

# 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

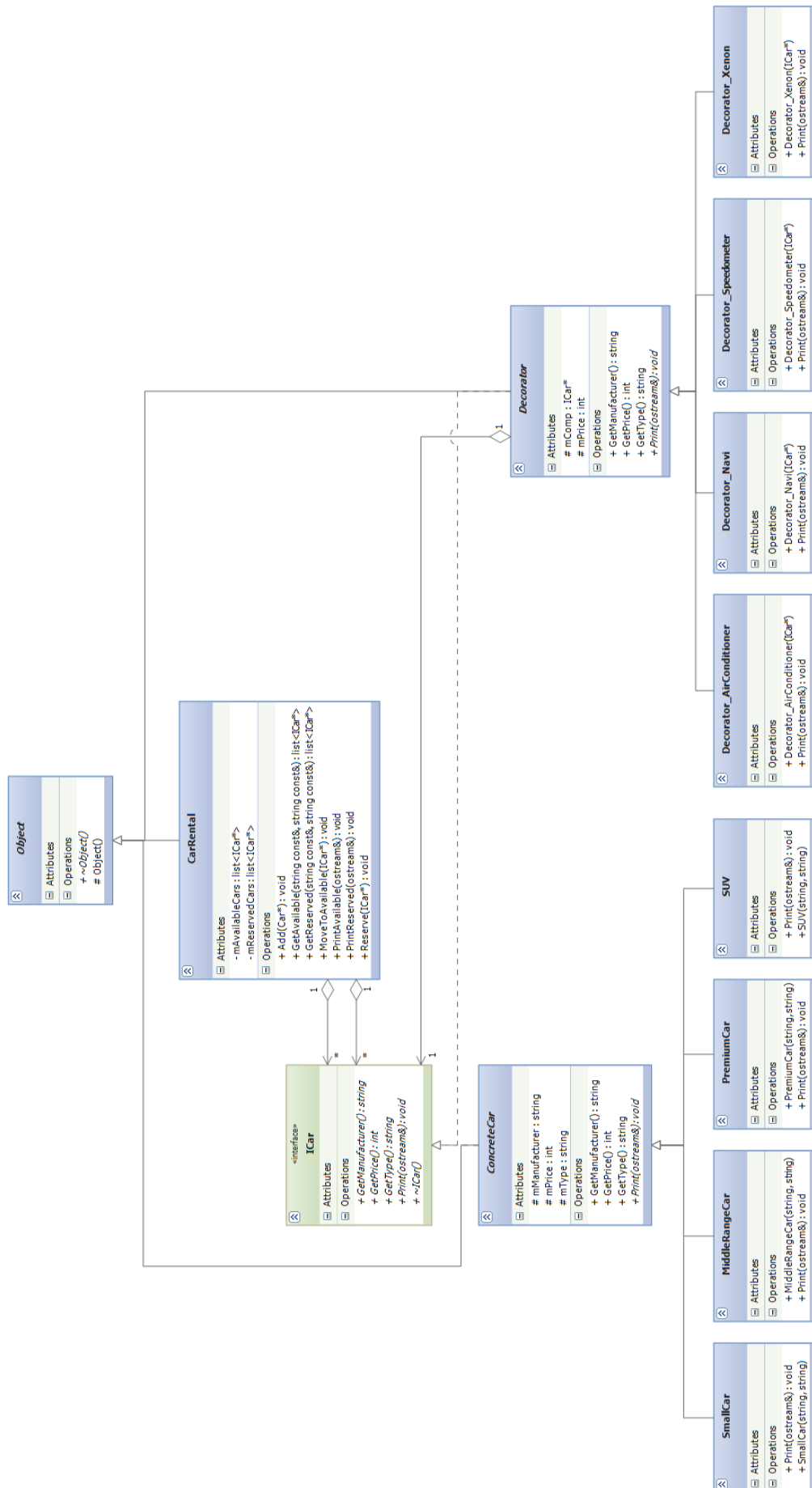
- geschätzte Mh: 12
- tatsächlich: Reinhard (8h), Bernhard (7h)

# 2 Systemspezifikation

Eine Software 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&

Rückgabety: 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&

Rückgabetyt: 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&

Rückgabetyt: 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 &

Rückgabetyt: 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

```
1  //////////////////////////////////////
2  // Workfile : Object.h
3  // Author : Reinhard Penn, Bernhard Selymes
4  // Date : 6.11.2012
5  // Description : Header for Object.cpp
6  //////////////////////////////////////
7
8  #ifndef OBJECT_H
9  #define OBJECT_H
10
11  class Object
12  {
13  public:
14      //virtual Destructor for baseclass
15      virtual ~Object();
16  protected:
17      //Default Ctor for baseclass
18      Object();
19  };
20
21  #endif

