1 Organisatorisches

1.1 Team

- Reinhard Penn, s1110306019
- Bernhard Selymes, s1110306024

1.2 Aufteilung

- Reinhard Penn
  - Planung
  - Klassendiagramm
  - Implementierung der Klassen Cryptographer, CryptographerCaesar, NortelNetworksAdapter
  - Dokumentation
- Bernhard Selymes
  - Planung
  - Klassendiagramm
  - Implementierung der Klassen Cryptographer, CryptographerRSA, EpcosAdapter
  - Testen aller Klassen

1.3 Zeitaufwand

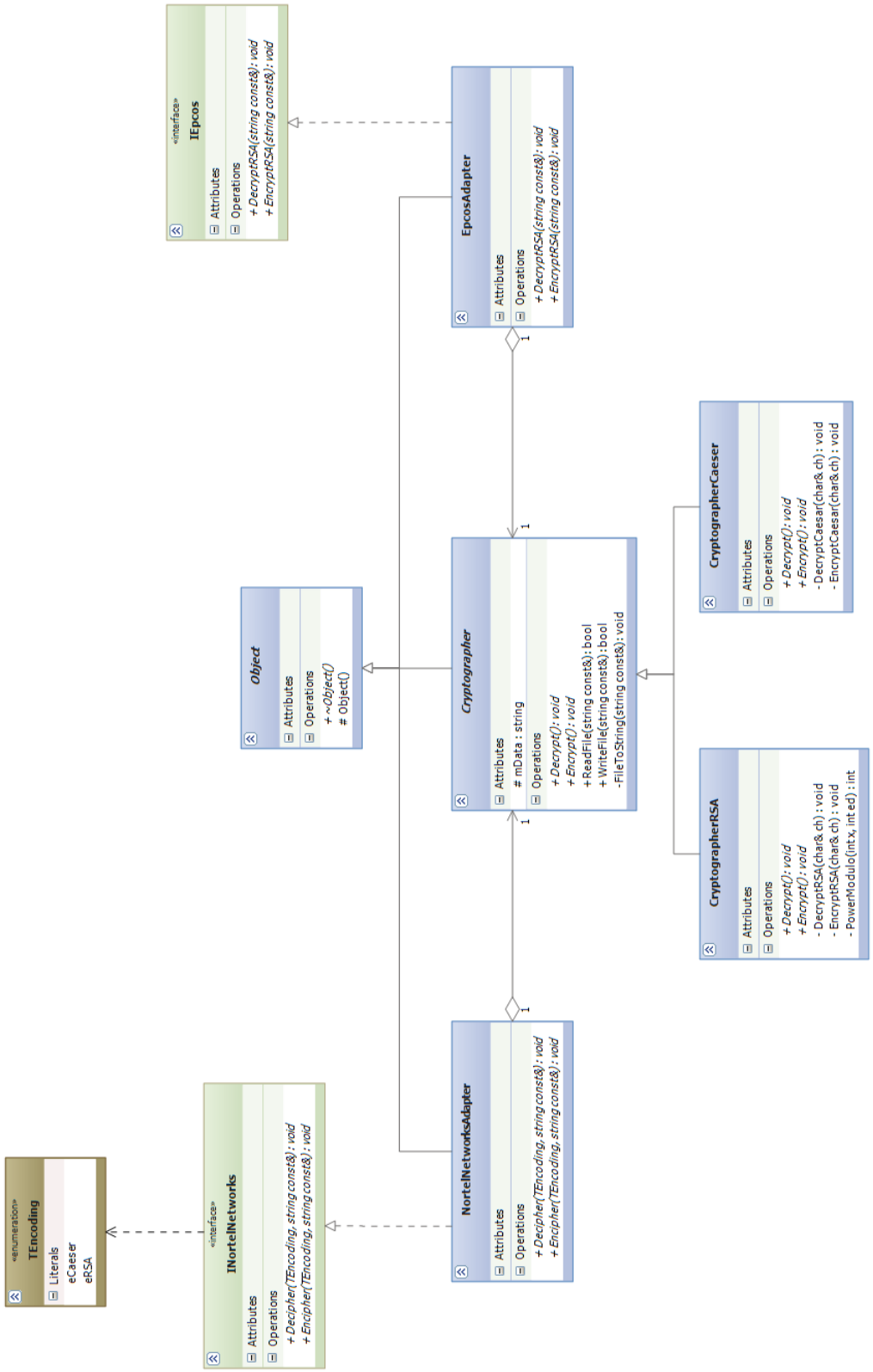
- geschätzte Mh: 7h
- tatsächlich: Reinhard (5h), Bernhard (7h)

