1 Organisatorisches

1.1 Team

- Reinhard Penn, s1110306019
- Bernhard Selymes, s1110306024

1.2 Aufteilung

- Reinhard Penn
  - Planung
  - Klassendiagramm
  - Implementierung der Klassen CarRental, ConcreteCar und Unterklassen
  - Testen aller Klassen
- Bernhard Selymes
  - Planung
  - Klassendiagramm
  - Implementierung der Klassen ICar, Decorator und Unterklassen
  - Dokumentation

1.3 Zeitaufwand

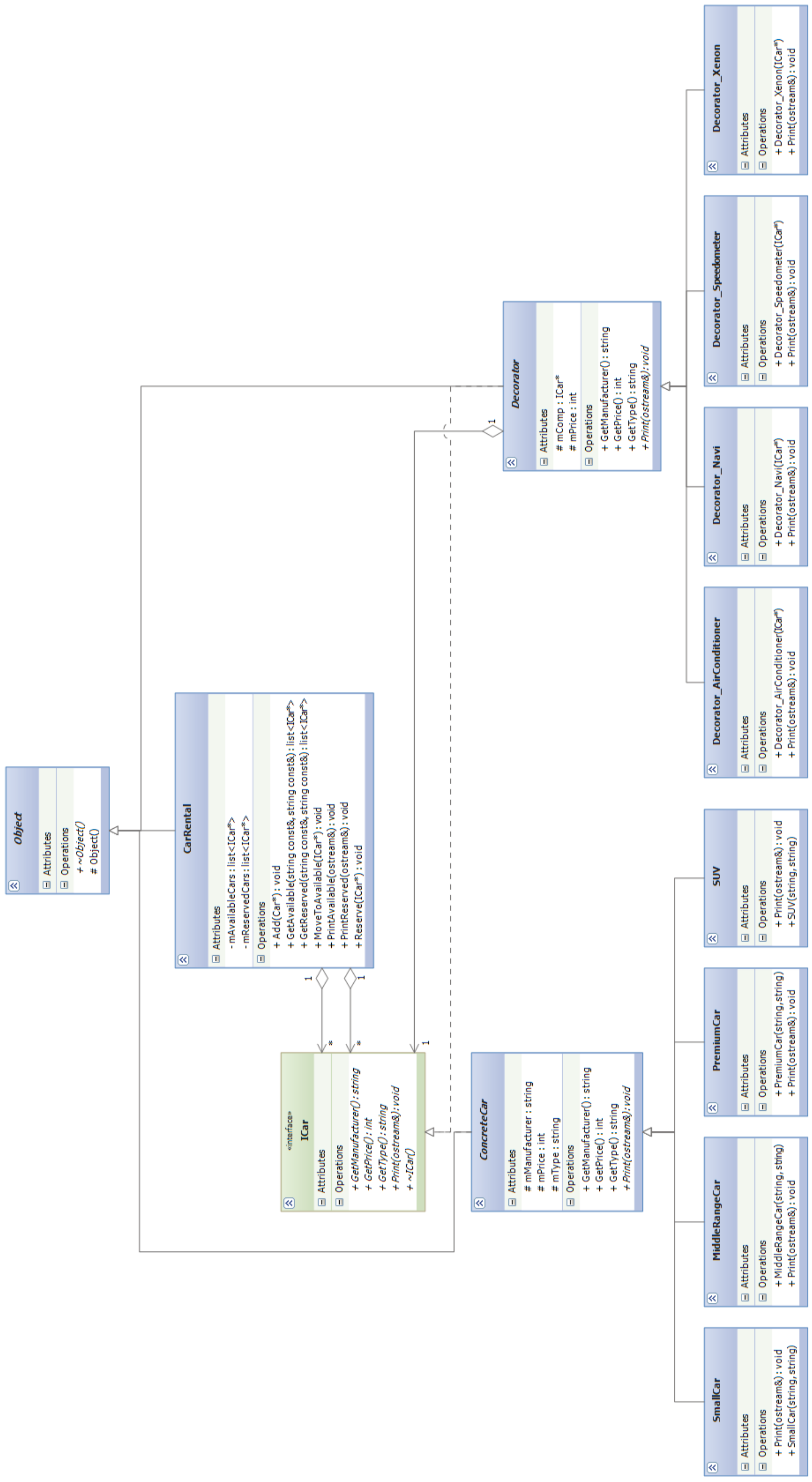
- geschätzte Mh: 12
- tatsächlich: Reinhard (8h), Bernhard (7h)

2 Systemspezifikation

Eine Sofware für die Verwaltung von Kraftfahrzeuge in einer Autovermietung soll entworfen werden. Die Kraftfahrzeuge gehören zu einer Klasse, die den Preis des Fahrzeugs bestimmt. Ein Kraftfahrzeug kann zusätzlich Sonderausstattungen haben, die zusätzlich etwas kosten.