```

```
1  //////////////////////////////////////
2  // Workfile : Object.cpp
3  // Author : Reinhard Penn, Bernhard Selymes
4  // Date : 6.11.2012
5  // Description : Baseclass with protected constructor
6  //////////////////////////////////////
7
8  #include "Object.h"
9
10 Object::Object()
11 {}
12
13 Object::~~Object()
14 {}

```

```
1  //////////////////////////////////////
2  // Workfile : ICar.h
3  // Author : Reinhard Penn, Bernhard Selymes
4  // Date : 6.11.2012
5  // Description : Interface
6  //////////////////////////////////////
7
8  #ifndef ICAR_H
9  #define ICAR_H
10
11  class ICar
12  {
13  public:
14      //virtual Dtor
15      ~ICar() {};
16
17      virtual int GetPrice() const = 0;
18      virtual void Print(std::ostream& ost) = 0;
19      virtual std::string GetManufacturer() const = 0;
20      virtual std::string GetType() const = 0;
21  };
22
23  #endif
```

```

1  //////////////////////////////////////
2  // Workfile : ConcreteCar.h
3  // Author : Reinhard Penn, Bernhard Selymes
4  // Date : 6.11.2012
5  // Description : Header for ConcreteCar.cpp
6  //////////////////////////////////////
7
8  #ifndef CONCRETECAR_H
9  #define CONCRETECAR_H
10
11 #include <fstream>
12 #include <string>
13 #include "Object.h"
14 #include "ICar.h"
15
16 class ConcreteCar :
17     public Object,
18     public ICar
19 {
20 public:
21     std::string GetManufacturer() const;
22     int GetPrice() const;
23     std::string GetType() const;
24     virtual void Print(std::ostream& stream) = 0;
25
26 protected:
27     std::string mManufacturer;
28     int mPrice;
29     std::string mType;
30 };
31
32 #endif

```



```
1  //////////////////////////////////////
2  // Workfile : ConcreteCar.cpp
3  // Author : Reinhard Penn, Bernhard Selymes
4  // Date : 6.11.2012
5  // Description : Implementation of class ConcreteCar
6  //////////////////////////////////////
7
8  #include "ConcreteCar.h"
9
10 std::string ConcreteCar::GetManufacturer() const
11 {
12     return mManufacturer;
13 }
14
15 int ConcreteCar::GetPrice() const
16 {
17     return mPrice;
18 }
19
20 std::string ConcreteCar::GetType() const
21 {
22     return mType;
23 }
```

```
1  //////////////////////////////////////
2  // Workfile : SmallCar.h
3  // Author : Reinhard Penn, Bernhard Selymes
4  // Date : 6.11.2012
5  // Description : Header for SmallCar.cpp
6  //////////////////////////////////////
7
8  #ifndef SMALLCAR_H
9  #define SMALLCAR_H
10
11  #include <string>
12  #include "ConcreteCar.h"
13
14  std::size_t const priceSmallCar = 7500;
15
16  class SmallCar :
17      public ConcreteCar
18  {
19  public:
20      SmallCar(std::string manufacturer, std::string type);
21      void Print(std::ostream& stream);
22  };
23
24  #endif
```

```

1  //////////////////////////////////////
2  // Workfile : SmallCar.cpp
3  // Author : Reinhard Penn, Bernhard Selymes
4  // Date : 6.11.2012
5  // Description : Implementation of class SmallCar
6  //////////////////////////////////////
7
8  #include <iostream>
9  #include "SmallCar.h"
10
11 SmallCar::SmallCar(std::string manufacturer, std::string type)
12 {
13     try
14     {
15         if(manufacturer == "")
16         {
17             std::string error = "no valid manufacturer";
18             throw (error);
19         }
20         if(type == "")
21         {
22             std::string error = "no valid type";
23             throw (error);
24         }
25         mManufacturer = manufacturer;
26         mPrice = priceSmallCar;
27         mType = type;
28     }
29     catch (std::string const& error)
30     {
31         std::cout << "Error in SmallCar::SmallCar: " << error << std::endl;
32     }
33     catch(...)
34     {
35         std::cerr << "SmallCar::SmallCar: Unknown Exception occured" << std::
36             endl;
37     }
38
39 void SmallCar::Print(std::ostream& stream)
40 {
41     try
42     {
43         if(stream == 0)
44         {
45             std::string error = "no valid stream";
46             throw (error);
47         }
48         stream << "Small Car: " << mManufacturer << " " << mType
49             << " - Price: " << mPrice << std::endl;
50     }
51     catch (std::string const& error)
52     {
53         std::cout << "Error in SmallCar::Print: " << error << std::endl;
54     }
55     catch(...)
56     {
57         std::cerr << "SmallCar::Print: Unknown Exception occured" << std::
58             endl;
59     }