2 Systemspezifikation

Es soll für die beiden Firmen Epcos und Nortel Networks ein Verschlüsselungssystem mit jeweils einem firmeneigenen Interface entworfen werden. Die geforderte Verschlüsselungssoftware soll in der Lage sein Daten mithilfe des Caesar und RSA Algorithmus zu verschlüsseln und entschlüsseln.
Das Interface der Firma Epcos bietet nur die RSA Verschlüsselung an.
Das Interface der Firma Nortel Networks bietet beide Verschlüsselungsalgorithmen an. Der entsprechende Algorithmus wird mithilfe einer Enumeration ausgewählt.

3 Systementwurf

3.1 Klassendiagramm



3.2 Komponentenübersicht

- Klasse ”Object”:
  Basis aller Basisklassen.
- Klasse ”Cryptographer”:
  Basisklasse für der Verschlüsselungsalgorithmen.
- Klasse ”CryptographerCaesar”:
  Stellt den Caesar Algorithmus bereit.
- Klasse ”CryptographerRSA”:
  Stellt den RSA Algorithmus bereit.
- Interface ”IEpcos”:
  Interface für die Firma Epcos. Stellt nur RSA zur Verfügung.
- Interface ”INortelNetworks”:
  Interface für die Firma NortelNetworks. Stellt RSA und Caesar zur Verfügung.
- Klasse ”EpcosAdapter”:
  Implementiert das Interface ”IEpcos”.
- Klasse ”NortelNetworksAdapter”:
  Implementiert das Interface ”INortelNetworks”.
- Enumeration ”TEncoding”:
  Dient zur Auswahl des Verschlüsselungsalgorithmus.

4 Komponentenentwurf

4.1 Klasse ”Object”

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

4.2 Klasse ”Cryptographer”

Hat einen String in dem der aktuelle Fileinhalt gespeichert wird.

Methode ”Decrypt”:
Schnittstelle: Rückgabetyp: void.
Pure virtual function.

Methode ”Encrypt”:
Schnittstelle:
Rückgabetyp: void.
Pure virtual function.

Methode ”ReadFile”:
Schnittstelle:
Parameter: string const& filename.
Rückgabetyp: void.
Liest ein File mithilfe von FileToString ein.

Methode ”WriteFile”:
Schnittstelle:
Parameter: string const& filename.
Rückgabetyp: void.
Schreibt die Daten in mData in ein File.

Methode ”FileToString”:
Schnittstelle:
Parameter: string const& filename.
Rückgabetyp: void.
Schreibt die Daten der Datei in den member mData.

4.3 Klasse ”CryptographerCaesar”

Ermöglicht es Daten mithilfe des Caesar Algorithmus zu verschlüsseln und entschlüsseln.

Methode ”Decrypt”:
Schnittstelle: Rückgabetyp: void.
Ruft für jeden Character von mData die Funktion DecryptCaesar auf.

Methode ”Encrypt”:
Schnittstelle:
Rückgabetyp: void.

Ruft für jeden Character von mData die Funktion EncryptCaesar auf.

Methode ”EncryptCaesar”:
Schnittstelle:
Parameter: char& ch.
Rückgabetyp: void.
Verschlüsselt den aktuellen character.

Methode ”DecryptCaesar”:
Schnittstelle:
Parameter: char& ch.
Rückgabetyp: void.
Entschlüsselt den aktuellen character.

4.4 Klasse ”CryptographerRSA”

Ermöglicht es Daten mithilfe des RSA Algorithmus zu verschlüsseln und entschlüsseln.

Methode ”Decrypt”:
Schnittstelle: Rückgabetyp: void.
Ruft für jeden Character von mData die Funktion DecryptRSA auf.

Methode ”Encrypt”:
Schnittstelle:
Rückgabetyp: void.
Ruft für jeden Character von mData die Funktion EncryptRSA auf.

Methode ”EncryptRSA”:
Schnittstelle:
Parameter: char& ch.
Rückgabetyp: void.
Ruft für jeden character die Funktion PowerModulo mit den entsprechenden Werten auf.

Methode ”DecryptRSA”:
Schnittstelle:
Parameter: char& ch.
Rückgabetyp: void.
Ruft für jeden character die Funktion PowerModulo mit den entsprechenden Werten auf.

Methode ”PowerModulo”:
Schnittstelle:
Parameter: int x, int ed.
Rückgabetyp: void.
Berechnet folgenden Ausdruck: x^(e—d) mod n.