3 Systementwurf

3.1 Klassendiagramm



3.2 Komponentenübersicht

- Klasse ”Object”:
  Basis aller Basisklassen.
- Interface ”ICar”:
  Schnittstellen der Funktionen.
- Klasse ”ConcreteCar”:
  Basisklasse für die einzelnen konkreten Kraftfahrzeuge.
- Klassen ”SmallCar, MiddleRangeCar, PremiumCar und SUV”:
  Konkrete Klassen von Kraftfahrzeugen.
- Klasse ”Decorator”:
  Basisklasse für die konkreten Sonderausstattungen.
- Klassen ”AirConditioner, Navi, Speedometer und Xenon”:
  Konkrete Sonderausstattungen.
- Klasse ”CarRental”:
  Verwaltet die Kraftfahrzeuge.

4 Komponentenentwurf

4.1 Klasse ”Object”

Abstrakte Basisklasse aller Klassen. Von ihr werden alle anderen Klassen abgeleitet. Beinhaltet einen virtuellen Destruktor.

4.2 Interface ”ICar”

Definiert die Schnittstellen der Methoden. Hat einen virtuellen Destruktor.

4.3 Klasse ”ConcreteCar”

Basisklasse für die konkreten Kraftfahrzeuge. Hat protected Member die den Hersteller, den Preis und den Typ speichern. Hat drei Get-Methoden für diese member. Hat eine abstrakte Methode ”Print” die in den Unterklassen implementiert wird.

4.4 Klassen ”SmallCar, MiddleRangeCar, PremiumCar und SUV”

Konkrete Klassen von Kraftfahrzeugen.

Methode ”Print”:
Schnittstelle:
Parameter: ostream&
Rückgabetyp: void
Wird je nach Klassen entsprechend implementiert. Gibt aus um welche Klasse es sich handelt und danach die entsprechenden Daten des Fahrzeugs.

4.5 Klasse ”Decorator”

Hat einen Member der den Preis speichert und einen der einen Pointer auf das Objekt, das er dekoriert, speichert. Die Funktion ”Print” ist abstrakt. Hat drei Get-Methoden, die bis ganz in die Tiefe gehen (bis zum Kraftfahrzeug) und dann den Wert von dort zurückliefern. Beim Preis werden die Werte aufaddiert.

4.6 Klassen ”AirConditioner, Navi, Speedometer und Xenon”

Konkrete Ausstattungen.

Konstruktoren:
Schnittstelle:
Parameter: ICar*
Überprüft den übergebenen Parameter auf Gültigkeit und weist den konkreten Preis zu.

Methode ”Print”:
Schnittstelle:
Parameter: ostream&

Rückgabetyp: void
Ruft die Printfunktion des Objektes, das es dekoriert, auf und gibt dann aus um welche Sonderausstattung es sich handelt und den Preis davon.

4.7 Klasse ”CarRental”

Enthält eine Liste mit verfügbaren Kraftfahrzeugen und eine mit reservierten. Hat Get-Methoden für diese. Hat Methoden zum hinzufügen und verschieben zwischen den zwei Listen.

Methoden ”PrintAvailable” und ”PrintReserved”:
Schnittstelle:
Parameter: ostream&
Rückgabetyp: void
Gibt die Daten (Hersteller, Typ, Preis vom Fahrzeug, Sonderausstattungen und Preis davon und Gesamtpreis (Fahrzeug mit Ausstattungen)) aus.

Methoden ”GetAvailable” und ”GetReserved”:
Schnittstelle:
Parameter: string const&, string const &
Rückgabetyp: list mit ICar*
Geben eine Liste zurück in der die Kraftfahrzeuge enthalten sind die der angegebenen Herstellermarke und Typ des Fahrzeugs entsprechen.