```



```

1  //////////////////////////////////////
2  // Workfile : MiddleRangeCar.h
3  // Author : Reinhard Penn, Bernhard Selymes
4  // Date : 6.11.2012
5  // Description : Header for MiddleRange.cpp
6  //////////////////////////////////////
7
8  #ifndef MIDDLERANGECAR_H
9  #define MIDDLERANGECAR_H
10
11  #include <string>
12  #include "ConcreteCar.h"
13
14  std::size_t const priceMiddleRangeCar = 16000;
15
16  class MiddleRangeCar :
17      public ConcreteCar
18  {
19  public:
20      MiddleRangeCar(std::string manufacturer, std::string type);
21      void Print(std::ostream& stream);
22  };
23
24  #endif

```

```

1  //////////////////////////////////////
2  // Workfile : MiddleRangeCar.cpp
3  // Author : Reinhard Penn, Bernhard Selymes
4  // Date : 6.11.2012
5  // Description : Implementation of class MiddleRangeCar
6  //////////////////////////////////////
7
8  #include <iostream>
9  #include "MiddleRangeCar.h"
10
11 MiddleRangeCar::MiddleRangeCar(std::string manufacturer, std::string type)
12 {
13     try
14     {
15         if(manufacturer == "")
16         {
17             std::string error = "no valid manufacturer";
18             throw (error);
19         }
20         if(type == "")
21         {
22             std::string error = "no valid type";
23             throw (error);
24         }
25         mManufacturer = manufacturer;
26         mPrice = priceMiddleRangeCar;
27         mType = type;
28     }
29     catch (std::string const& error)
30     {
31         std::cout << "Error in MiddleRangeCar::MiddleRangeCar: " << error <<
            std::endl;
32     }
33     catch(...)
34     {
35         std::cerr << "MiddleRangeCar::MiddleRangeCar: Unknown Exception
            occured" << std::endl;
36     }
37 }
38
39 void MiddleRangeCar::Print(std::ostream& stream)
40 {
41     try
42     {
43         if(stream == 0)
44         {
45             std::string error = "no valid stream";
46             throw (error);
47         }
48         stream << "Middlerange Car: " << mManufacturer << " " << mType
49             << " - Price: " << mPrice << std::endl;
50     }
51     catch (std::string const& error)
52     {
53         std::cout << "Error in MiddleRangeCar::Print: " << error << std::endl
54             ;
55     }
56     catch(...)
57     {
58         std::cerr << "MiddleRangeCar::Print: Unknown Exception occured" <<

```

```
std::endl;
```

```
58     }
```

```
59 }
```

```
1  //////////////////////////////////////
2  // Workfile : PremiumCar.h
3  // Author : Reinhard Penn, Bernhard Selymes
4  // Date : 6.11.2012
5  // Description : Header for PremiumCar.cpp
6  //////////////////////////////////////
7
8  #ifndef PREMIUMCAR_H
9  #define PREMIUMCAR_H
10
11  #include <string>
12  #include "ConcreteCar.h"
13
14  std::size_t const pricePremiumCar = 45000;
15
16  class PremiumCar :
17      public ConcreteCar
18  {
19  public:
20      PremiumCar(std::string manufacturer, std::string type);
21      void Print(std::ostream& stream);
22  };
23
24  #endif
```



```

1  //////////////////////////////////////
2  // Workfile : PremiumCar.cpp
3  // Author : Reinhard Penn, Bernhard Selymes
4  // Date : 6.11.2012
5  // Description : Implementation of class PremiumCar
6  //////////////////////////////////////
7
8  #include <iostream>
9  #include "PremiumCar.h"
10
11 PremiumCar::PremiumCar(std::string manufacturer, std::string type)
12 {
13     try
14     {
15         if(manufacturer == "")
16         {
17             std::string error = "no valid manufacturer";
18             throw (error);
19         }
20         if(type == "")
21         {
22             std::string error = "no valid type";
23             throw (error);
24         }
25         mManufacturer = manufacturer;
26         mPrice = pricePremiumCar;
27         mType = type;
28     }
29     catch (std::string const& error)
30     {
31         std::cout << "Error in PremiumCar::PremiumCar: " << error << std:::
32             endl;
33     }
34     catch(...)
35     {
36         std::cerr << "PremiumCar::PremiumCar: Unknown Exception occured" <<
37             std::endl;
38     }
39 }
40
41 void PremiumCar::Print(std::ostream& stream)
42 {
43     try
44     {
45         if(stream == 0)
46         {
47             std::string error = "no valid stream";
48             throw (error);
49         }
50         stream << "Premium Car: " << mManufacturer << " " << mType
51             << " - Price: " << mPrice << std::endl;
52     }
53     catch (std::string const& error)
54     {
55         std::cout << "Error in PremiumCar::Print: " << error << std::endl;
56     }
57     catch(...)
58     {
59         std::cerr << "PremiumCar::Print: Unknown Exception occured" << std:::
60             endl;
61     }
62 }