4.5 Klasse ”INortelNetworks”

Bietet jeweils eine Funktion zum ent- und verschlüsseln an. Der Algorithmus wird mithilfe einer Enum festgelegt.

Methode ”Decipher”:
Schnittstelle: Parameter: TEncoding encoding, std::string const& filename Rückgabetyp: void.
Pure virtual function.

Methode ”Encipher”:
Schnittstelle: Parameter: TEncoding encoding, std::string const& filename Rückgabetyp: void.
Pure virtual function.

4.6 Klasse ”NortelNetworksAdapter”

Implementiert das Interface INortelNetworks. Je nach Verschlüsselungsart wird ein dynamischer Cryptographer angelegt, entweder vom Typ RSA oder Caesar. Nach dem De-/Encipher Aufruf wird dieser wieder freigegeben.

Methode ”Decipher”:
Schnittstelle: Parameter: TEncoding encoding, std::string const& filename Rückgabetyp: void.
Die Datei wird mithilfe des entsprechenden Cryptographers ausgelesen und entschlüsselt. Danach wird mit ihm eine neue unverschlüsselete Datei erstellt. Die Dateiendung wird auf .txt gesetzt.

Methode ”Encipher”:
Schnittstelle: Parameter: TEncoding encoding, std::string const& filename Rückgabetyp: void.
Die Datei wird mithilfe des entsprechenden Cryptographers ausgelesen und verschlüsselt. Danach wird mit ihm eine neue verschlüsselte Datei erstellt. Die Dateiendung entspricht dabei der jeweiligen Verschlüsselungstechnick.

4.7 Klasse ”IEpcos”

Bietet jeweils eine Funktion zum ent- und verschlüsseln an. Der Algorithmus ist nicht wählbar, es wird immer RSA verwendet.

Methode ”DecryptRSA”:
Schnittstelle: Parameter: std::string const& filename Rückgabetyp: void.
Pure virtual function.

Methode ”EncryptRSA”:
Schnittstelle: Parameter: std::string const& filename Rückgabetyp: void.
Pure virtual function.

4.8 Klasse ”EpcosAdapter”

Bietet jeweils eine Funktion zum ent- und verschlüsseln an. Der Algorithmus ist nicht wählbar, es wird immer RSA verwendet. Die Ver-/Entschlüsselung erfolgt durch einen dynamisch an-

gelegten CryptographerRSA.

Methode ”DecryptRSA”:
Schnittstelle: Parameter: std::string const& filename Rückgabetyp: void.
Die Datei wird mithilfe des CryptographerRSA ausgelesen und entschlüsselt. Danach wird mit ihm eine neue entschlüsselte Datei erstellt. Die Dateiendung wird auf .txt gesetzt.

Methode ”EncryptRSA”:
Schnittstelle: Parameter: std::string const& filename Rückgabetyp: void.
Die Datei wird mithilfe des CryptographerRSA ausgelesen und verschlüsselt. Danach wird mit ihm eine neue verschlüsselte Datei erstellt. Die Dateiendung wird auf .RSA gesetzt.

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 {}
```

```cpp
////////////////////////////////////////////////////////////////////////
// Workfile : Cryptographer.h
// Author : Reinhard Penn, Bernhard Selymes
// Date : 12.11.2012
// Description : Header of Cryptographer.cpp
////////////////////////////////////////////////////////////////////////

#ifndef CRYPTOGRAPHER_H
#define CRYPTOGRAPHER_H

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

std::string const extensionRSA = ".RSA";
std::string const extensionCaesar = ".Caesar";
std::string const extensionDecrypted = ".Decrypted";

class Cryptographer :
   public Object
{
public:
   virtual ~Cryptographer();
   virtual void Decrypt() = 0;
   virtual void Encrypt() = 0;
   void ReadFile(std::string const& filename);
   void WriteFile(std::string const& filename);
protected:
   std::string mData;
private:
   void FileToString(std::string const& filename);
};

#endif
```

```cpp
////////////////////////////////////////////////////////////////////////
// Workfile : Cryptographer.cpp
// Author : Reinhard Penn, Bernhard Selymes
// Date : 12.11.2012
// Description : Implementation of class Cryptographer
////////////////////////////////////////////////////////////////////////

#include <fstream>
#include <sstream>
#include <string>
#include "Cryptographer.h"