5 Source Code

```cpp
///////////////////////////////////////////////////////////////////////////
// Workfile : Object.h
// Author : Reinhard Penn, Bernhard Selymes
// Date : 6.11.2012
// Description : Header for Object.cpp
///////////////////////////////////////////////////////////////////////////

#ifndef OBJECT_H
#define OBJECT_H

class Object
{
public:
   //virtual Destructor for baseclass
   virtual ~Object();
protected:
   //Default CTor for baseclass
   Object();
};

#endif
```

```cpp
///////////////////////////////////////////////////////////////////////////
// Workfile : Object.cpp
// Author : Reinhard Penn, Bernhard Selymes
// Date : 6.11.2012
// Description : Baseclass with protected constructor
///////////////////////////////////////////////////////////////////////////

#include "Object.h"

Object::Object()
{}

Object::~Object()
{}
```

```
///////////////////////////////////////////////////////////////////////////
// Workfile : ICar.h
// Author : Reinhard Penn, Bernhard Selymes
// Date : 6.11.2012
// Description : Interface
///////////////////////////////////////////////////////////////////////////

#ifndef ICAR_H
#define ICAR_H

class ICar
{
public:
   //virtual DTor
   virtual ~ICar() {};

   virtual int GetPrice() const = 0;
   virtual void Print(std::ostream& ost) = 0;
   virtual std::string GetManufacturer() const = 0;
   virtual std::string GetType() const = 0;
};

#endif
```

```cpp
///////////////////////////////////////////////////////////////////////////
// Workfile : ConcreteCar.h
// Author : Reinhard Penn, Bernhard Selymes
// Date : 6.11.2012
// Description : Header for ConcreteCar.cpp
///////////////////////////////////////////////////////////////////////////

#ifndef CONCRETECAR_H
#define CONCRETECAR_H

#include <fstream>
#include <string>
#include "Object.h"
#include "ICar.h"

class ConcreteCar :
   public Object,
   public ICar
{
public:
   std::string GetManufacturer() const;
   int GetPrice() const;
   std::string GetType() const;
   virtual void Print(std::ostream& stream) = 0;

protected:
   std::string mManufacturer;
   int mPrice;
   std::string mType;
};

#endif
```

```cpp
///////////////////////////////////////////////////////////////////////////
// Workfile : ConcreteCar.cpp
// Author : Reinhard Penn, Bernhard Selymes
// Date : 6.11.2012
// Description : Implementation of class ConcreteCar
///////////////////////////////////////////////////////////////////////////

#include "ConcreteCar.h"

std::string ConcreteCar::GetManufacturer() const
{
   return mManufacturer;
}

int ConcreteCar::GetPrice() const
{
   return mPrice;
}

std::string ConcreteCar::GetType() const
{
   return mType;
}
```

```cpp
///////////////////////////////////////////////////////////////////////////
// Workfile : SmallCar.h
// Author : Reinhard Penn, Bernhard Selymes
// Date : 6.11.2012
// Description : Header for SmallCar.cpp
///////////////////////////////////////////////////////////////////////////

#ifndef SMALLCAR_H
#define SMALLCAR_H

#include <string>
#include "ConcreteCar.h"

std::size_t const priceSmallCar = 7500;

class SmallCar :
   public ConcreteCar
{
public:
   SmallCar(std::string manufacturer, std::string type);
   void Print(std::ostream& stream);
};

#endif
```

```cpp
///////////////////////////////////////////////////////////////////////////
// Workfile : SmallCar.cpp
// Author : Reinhard Penn, Bernhard Selymes
// Date : 6.11.2012
// Description : Implementation of class SmallCar
///////////////////////////////////////////////////////////////////////////

#include <iostream>
#include "SmallCar.h"

SmallCar::SmallCar(std::string manufacturer, std::string type)
{
   try
   {
      if(manufacturer == "")
      {
         std::string error = "no valid manufacturer";
         throw (error);
      }
      if(type == "")
      {
         std::string error = "no valid type";
         throw (error);
      }
      mManufacturer = manufacturer;
      mPrice = priceSmallCar;
      mType = type;
   }
   catch (std::string const& error)
   {
      std::cout << "Error in SmallCar::SmallCar: " << error << std::endl;
   }
   catch(...)
   {
      std::cerr << "SmallCar::SmallCar: Unknown Exception occured" << std::
          endl;
   }
}

void SmallCar::Print(std::ostream& stream)
{
   try
   {
      if(stream == 0)
      {
         std::string error = "no valid stream";
         throw (error);
      }
      stream << "Small Car: " << mManufacturer << " " << mType
            << " - Price: " << mPrice << std::endl;
   }
   catch (std::string const& error)
   {
      std::cout << "Error in SmallCar::Print: " << error << std::endl;
   }
   catch(...)
   {
      std::cerr << "SmallCar::Print: Unknown Exception occured" << std::
          endl;
   }
}
```

```cpp
///////////////////////////////////////////////////////////////////////////
// Workfile : MiddleRangeCar.h
// Author : Reinhard Penn, Bernhard Selymes
// Date : 6.11.2012
// Description : Header for MiddleRange.cpp
///////////////////////////////////////////////////////////////////////////

#ifndef MIDDLERANGECAR_H
#define MIDDLERANGECAR_H

#include <string>
#include "ConcreteCar.h"

std::size_t const priceMiddleRangeCar = 16000;

class MiddleRangeCar :
   public ConcreteCar
{
public:
   MiddleRangeCar(std::string manufacturer, std::string type);
   void Print(std::ostream& stream);
};

#endif
```

```cpp
///////////////////////////////////////////////////////////////////////////
// Workfile : MiddleRangeCar.cpp
// Author : Reinhard Penn, Bernhard Selymes
// Date : 6.11.2012
// Description : Implementation of class MiddleRangeCar
///////////////////////////////////////////////////////////////////////////

#include <iostream>
#include "MiddleRangeCar.h"

MiddleRangeCar::MiddleRangeCar(std::string manufacturer, std::string type)
{
   try
   {
      if(manufacturer == "")
      {
         std::string error = "no valid manufacturer";
         throw (error);
      }
      if(type == "")
      {
         std::string error = "no valid type";
         throw (error);
      }
      mManufacturer = manufacturer;
      mPrice = priceMiddleRangeCar;
      mType = type;
   }
   catch (std::string const& error)
   {
      std::cout << "Error in MiddleRangeCar::MiddleRangeCar: " << error <<
          std::endl;
   }
   catch(...)
   {