```

58        }  
59    }

```

1  //////////////////////////////////////
2  // Workfile : SUV.h
3  // Author : Reinhard Penn, Bernhard Selymes
4  // Date : 6.11.2012
5  // Description : Header for SUV.cpp
6  //////////////////////////////////////
7
8  #ifndef SUV_H
9  #define SUV_H
10
11  #include <string>
12  #include "ConcreteCar.h"
13
14  std::size_t const priceSUV = 22000;
15
16  class SUV :
17      public ConcreteCar
18  {
19  public:
20      SUV(std::string manufacturer, std::string type);
21      void Print(std::ostream& stream);
22  };
23
24  #endif

```

```

1  //////////////////////////////////////
2  // Workfile : SUV.cpp
3  // Author : Reinhard Penn, Bernhard Selymes
4  // Date : 6.11.2012
5  // Description : Implementation of class SUV
6  //////////////////////////////////////
7
8  #include <iostream>
9  #include "SUV.h"
10
11 SUV::SUV(std::string manufacturer, std::string type)
12 {
13     try
14     {
15         if(manufacturer == "")
16         {
17             std::string error = "no valid manufacturer";
18             throw (error);
19         }
20         if(type == "")
21         {
22             std::string error = "no valid type";
23             throw (error);
24         }
25         mManufacturer = manufacturer;
26         mPrice = priceSUV;
27         mType = type;
28     }
29     catch (std::string const& error)
30     {
31         std::cout << "Error in SUV::SUV: " << error << std::endl;
32     }
33     catch(...)
34     {
35         std::cerr << "SUV::SUV: Unknown Exception occured" << std::endl;
36     }
37 }
38
39 void SUV::Print(std::ostream& stream)
40 {
41     try
42     {
43         if(stream == 0)
44         {
45             std::string error = "no valid stream";
46             throw (error);
47         }
48         stream << "SUV: " << mManufacturer << " " << mType
49             << " - Price: " << mPrice << std::endl;
50     }
51     catch (std::string const& error)
52     {
53         std::cout << "Error in SUV::Print: " << error << std::endl;
54     }
55     catch(...)
56     {
57         std::cerr << "SUV::Print: Unknown Exception occured" << std::endl;
58     }
59 }

```

```

1
2 //////////////////////////////////////////////////
3 // Workfile : Decorator.h
4 // Author : Reinhard Penn, Bernhard Selymes
5 // Date : 6.11.2012
6 // Description : Header for Decorator.cpp
7 //////////////////////////////////////////////////
8
9 #ifndef DECORATOR_H
10 #define DECORATOR_H
11
12 #include <string>
13 #include <fstream>
14 #include "Object.h"
15 #include "ICar.h"
16
17 class Decorator :
18     public Object,
19     public ICar
20 {
21 public:
22     std::string GetManufacturer() const;
23     int GetPrice() const;
24     std::string GetType() const;
25     void Print(std::ostream& stream) = 0;
26
27 protected:
28     ICar* mComp;
29     int mPrice;
30 };
31
32 #endif

```

```

1
2  //////////////////////////////////////
3  // Workfile : Decorator.cpp
4  // Author : Reinhard Penn, Bernhard Selymes
5  // Date : 6.11.2012
6  // Description : Implementation of class Decorator
7  //////////////////////////////////////
8
9  #include <iostream>
10 #include "Decorator.h"
11
12 std::string Decorator::GetManufacturer() const
13 {
14     return mComp->GetManufacturer();
15 }
16
17 //returns the price of the whole car (incl. all decorators)
18 int Decorator::GetPrice() const
19 {
20     return mPrice + mComp->GetPrice();
21 }
22
23 std::string Decorator::GetType() const
24 {
25     return mComp->GetType();
26 }

