // Remarks : -
7  // Revision : 0
8  ////////////////////////////////////////////////////////////////////////
9
10 #include <iostream>
11 #include <string>
12 #include "CarPool/CarPool.h"
13 #include "Vehicles/Truck.h"
14 #include "Vehicles/Car.h"
15 #include "Vehicles/Motorcycle.h"
16 #include "Vehicles/Vehicle.h"
17 #include <vld.h>
18
19 using namespace std;
20
21
22 int main()
23 {
24   string lic_mc_1 = "LL-HAGE1";
25   string brand_mc_1 = "Kawazaki";
26
27   string lic_mc_2 = "LL-BERG2";
28   string brand_mc_2 = "Yamaha";
29
30   string lic_tr_1 = "LL-HARD3";
31   string brand_tr_1 = "Mercedes";
32
33   string lic_tr_2 = "LL-WARE4";
34   string brand_tr_2 = "Koenigsegg";
35
36   string lic_c_1 = "LL-ISSO5";
37   string brand_c_1 = "Opel Corsa (nagelneu)";
38
39   string lic_c_2 = "LL-C000L";
40   string brand_c_2 = "Opel Corsa (verrostet und ohne Klima)";
41
42
43   CarPool PandA_Solutions;
44
45   Motorcycle adams_bike{lic_mc_1, brand_mc_1};
```

```
46  Motorcycle phils_bike{lic_mc_2, brand_mc_2};
47
48  Truck adams_truck{lic_tr_1, brand_tr_1};
49  Truck phils_truck{lic_tr_2, brand_tr_2};
50
51  Car adams_car{lic_c_1, brand_c_1};
52  Car phils_car{lic_c_2, brand_c_2};
53
54  tm t1;
55  t1.tm_year = 108;
56  t1.tm_mon = 0;
57  t1.tm_mday = 12;
58
59  tm t2;
60  t2.tm_year = 93;
61  t2.tm_mon = 2;
62  t2.tm_mday = 24;
63
64  tm t3;
65  t3.tm_year = 119;
66  t3.tm_mon = 6;
67  t3.tm_mday = 31;
68
69  tm t4;
70  t4.tm_year = 119;
71  t4.tm_mon = 11;
72  t4.tm_mday = 4;
73
74  tm t5;
75  t5.tm_year = 119;
76  t5.tm_mon = 8;
77  t5.tm_mday = 17;
78
79  adams_bike.mLogBook.NewEntry(t1, 17);
80  adams_bike.mLogBook.NewEntry(t2, 43);
81  adams_bike.mLogBook.NewEntry(t3, 21);
82  adams_bike.mLogBook.NewEntry(t4, 43);
83  adams_bike.mLogBook.NewEntry(t5, 89);
84  adams_bike.mLogBook.NewEntry(t5, 110);
85  adams_bike.mLogBook.NewEntry(t5, 89);
86
87  cout << "**********************************" << endl;
88  cout << "Testsection LogBook " << endl;
89  cout << "**********************************" << endl;
90
91  adams_bike.mLogBook.PrintLogs(cout);
92  cout << endl;
93
94  cout << "Entry got removed: " << endl;
95  cout << "-----------------------------------" << endl;
96
97  adams_bike.mLogBook.RemoveEntry(t5, 89);
98  adams_bike.mLogBook.PrintLogs(cout);
99  cout << endl;
100
101 cout << " Removing a nonexisting entry: " << endl;
102 cout << "-----------------------------------" << endl;
103
104 adams_bike.mLogBook.RemoveEntry(t5, 1717);
105
106 cout << endl;
107
108 cout << "Print km-sum: " << endl;
109 cout << "-----------------------------------" << endl;
110
111 cout << adams_bike.mLogBook.GetKMSum() << endl;
112 cout << endl;
113
114 cout << "Log cleared (for the next testcases it got filled again): " << endl;
115 cout << "-----------------------------------" << endl;
116 cout << endl;
117
118 adams_bike.mLogBook.NewEntry(t1, 17);
```