      std::cerr << "MiddleRangeCar::MiddleRangeCar: Unknown Exception
          occured" << std::endl;
   }
}

void MiddleRangeCar::Print(std::ostream& stream)
{
   try
   {
      if(stream == 0)
      {
         std::string error = "no valid stream";
         throw (error);
      }
      stream << "Middlerange Car: " << mManufacturer << " " << mType
            << " - Price: " << mPrice << std::endl;
   }
   catch (std::string const& error)
   {
      std::cout << "Error in MiddleRangeCar::Print: " << error << std::endl
          ;
   }
   catch(...)
   {
      std::cerr << "MiddleRangeCar::Print: Unknown Exception occured" <<
```

```
              std::endl;
    }
}
```

```cpp
///////////////////////////////////////////////////////////////////////////
// Workfile : PremiumCar.h
// Author : Reinhard Penn, Bernhard Selymes
// Date : 6.11.2012
// Description : Header for PremiumCar.cpp
///////////////////////////////////////////////////////////////////////////

#ifndef PREMIUMCAR_H
#define PREMIUMCAR_H

#include <string>
#include "ConcreteCar.h"

std::size_t const pricePremiumCar = 45000;

class PremiumCar :
   public ConcreteCar
{
public:
   PremiumCar(std::string manufacturer, std::string type);
   void Print(std::ostream& stream);
};

#endif
```

```cpp
///////////////////////////////////////////////////////////////////////////
// Workfile : PremiumCar.cpp
// Author : Reinhard Penn, Bernhard Selymes
// Date : 6.11.2012
// Description : Implementation of class PremiumCar
///////////////////////////////////////////////////////////////////////////

#include <iostream>
#include "PremiumCar.h"

PremiumCar::PremiumCar(std::string manufacturer, std::string type)
{
   try
   {
      if(manufacturer == "")
      {
         std::string error = "no valid manufacturer";
         throw (error);
      }
      if(type == "")
      {
         std::string error = "no valid type";
         throw (error);
      }
      mManufacturer = manufacturer;
      mPrice = pricePremiumCar;
      mType = type;
   }
   catch (std::string const& error)
   {
      std::cout << "Error in PremiumCar::PremiumCar: " << error << std::
          endl;
   }
   catch(...)
   {
      std::cerr << "PremiumCar::PremiumCar: Unknown Exception occured" <<
          std::endl;
   }
}

void PremiumCar::Print(std::ostream& stream)
{
   try
   {
      if(stream == 0)
      {
         std::string error = "no valid stream";
         throw (error);
      }
      stream << "Premium Car: " << mManufacturer << " " << mType
            << " - Price: " << mPrice << std::endl;
   }
   catch (std::string const& error)
   {
      std::cout << "Error in PremiumCar::Print: " << error << std::endl;
   }
   catch(...)
   {
      std::cerr << "PremiumCar::Print: Unknown Exception occured" << std::
          endl;
```

```
        }
}
```

```
///////////////////////////////////////////////////////////////////////////
// Workfile : SUV.h
// Author : Reinhard Penn, Bernhard Selymes
// Date : 6.11.2012
// Description : Header for SUV.cpp
///////////////////////////////////////////////////////////////////////////

#ifndef SUV_H
#define SUV_H

#include <string>
#include "ConcreteCar.h"

std::size_t const priceSUV = 22000;

class SUV :
   public ConcreteCar
{
public:
   SUV(std::string manufacturer, std::string type);
   void Print(std::ostream& stream);
};

#endif
```

```
///////////////////////////////////////////////////////////////////////////
// Workfile : SUV.cpp
// Author : Reinhard Penn, Bernhard Selymes
// Date : 6.11.2012
// Description : Implementation of class SUV
///////////////////////////////////////////////////////////////////////////

#include <iostream>
#include "SUV.h"

SUV::SUV(std::string manufacturer, std::string type)
{
   try
   {
      if(manufacturer == "")
      {
         std::string error = "no valid manufacturer";
         throw (error);
      }
      if(type == "")
      {
         std::string error = "no valid type";
         throw (error);
      }
      mManufacturer = manufacturer;
      mPrice = priceSUV;
      mType = type;
   }
   catch (std::string const& error)
   {
      std::cout << "Error in SUV::SUV: " << error << std::endl;
   }
   catch(...)
   {
      std::cerr << "SUV::SUV: Unknown Exception occured" << std::endl;
   }
}

void SUV::Print(std::ostream& stream)
{
   try
   {
      if(stream == 0)
      {
         std::string error = "no valid stream";
         throw (error);
      }
      stream << "SUV: " << mManufacturer << " " << mType
            << " - Price: " << mPrice << std::endl;
   }
   catch (std::string const& error)
   {
      std::cout << "Error in SUV::Print: " << error << std::endl;
   }
   catch(...)
   {
      std::cerr << "SUV::Print: Unknown Exception occured" << std::endl;
   }
}
```

```cpp

///////////////////////////////////////////////////////////////////////////
// Workfile : Decorator.h
// Author : Reinhard Penn, Bernhard Selymes
// Date : 6.11.2012
// Description : Header for Decorator.cpp
///////////////////////////////////////////////////////////////////////////

#ifndef DECORATOR_H
#define DECORATOR_H

#include <string>
#include <fstream>
#include "Object.h"
#include "ICar.h"

class Decorator :
   public Object,
   public ICar
{
public:
   virtual ~Decorator();
   std::string GetManufacturer() const;
   int GetPrice() const;
   std::string GetType() const;
   void Print(std::ostream& stream) = 0;

protected:
   ICar* mComp;
   int mPrice;
};

#endif
```

```cpp

///////////////////////////////////////////////////////////////////////////
// Workfile : Decorator.cpp
// Author : Reinhard Penn, Bernhard Selymes
// Date : 6.11.2012
// Description : Implementation of class Decorator
///////////////////////////////////////////////////////////////////////////

#include <iostream>
#include "Decorator.h"

Decorator::~Decorator()
{
   delete mComp;
}

std::string Decorator::GetManufacturer() const
{
   return mComp->GetManufacturer();
}

//returns the price of the whole car (incl. all decorators)
int Decorator::GetPrice() const
{
   return mPrice + mComp->GetPrice();
}

std::string Decorator::GetType() const
{
   return mComp->GetType();
}
```

```cpp
///////////////////////////////////////////////////////////////////////////
// Workfile : Decorator_AirConditioner.h
// Author : Reinhard Penn, Bernhard Selymes
// Date : 6.11.2012