```

```
1  //////////////////////////////////////
2  // Workfile : Decorator_AirConditioner.h
3  // Author : Reinhard Penn, Bernhard Selymes
4  // Date : 6.11.2012
5  // Description : Header for Decorator_AirConditioner.cpp
6  //////////////////////////////////////
7
8  #ifndef DECORATOR_AIRCONDITIONER_H
9  #define DECORATOR_AIRCONDITIONER_H
10
11  #include "Decorator.h"
12
13  int const airConditionerPrice = 1500;
14
15  class Decorator_AirConditioner :
16      public Decorator
17  {
18  public:
19      Decorator_AirConditioner(ICar* car);
20      void Print(std::ostream& stream);
21  };
22
23  #endif
```

```

1  //////////////////////////////////////
2  // Workfile : Decorator_AirConditioner.cpp
3  // Author : Reinhard Penn, Bernhard Selymes
4  // Date : 6.11.2012
5  // Description : Implementation of class Decorator_AirConditioner
6  //////////////////////////////////////
7
8  #include <iostream>
9  #include "Decorator_AirConditioner.h"
10
11 Decorator_AirConditioner::Decorator_AirConditioner(ICar* car)
12 {
13     try
14     {
15         if(car == 0)
16         {
17             std::string error = "no valid pointer";
18             throw (error);
19         }
20         mComp = car;
21         mPrice = airConditionerPrice;
22     }
23     catch (std::string const& error)
24     {
25         std::cout << "Error in Decorator_AirConditioner::
26             Decorator_AirConditioner: " << error << std::endl;
27     }
28     catch(...)
29     {
30         std::cerr << "Decorator_AirConditioner::Decorator_AirConditioner:
31             Unknown Exception occured" << std::endl;
32     }
33 }
34
35 void Decorator_AirConditioner::Print(std::ostream& stream)
36 {
37     try
38     {
39         if(stream == 0)
40         {
41             std::string error = "no valid stream";
42             throw (error);
43         }
44         mComp->Print(stream);
45         stream << "Air Conditioner" << " - Price: " << mPrice << std::endl;
46     }
47     catch (std::string const& error)
48     {
49         std::cout << "Error in Decorator_AirConditioner::Print: " << error <<
50             std::endl;
51     }
52     catch(...)
53     {
54         std::cerr << "Decorator_AirConditioner::Print: Unknown Exception
55             occured" << std::endl;
56     }
57 }

```



```
1  //////////////////////////////////////
2  // Workfile : Decorator_Navi.h
3  // Author : Reinhard Penn, Bernhard Selymes
4  // Date : 6.11.2012
5  // Description : Header for Decorator_Navi.cpp
6  //////////////////////////////////////
7
8  #ifndef DECORATOR_NAVI_H
9  #define DECORATOR_NAVI_H
10
11 #include "Decorator.h"
12
13 int const naviPrice = 2000;
14
15 class Decorator_Navi :
16     public Decorator
17 {
18 public:
19     Decorator_Navi(ICar* car);
20     void Print(std::ostream& stream);
21 };
22
23 #endif
```

```

1  //////////////////////////////////////
2  // Workfile : Decorator_Navi.cpp
3  // Author : Reinhard Penn, Bernhard Selymes
4  // Date : 6.11.2012
5  // Description : Implementation of class Decorator_Navi
6  //////////////////////////////////////
7
8  #include "Decorator_Navi.h"
9  #include <iostream>
10
11 Decorator_Navi::Decorator_Navi(ICar* car)
12 {
13     try
14     {
15         if(car == 0)
16         {
17             std::string error = "no valid pointer";
18             throw (error);
19         }
20         mComp = car;
21         mPrice = naviPrice;
22     }
23     catch (std::string const& error)
24     {
25         std::cout << "Error in Decorator_Navi::Decorator_Navi: " << error <<
26             std::endl;
27     }
28     catch(...)
29     {
30         std::cerr << "Decorator_Navi::Decorator_Navi: Unknown Exception
31             occured" << std::endl;
32     }
33 }
34
35 void Decorator_Navi::Print(std::ostream& stream)
36 {
37     try
38     {
39         if(stream == 0)
40         {
41             std::string error = "no valid stream";
42             throw (error);
43         }
44         mComp->Print(stream);
45         stream << "Navi" << " - Price: " << mPrice << std::endl;
46     }
47     catch (std::string const& error)
48     {
49         std::cout << "Error in Decorator_Navi::Print: " << error << std::endl
50             ;
51     }
52     catch(...)
53     {
54         std::cerr << "Decorator_Navi::Print: Unknown Exception occured" <<
55             std::endl;
56     }
57 }