```
119   adams_bike.mLogBook.NewEntry(t2, 43);
120   adams_bike.mLogBook.NewEntry(t3, 21);
121   adams_bike.mLogBook.NewEntry(t4, 43);
122   adams_bike.mLogBook.NewEntry(t5, 89);
123   adams_bike.mLogBook.NewEntry(t5, 110);
124   adams_bike.mLogBook.NewEntry(t5, 89);
125
126   adams_bike.mLogBook.Clear();
127   adams_bike.mLogBook.PrintLogs(cout);
128
129   phils_bike.mLogBook.NewEntry(t1, 432);
130   phils_bike.mLogBook.NewEntry(t2, 23);
131   phils_bike.mLogBook.NewEntry(t3, 26);
132   phils_bike.mLogBook.NewEntry(t4, 117);
133   phils_bike.mLogBook.NewEntry(t5, 45);
134
135   adams_truck.mLogBook.NewEntry(t1, 67);
136   adams_truck.mLogBook.NewEntry(t2, 98);
137   adams_truck.mLogBook.NewEntry(t3, 45);
138   adams_truck.mLogBook.NewEntry(t4, 34);
139   adams_truck.mLogBook.NewEntry(t5, 21);
140
141   phils_truck.mLogBook.NewEntry(t1, 4567);
142   phils_truck.mLogBook.NewEntry(t2, 82456);
143   phils_truck.mLogBook.NewEntry(t3, 4332);
144   phils_truck.mLogBook.NewEntry(t4, 6789);
145   phils_truck.mLogBook.NewEntry(t5, 4321);
146
147   adams_car.mLogBook.NewEntry(t1, 231);
148   adams_car.mLogBook.NewEntry(t2, 927);
149   adams_car.mLogBook.NewEntry(t3, 7028);
150   adams_car.mLogBook.NewEntry(t4, 211);
151   adams_car.mLogBook.NewEntry(t5, 6837);
152
153   phils_car.mLogBook.NewEntry(t1, 265);
154   phils_car.mLogBook.NewEntry(t2, 1093);
155   phils_car.mLogBook.NewEntry(t3, 483);
156   phils_car.mLogBook.NewEntry(t4, 46);
157   phils_car.mLogBook.NewEntry(t5, 7392);
158
159   PandA_Solutions.AddVehicle(adams_bike.Clone());
160   PandA_Solutions.AddVehicle(adams_truck.Clone());
161   PandA_Solutions.AddVehicle(adams_car.Clone());
162
163   PandA_Solutions.AddVehicle(phils_bike.Clone());
164   PandA_Solutions.AddVehicle(phils_truck.Clone());
165   PandA_Solutions.AddVehicle(phils_car.Clone());
166
167   cout << "**********************************" << endl;
168   cout << "Testsection CarPool " << endl;
169   cout << "**********************************" << endl;
170
171   cout << "All added vehicles: " << endl;
172   cout << "-----------------------------------" << endl;
173   PandA_Solutions.PrintVehicles(cout);
174
175   PandA_Solutions.RemoveVehicle(adams_car.GetLicense());
176
177   cout << "Adams Car (LL-ISSO5) got removed: " << endl;
178   cout << "-----------------------------------" << endl;
179
180   PandA_Solutions.PrintVehicles(cout);
181
182   cout << "Print LKW with license 'LL-HARD3' (SearchByLicense): " << endl;
183   cout << "-----------------------------------" << endl;
184
185   TVehiclePointerItor it;
186   if (PandA_Solutions.SearchByLicense("LL-HARD3", it))
187   {
188     (*it)->Print(cout);
189   }
190   else
191   {
```

```
192     cerr << "LKW with the license 'LL-HARD3' does not exist";
193   }
194
195   cout << "Number of vehicles in the car pool: " << endl;
196   cout << PandA_Solutions.GetVehicleAmount() << endl << endl;
197
198   cout << "Print vehicles of copied object (Copy CTOR): " << endl;
199   cout << "-----------------------------------" << endl;
200
201   CarPool carpool1{ PandA_Solutions };
202   carpool1.PrintVehicles(cout);
203
204   cout << "Remove vehicle with license 'LL-HARD3' and assign PandA_Solutions to carpool1 (
          assignment operator): " << endl;
205   cout << "-----------------------------------" << endl;
206   PandA_Solutions.RemoveVehicle("LL-HARD3");
207   carpool1 = PandA_Solutions;
208   carpool1.PrintVehicles(cout);
209
210   //test self assignment
211   carpool1 = carpool1;
212
213   return 0;
214 }
```