// Description : Header for Decorator_AirConditioner.cpp
///////////////////////////////////////////////////////////////////////////

#ifndef DECORATOR_AIRCONDITIONER_H
#define DECORATOR_AIRCONDITIONER_H

#include "Decorator.h"

int const airConditionerPrice = 1500;

class Decorator_AirConditioner :
   public Decorator
{
public:
   Decorator_AirConditioner(ICar* car);
   void Print(std::ostream& stream);
};

#endif
```

```cpp
///////////////////////////////////////////////////////////////////////////
// Workfile : Decorator_AirConditioner.cpp
// Author : Reinhard Penn, Bernhard Selymes
// Date : 6.11.2012
// Description : Implementation of class Decorator_AirConditioner
///////////////////////////////////////////////////////////////////////////

#include <iostream>
#include "Decorator_AirConditioner.h"

Decorator_AirConditioner::Decorator_AirConditioner(ICar* car)
{
   try
   {
      if(car == 0)
      {
         std::string error = "no valid pointer";
         throw (error);
      }
      mComp = car;
      mPrice = airConditionerPrice;
   }
   catch (std::string const& error)
   {
      std::cout << "Error in Decorator_AirConditioner::
          Decorator_AirConditioner: " << error << std::endl;
   }
   catch(...)
   {
      std::cerr << "Decorator_AirConditioner::Decorator_AirConditioner:
          Unknown Exception occured" << std::endl;
   }
}

void Decorator_AirConditioner::Print(std::ostream& stream)
{
   try
   {
      if(stream == 0)
      {
         std::string error = "no valid stream";
         throw (error);
      }
      mComp->Print(stream);
      stream << "Air Conditioner" << " - Price: " << mPrice << std::endl;
   }
   catch (std::string const& error)
   {
      std::cout << "Error in Decorator_AirConditioner::Print: " << error <<
           std::endl;
   }
   catch(...)
   {
      std::cerr << "Decorator_AirConditioner::Print: Unknown Exception
          occured" << std::endl;
   }
}
```

```cpp
///////////////////////////////////////////////////////////////////////////
// Workfile : Decorator_Navi.h
// Author : Reinhard Penn, Bernhard Selymes
// Date : 6.11.2012
// Description : Header for Decorator_Navi.cpp
///////////////////////////////////////////////////////////////////////////

#ifndef DECORATOR_NAVI_H
#define DECORATOR_NAVI_H

#include "Decorator.h"

int const naviPrice = 2000;

class Decorator_Navi :
   public Decorator
{
public:
   Decorator_Navi(ICar* car);
   void Print(std::ostream& stream);
};

#endif
```

```cpp
///////////////////////////////////////////////////////////////////////////
// Workfile : Decorator_Navi.cpp
// Author : Reinhard Penn, Bernhard Selymes
// Date : 6.11.2012
// Description : Implementation of class Decorator_Navi
///////////////////////////////////////////////////////////////////////////

#include "Decorator_Navi.h"
#include <iostream>

Decorator_Navi::Decorator_Navi(ICar* car)
{
   try
   {
      if(car == 0)
      {
         std::string error = "no valid pointer";
         throw (error);
      }
      mComp = car;
      mPrice = naviPrice;
   }
   catch (std::string const& error)
   {
      std::cout << "Error in Decorator_Navi::Decorator_Navi: " << error <<
          std::endl;
   }
   catch(...)
   {
      std::cerr << "Decorator_Navi::Decorator_Navi: Unknown Exception
          occured" << std::endl;
   }
}

void Decorator_Navi::Print(std::ostream& stream)
{
   try
   {
      if(stream == 0)
      {
         std::string error = "no valid stream";
         throw (error);
      }
      mComp->Print(stream);
      stream << "Navi" << " - Price: " << mPrice << std::endl;
   }
   catch (std::string const& error)
   {
      std::cout << "Error in Decorator_Navi::Print: " << error << std::endl
          ;
   }
   catch(...)
   {
      std::cerr << "Decorator_Navi::Print: Unknown Exception occured" <<
          std::endl;
   }
}
```

```cpp
//////////////////////////////////////////////////////////////////////////
// Workfile : Decorator_Speedometer.h
// Author : Reinhard Penn, Bernhard Selymes
// Date : 6.11.2012
// Description : Header for Decorator_Speedometer.cpp
//////////////////////////////////////////////////////////////////////////

#ifndef DECORATOR_SPEEDOMETER_H
#define DECORATOR_SPEEDOMETER_H

#include "Decorator.h"

int const speedometerPrice = 2500;

class Decorator_Speedometer :
   public Decorator
{
public:
   Decorator_Speedometer(ICar* car);
   void Print(std::ostream& stream);
};

#endif
```

```cpp
///////////////////////////////////////////////////////////////////////////
// Workfile : Decorator_Speedometer.cpp
// Author : Reinhard Penn, Bernhard Selymes
// Date : 6.11.2012
// Description : Implementation of class Decorator_Speedometer
///////////////////////////////////////////////////////////////////////////

#include <iostream>
#include "Decorator_Speedometer.h"

Decorator_Speedometer::Decorator_Speedometer(ICar* car)
{
   try
   {
      if(car == 0)
      {
         std::string error = "no valid pointer";
         throw (error);
      }
      mComp = car;
      mPrice = speedometerPrice;
   }
   catch (std::string const& error)
   {
      std::cout << "Error in Decorator_Speedometer::Decorator_Speedometer: " << error << std::endl;
   }
   catch(...)
   {
      std::cerr << "Decorator_Speedometer::Decorator_Speedometer: Unknown Exception occured" << std::endl;
   }
}

void Decorator_Speedometer::Print(std::ostream& stream)
{
   try
   {
      if(stream == 0)
      {
         std::string error = "no valid stream";
         throw (error);
      }
      mComp->Print(stream);
      stream << "Speedometer" << " - Price: " << mPrice << std::endl;
   }
   catch (std::string const& error)
   {
      std::cout << "Error in Decorator_Speedometer::Print: " << error << std::endl;
   }
   catch(...)
   {
      std::cerr << "Decorator_Speedometer::Print: Unknown Exception occured " << std::endl;
   }
}
```

```cpp
///////////////////////////////////////////////////////////////////////////
// Workfile : Decorator_Xenion.h
// Author : Reinhard Penn, Bernhard Selymes
// Date : 6.11.2012
// Description : Header for Decorator_Xenion.cpp
///////////////////////////////////////////////////////////////////////////

#ifndef DECORATOR_XENION_H
#define DECORATOR_XENION_H

#include "Decorator.h"

int const xenionPrice = 3000;

class Decorator_Xenion :
   public Decorator
{
public:
   Decorator_Xenion(ICar* car);
   void Print(std::ostream& stream);
};

#endif
```

```cpp
///////////////////////////////////////////////////////////////////////////
// Workfile : Decorator_Xenion.cpp
// Author : Reinhard Penn, Bernhard Selymes
// Date : 6.11.2012
// Description : Implementation of class Decorator_Xenion
///////////////////////////////////////////////////////////////////////////

#include <iostream>
#include "Decorator_Xenion.h"

Decorator_Xenion::Decorator_Xenion(ICar* car)
{
   try
   {
      if(car == 0)
      {
         std::string error = "no valid pointer";
         throw (error);
      }
      mComp = car;
      mPrice = xenionPrice;
   }
   catch (std::string const& error)
   {
      std::cout << "Error in Decorator_Xenion::Decorator_Xenion: " << error << std::endl;
   }
   catch(...)
   {
      std::cerr << "Decorator_Xenion::Decorator_Xenion: Unknown Exception occured" << std::endl;
   }
}

void Decorator_Xenion::Print(std::ostream& stream)
{
   try
   {
      if(stream == 0)
      {
         std::string error = "no valid stream";
         throw (error);
      }
      mComp->Print(stream);
      stream << "Xenion" << " - Price: " << mPrice << std::endl;
   }
   catch (std::string const& error)
   {
      std::cout << "Error in Decorator_Xenion::Print: " << error << std::endl;
   }
   catch(...)
   {
      std::cerr << "Decorator_Xenion::Print: Unknown Exception occured" << std::endl;
   }
}
```

```cpp
///////////////////////////////////////////////////////////////////////////
// Workfile : CarRental.h
// Author : Reinhard Penn, Bernhard Selymes
// Date : 18.12.2012
// Description : Header for CarRental.cpp
///////////////////////////////////////////////////////////////////////////

#ifndef CARRENTAL_H
#define CARRENTAL_H

#include <list>
#include "ICar.h"

typedef std::list<ICar*> TCarList;
typedef TCarList::iterator TCarListItor;

class CarRental
{
public:
   //Destructor
   virtual ~CarRental();

   void Add(ICar* c);
   void PrintAvailable(std::ostream& ost) const;
   void PrintReserved(std::ostream& ost) const;
   TCarList GetAvailable(std::string const& type="", std::string const&
      manufacturer="") const;
   TCarList GetReserved(std::string const& type="", std::string const&
      manufacturer="") const;
   void Reserve(ICar* c);
   void MoveToAvailable(ICar* c);

private:
   TCarList mAvailableCars;
   TCarList mReservedCars;
};

#endif
```

```cpp
///////////////////////////////////////////////////////////////////////////
// Workfile : CarRental.cpp
// Author : Reinhard Penn, Bernhard Selymes
// Date : 18.12.2012
// Description : Implementation of class CarRental
///////////////////////////////////////////////////////////////////////////

#include <algorithm>
#include <iostream>
#include <string>
#include "CarRental.h"


CarRental::~CarRental()
{
   std::for_each(mAvailableCars.begin(),mAvailableCars.end(),[&](ICar* m)
   {
      delete m;
   });
   std::for_each(mReservedCars.begin(),mReservedCars.end(),[&](ICar* m)
   {
      delete m;
   });
}

void CarRental::Add(ICar* c)
{
   try
   {
      if(c == 0)
      {
         std::string error = "no valid pointer";
         throw (error);
      }
      mAvailableCars.push_back(c);
   }
   catch (std::string const& error)
   {
      std::cout << "Error in CarRental::Add: " << error << std::endl;
   }
   catch(...)
   {
      std::cerr << "CarRental::Add: Unknown Exception occured" << std::endl
         ;
   }
}

void CarRental::PrintAvailable(std::ostream& ost) const
{
   try
   {
      if(ost == 0)
      {
         std::string error = "no valid stream";
         throw (error);
      }

      std::for_each(mAvailableCars.begin(),mAvailableCars.end(),[&](ICar* m
         )
      {
         m->Print(ost);
         ost << "Total price: " << m->GetPrice() << std::endl;
      });
   }
   catch (std::string const& error)
   {
      std::cout << "Error in CarRental::PrintAvailable: " << error << std::endl;
   }
   catch(...)
   {
      std::cerr << "CarRental::PrintAvailable: Unknown Exception occured" << std::endl;
   }
}

void CarRental::PrintReserved(std::ostream& ost) const
{
   try
   {
      if(ost == 0)
      {
         std::string error = "no valid stream";
         throw (error);
      }

      std::for_each(mReservedCars.begin(),mReservedCars.end(),[&](ICar* m)
      {
         m->Print(ost);
         ost << "Total price: " << m->GetPrice() << std::endl;
      });
   }
   catch (std::string const& error)
   {
      std::cout << "Error in CarRental::PrintReserved: " << error << std::endl;
   }
   catch(...)
   {
      std::cerr << "CarRental::PrintReserved: Unknown Exception occured" << std::endl;
   }
}

TCarList CarRental::GetAvailable(std::string const& type, std::string const& manufacturer) const
{
   TCarList carList;

   std::for_each(mAvailableCars.begin(),mAvailableCars.end(),[&](ICar* m)
   {
      if(m->GetManufacturer() == manufacturer && m->GetType() == type)
      {
         carList.push_back(m);
      }
   });

   return carList;
}
```

```cpp
TCarList CarRental::GetReserved(std::string const& type, std::string const&
     manufacturer) const
{
   TCarList carList;

   std::for_each(mReservedCars.begin(),mReservedCars.end(),[&](ICar* m)
   {
      if(m->GetManufacturer() == manufacturer && m->GetType() == type)
      {
         carList.push_back(m);
      }
   });

   return carList;
}

void CarRental::Reserve(ICar* c)
{
   try
   {
      if(c == 0)
      {
         std::string error = "no valid pointer";
         throw (error);
      }
      TCarListItor itor = std::find(mAvailableCars.begin(),mAvailableCars.
          end(),c);

      if(itor == mAvailableCars.end())
      {
         std::string error = "car not found";
         throw (error);
      }

      mReservedCars.push_back(*itor);
      mAvailableCars.remove(*itor);
   }
   catch (std::string const& error)
   {
      std::cout << "Error in CarRental::Reserve: " << error << std::endl;
   }
   catch(...)
   {
      std::cerr << "CarRental::Reserve: Unknown Exception occured" << std::
          endl;
   }
}

void CarRental::MoveToAvailable(ICar* c)
{
   try
   {
      if(c == 0)
      {
         std::string error = "no valid pointer";
         throw (error);
      }
      TCarListItor itor = std::find(mReservedCars.begin(),mReservedCars.end
          (),c);