```

```
1  //////////////////////////////////////
2  // Workfile : Decorator_Speedometer.h
3  // Author : Reinhard Penn, Bernhard Selymes
4  // Date : 6.11.2012
5  // Description : Header for Decorator_Speedometer.cpp
6  //////////////////////////////////////
7
8  #ifndef DECORATOR_SPEEDOMETER_H
9  #define DECORATOR_SPEEDOMETER_H
10
11 #include "Decorator.h"
12
13 int const speedometerPrice = 2500;
14
15 class Decorator_Speedometer :
16     public Decorator
17 {
18 public:
19     Decorator_Speedometer(ICar* car);
20     void Print(std::ostream& stream);
21 };
22
23 #endif
```

```

1  //////////////////////////////////////
2  // Workfile : Decorator_Speedometer.cpp
3  // Author : Reinhard Penn, Bernhard Selymes
4  // Date : 6.11.2012
5  // Description : Implementation of class Decorator_Speedometer
6  //////////////////////////////////////
7
8  #include <iostream>
9  #include "Decorator_Speedometer.h"
10
11 Decorator_Speedometer::Decorator_Speedometer(ICar* car)
12 {
13     try
14     {
15         if(car == 0)
16         {
17             std::string error = "no valid pointer";
18             throw (error);
19         }
20         mComp = car;
21         mPrice = speedometerPrice;
22     }
23     catch (std::string const& error)
24     {
25         std::cout << "Error in Decorator_Speedometer::Decorator_Speedometer:
26             " << error << std::endl;
27     }
28     catch(...)
29     {
30         std::cerr << "Decorator_Speedometer::Decorator_Speedometer: Unknown
31             Exception occured" << std::endl;
32     }
33 }
34
35 void Decorator_Speedometer::Print(std::ostream& stream)
36 {
37     try
38     {
39         if(stream == 0)
40         {
41             std::string error = "no valid stream";
42             throw (error);
43         }
44         mComp->Print(stream);
45         stream << "Speedometer" << " - Price: " << mPrice << std::endl;
46     }
47     catch (std::string const& error)
48     {
49         std::cout << "Error in Decorator_Speedometer::Print: " << error <<
50             std::endl;
51     }
52     catch(...)
53     {
54         std::cerr << "Decorator_Speedometer::Print: Unknown Exception occured
55             " << std::endl;
56     }
57 }

```

```
1  //////////////////////////////////////
2  // Workfile : Decorator_Xenion.h
3  // Author : Reinhard Penn, Bernhard Selymes
4  // Date : 6.11.2012
5  // Description : Header for Decorator_Xenion.cpp
6  //////////////////////////////////////
7
8  #ifndef DECORATOR_XENION_H
9  #define DECORATOR_XENION_H
10
11 #include "Decorator.h"
12
13 int const xenionPrice = 3000;
14
15 class Decorator_Xenion :
16     public Decorator
17 {
18 public:
19     Decorator_Xenion(ICar* car);
20     void Print(std::ostream& stream);
21 };
22
23 #endif
```

```

1  //////////////////////////////////////
2  // Workfile : Decorator_Xenion.cpp
3  // Author : Reinhard Penn, Bernhard Selymes
4  // Date : 6.11.2012
5  // Description : Implementation of class Decorator_Xenion
6  //////////////////////////////////////
7
8  #include <iostream>
9  #include "Decorator_Xenion.h"
10
11 Decorator_Xenion::Decorator_Xenion(ICar* car)
12 {
13     try
14     {
15         if(car == 0)
16         {
17             std::string error = "no valid pointer";
18             throw (error);
19         }
20         mComp = car;
21         mPrice = xenionPrice;
22     }
23     catch (std::string const& error)
24     {
25         std::cout << "Error in Decorator_Xenion::Decorator_Xenion: " << error
26             << std::endl;
27     }
28     catch(...)
29     {
30         std::cerr << "Decorator_Xenion::Decorator_Xenion: Unknown Exception
31             occured" << std::endl;
32     }
33 }
34
35 void Decorator_Xenion::Print(std::ostream& stream)
36 {
37     try
38     {
39         if(stream == 0)
40         {
41             std::string error = "no valid stream";
42             throw (error);
43         }
44         mComp->Print(stream);
45         stream << "Xenion" << " - Price: " << mPrice << std::endl;
46     }
47     catch (std::string const& error)
48     {
49         std::cout << "Error in Decorator_Xenion::Print: " << error << std:::
50             endl;
51     }
52     catch(...)
53     {
54         std::cerr << "Decorator_Xenion::Print: Unknown Exception occured" <<
55             std::endl;
56     }
57 }

```

```

1  //////////////////////////////////////
2  // Workfile : CarRental.h
3  // Author : Reinhard Penn, Bernhard Selymes
4  // Date : 18.12.2012
5  // Description : Header for CarRental.cpp
6  //////////////////////////////////////
7
8  #ifndef CARRENTAL_H
9  #define CARRENTAL_H
10
11  #include <list>
12  #include "ICar.h"
13
14  typedef std::list<ICar*> TCarList;
15  typedef TCarList::iterator TCarListItor;
16
17  class CarRental
18  {
19  public:
20      //Destructor
21      virtual ~CarRental();
22
23      void Add(ICar* c);
24      void PrintAvailable(std::ostream& ost) const;
25      void PrintReserved(std::ostream& ost) const;
26      TCarList GetAvailable(std::string const& type="", std::string const&
          manufacturer="") const;
27      TCarList GetReserved(std::string const& type="", std::string const&
          manufacturer="") const;
28      void Reserve(ICar* c);
29      void MoveToAvailable(ICar* c);
30
31  private:
32      TCarList mAvailableCars;
33      TCarList mReservedCars;
34  };
35
36  #endif

```

```

1  //////////////////////////////////////
2  // Workfile : CarRental.cpp
3  // Author : Reinhard Penn, Bernhard Selymes
4  // Date : 18.12.2012
5  // Description : Implementation of class CarRental
6  //////////////////////////////////////
7
8  #include <algorithm>
9  #include <iostream>
10 #include <string>
11 #include "CarRental.h"
12
13
14 CarRental::~CarRental()
15 {
16     std::for_each(mAvailableCars.begin(),mAvailableCars.end(),[&](ICar* m)
17     {
18         delete m;
19     });
20
21     std::for_each(mReservedCars.begin(),mReservedCars.end(),[&](ICar* m)
22     {
23         delete m;
24     });
25 }
26
27 void CarRental::Add(ICar* c)
28 {
29     try
30     {
31         if(c == 0)
32         {
33             std::string error = "no valid pointer";
34             throw (error);
35         }
36         mAvailableCars.push_back(c);
37     }
38     catch (std::string const& error)
39     {
40         std::cout << "Error in CarRental::Add: " << error << std::endl;
41     }
42     catch(...)
43     {
44         std::cerr << "CarRental::Add: Unknown Exception occured" << std::endl
45             ;
46     }
47 }
48
49 void CarRental::PrintAvailable(std::ostream& ost) const
50 {
51     try
52     {
53         if(ost == 0)
54         {
55             std::string error = "no valid stream";
56             throw (error);
57         }
58         std::for_each(mAvailableCars.begin(),mAvailableCars.end(),[&](ICar* m
59             )

```



```

59         {
60             m->Print(ost);
61         });
62     }
63     catch (std::string const& error)
64     {
65         std::cout << "Error in CarRental::PrintAvailable: " << error << std::
            endl;
66     }
67     catch(...)
68     {
69         std::cerr << "CarRental::PrintAvailable: Unknown Exception occured"
            << std::endl;
70     }
71 }
72
73 void CarRental::PrintReserved(std::ostream& ost) const
74 {
75     try
76     {
77         if(ost == 0)
78         {
79             std::string error = "no valid stream";
80             throw (error);
81         }
82
83         std::for_each(mReservedCars.begin(),mReservedCars.end(),[&](ICar* m)
84         {
85             m->Print(ost);
86         });
87     }
88     catch (std::string const& error)
89     {
90         std::cout << "Error in CarRental::PrintReserved: " << error << std::
            endl;
91     }
92     catch(...)
93     {
94         std::cerr << "CarRental::PrintReserved: Unknown Exception occured" <<
            std::endl;
95     }
96 }
97
98 TCarList CarRental::GetAvailable(std::string const& type, std::string const
    & manufacturer) const
99 {
100     TCarList carList;
101
102     std::for_each(mAvailableCars.begin(),mAvailableCars.end(),[&](ICar* m)
103     {
104         if(m->GetManufacturer() == manufacturer && m->GetType() == type)
105         {
106             carList.push_back(m);
107         }
108     });
109
110     return carList;
111 }
112
113 TCarList CarRental::GetReserved(std::string const& type, std::string const&