```

```
        if(itor == mReservedCars.end())
        {
            std::string error = "car not found";
            throw (error);
        }

        mAvailableCars.push_back(*itor);
        mReservedCars.remove(*itor);
    }
    catch (std::string const& error)
    {
        std::cout << "Error in CarRental::MoveToAvailable: " << error << std
            ::endl;
    }
    catch(...)
    {
        std::cerr << "CarRental::MoveToAvailable: Unknown Exception occured"
            << std::endl;
    }
}
```

```
///////////////////////////////////////////////////////////////////////////
// Workfile : Main.cpp
// Author : Reinhard Penn, Bernhard Selymes
// Date : 02.01.2013
// Description : Testdriver for CarRental
///////////////////////////////////////////////////////////////////////////

#include <iostream>
#include <algorithm>
#include <vld.h>
#include "ICar.h"
#include "CarRental.h"
#include "Decorator.h"
#include "Decorator_AirConditioner.h"
#include "Decorator_Navi.h"
#include "Decorator_Speedometer.h"
#include "Decorator_Xenion.h"
#include "SmallCar.h"
#include "MiddleRangeCar.h"
#include "PremiumCar.h"
#include "SUV.h"

using  namespace std;


void EmptyTestCase()
{
   cout << "Empty testcase with NULL pointer." << endl;

   CarRental Rental;

   Rental.Add(0);
   Rental.GetAvailable("","");
   Rental.GetReserved("","");
   Rental.MoveToAvailable(0);
   Rental.Reserve(0);
   Rental.PrintAvailable(cout);
   Rental.PrintReserved(cout);

   cout << endl << endl;
}

void SingleTestCase()
{
   cout << "Testcase with single entry" << endl;

   CarRental Rental;

   ICar* VW = new SmallCar("VW","Golf");
   ICar* MyCar = new Decorator_AirConditioner(VW);

   cout << "Add ...";
   Rental.Add(MyCar);
   cout << "done" << endl;

   cout << "GetAvailable ...";
   TCarList list = Rental.GetAvailable("VW","Golf");
   cout << "done" << endl;

   cout << "Reserve ...";
   Rental.Reserve(MyCar);
   cout << "done" << endl;

   cout << "GetReserved ...";
   list = Rental.GetReserved("VW","Golf");
   cout << "done" << endl;

   cout << "PrintReserved ...";
   Rental.PrintReserved(cout);
   cout << "done" << endl;

   cout << "MoveToAvailable ...";
   Rental.MoveToAvailable(MyCar);
   cout << "done" << endl;

   cout << "PrintAvailable ...";
   Rental.PrintAvailable(cout);
   cout << "done" << endl;

   cout << endl << endl;
}

void MultiTestCase()
{
   cout << "Testcase with several entries" << endl;

   CarRental Rental;

   ICar* VW = new SmallCar("VW","Golf");
   ICar* MyCar = new Decorator_AirConditioner(VW);

   ICar* Audi = new PremiumCar("Audi","R8");
   ICar* Xenon = new Decorator_Xenion(Audi);
   ICar* MySecondCar = new Decorator_Navi(Xenon);

   ICar* MySUV = new SUV("Toyota","RAV4");

   ICar* BMW = new MiddleRangeCar("BMW","3");
   ICar* MyMiddleRangeCar = new Decorator_Speedometer(BMW);

   cout << "Add ...";
   Rental.Add(MyCar);
   Rental.Add(MySecondCar);
   Rental.Add(MySUV);
   Rental.Add(MyMiddleRangeCar);
   cout << "done" << endl;

   cout << "GetAvailable ...";
   TCarList list = Rental.GetAvailable("VW","Golf");
   cout << "done" << endl;

   cout << "Reserve ...";
   Rental.Reserve(MySecondCar);
   Rental.Reserve(MySUV);
   cout << "done" << endl;

   cout << "GetReserved ...";
   list = Rental.GetReserved("VW","Golf");
   cout << "done" << endl;

   cout << "PrintReserved ...";
   Rental.PrintReserved(cout);
   cout << "done" << endl;

   cout << "MoveToAvailable ...";
   Rental.MoveToAvailable(MySUV);
   cout << "done" << endl;

   cout << "PrintAvailable ...";
   Rental.PrintAvailable(cout);
   cout << "done" << endl;

   cout << endl << endl;
}

int main()
{
   EmptyTestCase();
   SingleTestCase();
   MultiTestCase();

   return 0;
}
```