```

```

        manufacturer) const
114 {
115     TCarList carList;
116
117     std::for_each(mReservedCars.begin(), mReservedCars.end(), [&] (ICar* m)
118     {
119         if(m->GetManufacturer() == manufacturer && m->GetType() == type)
120         {
121             carList.push_back(m);
122         }
123     });
124
125     return carList;
126 }
127
128 void CarRental::Reserve(ICar* c)
129 {
130     try
131     {
132         if(c == 0)
133         {
134             std::string error = "no valid pointer";
135             throw (error);
136         }
137         TCarListItor itor = std::find(mAvailableCars.begin(), mAvailableCars.
            end(), c);
138
139         if(itor == mAvailableCars.end())
140         {
141             std::string error = "car not found";
142             throw (error);
143         }
144
145         mAvailableCars.remove(*itor);
146         mReservedCars.push_back(*itor);
147     }
148     catch (std::string const& error)
149     {
150         std::cout << "Error in CarRental::Reserve: " << error << std::endl;
151     }
152     catch(...)
153     {
154         std::cerr << "CarRental::Reserve: Unknown Exception occured" << std::
            endl;
155     }
156 }
157
158 void CarRental::MoveToAvailable(ICar* c)
159 {
160     try
161     {
162         if(c == 0)
163         {
164             std::string error = "no valid pointer";
165             throw (error);
166         }
167         TCarListItor itor = std::find(mReservedCars.begin(), mReservedCars.end
            (), c);
168
169         if(itor == mReservedCars.end())

```

```
170     {
171         std::string error = "car not found";
172         throw (error);
173     }
174
175     mReservedCars.remove(*itor);
176     mAvailableCars.push_back(*itor);
177 }
178 catch (std::string const& error)
179 {
180     std::cout << "Error in CarRental::MoveToAvailable: " << error << std
        ::endl;
181 }
182 catch(...)
183 {
184     std::cerr << "CarRental::MoveToAvailable: Unknown Exception occured"
        << std::endl;
185 }
186 }
```

```

1  //////////////////////////////////////
2  // Workfile : Main.cpp
3  // Author : Reinhard Penn, Bernhard Selymes
4  // Date : 02.01.2013
5  // Description : Testdriver for CarRental
6  //////////////////////////////////////
7
8  #include <iostream>
9  #include <algorithm>
10 #include <vld.h>
11 #include "ICar.h"
12 #include "CarRental.h"
13 #include "Decorator.h"
14 #include "Decorator_AirConditioner.h"
15 #include "Decorator_Navi.h"
16 #include "Decorator_Speedometer.h"
17 #include "Decorator_Xenion.h"
18 #include "SmallCar.h"
19 #include "MiddleRangeCar.h"
20 #include "PremiumCar.h"
21 #include "SUV.h"
22
23 using namespace std;
24
25
26 void EmptyTestCase()
27 {
28     cout << "Empty testcase with NULL pointer." << endl;
29
30     CarRental Rental;
31
32     Rental.Add(0);
33     Rental.GetAvailable("", "");
34     Rental.GetReserved("", "");
35     Rental.MoveToAvailable(0);
36     Rental.Reserve(0);
37     Rental.PrintAvailable(cout);
38     Rental.PrintReserved(cout);
39
40     cout << endl << endl;
41 }
42
43 void SingleTestCase()
44 {
45     cout << "Testcase with single entry" << endl;
46
47     CarRental Rental;
48
49     ICar* VW = new SmallCar("VW", "Golf");
50     ICar* MyCar = new Decorator_AirConditioner(VW);
51
52     cout << "Add ...";
53     Rental.Add(MyCar);
54     cout << "done" << endl;
55
56     cout << "GetAvailable ...";
57     TCarList list = Rental.GetAvailable("VW", "Golf");
58     cout << "done" << endl;
59
60     cout << "Reserve ...";

```

```

61     for_each(list.begin(), list.end(), [&](ICar* m)
62     {
63         Rental.Reserve(m);
64     });
65     cout << "done" << endl;
66
67     cout << "GetReserved ...";
68     Rental.GetReserved("VW", "Golf");
69     cout << "done" << endl;
70
71     cout << "PrintReserved ...";
72     Rental.PrintReserved(cout);
73     cout << "done" << endl;
74
75     cout << "MoveToAvailable ...";
76     for_each(list.begin(), list.end(), [&](ICar* m)
77     {
78         Rental.MoveToAvailable(m);
79     });
80     cout << "done" << endl;
81
82     cout << "PrintAvailable ...";
83     Rental.PrintAvailable(cout);
84     cout << "done" << endl;
85 }
86
87 int main()
88 {
89     EmptyTestCase();
90     SingleTestCase();
91
92     return 0;
93 }

```

## **6 Testausgaben**