6 Testausgaben

```
Visual Leak Detector Version 2.2.3 installed.
Empty testcase with NULL pointer.
Error in CarRental::Add: no valid pointer
Error in CarRental::MoveToAvailable: no valid pointer
Error in CarRental::Reserve: no valid pointer


Testcase with single entry
Add ...done
GetAvailable ...done
Reserve ...done
GetReserved ...done
PrintReserved ...Small Car: VW Golf - Price: 7500
Air Conditioner - Price: 1500
Total price: 9000
done
MoveToAvailable ...done
PrintAvailable ...Small Car: VW Golf - Price: 7500
Air Conditioner - Price: 1500
Total price: 9000
done


Testcase with several entries
Add ...done
GetAvailable ...done
Reserve ...done
GetReserved ...done
PrintReserved ...Premium Car: Audi R8 - Price: 45000
Xenion - Price: 3000
Navi - Price: 2000
Total price: 50000
SUV: Toyota RAV4 - Price: 22000
Total price: 22000
done
MoveToAvailable ...done
PrintAvailable ...Small Car: VW Golf - Price: 7500
Air Conditioner - Price: 1500
Total price: 9000
Middlerange Car: BMW 3 - Price: 16000
Speedometer - Price: 2500
Total price: 18500
SUV: Toyota RAV4 - Price: 22000
Total price: 22000
done


No memory leaks detected.
```