Cryptographer::~Cryptographer()
{}

void Cryptographer::ReadFile(std::string const& filename)
{
   FileToString(filename);
}

void Cryptographer::WriteFile(std::string const& filename)
{
   std::fstream file(filename, std::fstream::out); //create new file, if it
       doesnt exsist
   if(!file.is_open())
   {
      std::string ex("File couldn't be opened");
      throw(ex);
   }
   file << mData;
   file.close();
}

void Cryptographer::FileToString(std::string const& filename)
{
      std::ifstream file(filename);
      if (!file.is_open())
      {
         std::string ex("File couldn't be opened");
         throw(ex);
      }
      mData = std::string(std::istreambuf_iterator<char>(file),std::
          istreambuf_iterator<char>());
      file.close();
}
```

```
1  ////////////////////////////////////////////////////////////////////////
2  // Workfile : CryptographerCaesar.h
3  // Author : Reinhard Penn, Bernhard Selymes
4  // Date : 12.11.2012
5  // Description : Header of CryptographerCaesar.cpp
6  ////////////////////////////////////////////////////////////////////////
7
8  #ifndef CRYPTOGRAPHERCAESAR_H
9  #define CRYPTOGRAPHERCAESAR_H
10
11 #include "Cryptographer.h"
12
13 int const Characters = 256;
14 int const key = 7;
15
16 class CryptographerCaeser :
17    public Cryptographer
18 {
19 public:
20    void Decrypt();
21    void Encrypt();
22 private:
23    void EncryptCaesar(char& ch);
24    void DecryptCaesar(char& ch);
25 };
26
27 #endif
```

```cpp
////////////////////////////////////////////////////////////////////////////
// Workfile : CryptographerCaesar.cpp
// Author : Reinhard Penn, Bernhard Selymes
// Date : 12.11.2012
// Description : Implementation of class CryptographerCaesar
////////////////////////////////////////////////////////////////////////////

#include "CryptographerCaeser.h"

void CryptographerCaeser::Decrypt()
{
   std::string::iterator itor;
   for(itor = mData.begin();itor != mData.end();++itor)
   {
      DecryptCaesar(*itor);
   }
}

void CryptographerCaeser::Encrypt()
{
   std::string::iterator itor;
   for(itor = mData.begin();itor != mData.end();++itor)
   {
      EncryptCaesar(*itor);
   }
}

void CryptographerCaeser::EncryptCaesar(char& ch)
{
   ch = (ch + key) % Characters;
}

void CryptographerCaeser::DecryptCaesar(char& ch)
{
   ch = (ch - key + Characters) % Characters;
}
```

```
1  ////////////////////////////////////////////////////////////////////////
2  // Workfile : CryptographerRSA.h
3  // Author : Reinhard Penn, Bernhard Selymes
4  // Date : 12.11.2012
5  // Description : Header of CryptographerRSA.cpp
6  ////////////////////////////////////////////////////////////////////////
7
8  #ifndef CRYPTOGRAPHERRSA_H
9  #define CRYPTOGRAPHERRSA_H
10
11 #include "Cryptographer.h"
12
13 int const e = 7;     //public key
14 int const d = 23; //private key
15 int const n = 255;
16
17 class CryptographerRSA :
18    public Cryptographer
19 {
20 public:
21    void Decrypt();
22    void Encrypt();
23 private:
24    void EncryptRSA(char& ch);
25    void DecryptRSA(char& ch);
26    int PowerModulo(int x, int ed);
27 };
28
29 #endif
```

```cpp
////////////////////////////////////////////////////////////////////////
// Workfile : CryptographerRSA.cpp
// Author : Reinhard Penn, Bernhard Selymes
// Date : 12.11.2012
// Description : Implementation of class CryptographerRSA
////////////////////////////////////////////////////////////////////////

#include "CryptographerRSA.h"

void CryptographerRSA::Decrypt()
{
   std::string::iterator itor;
   for(itor = mData.begin();itor != mData.end();++itor)
   {
      DecryptRSA(*itor);
   }
}

void CryptographerRSA::Encrypt()
{
   std::string::iterator itor;
   for(itor = mData.begin();itor != mData.end();++itor)
   {
      EncryptRSA(*itor);
   }
}

void CryptographerRSA::EncryptRSA(char& ch)
{
   unsigned char tmp = ch;
   tmp = PowerModulo(int(tmp),e);
   ch = tmp;
}

void CryptographerRSA::DecryptRSA(char& ch)
{
   unsigned char tmp = ch;
   tmp = PowerModulo(int(tmp),d);
   ch = tmp;
}

//calculate x^(e|d) mod n
int CryptographerRSA::PowerModulo(int x, int ed)
{
   int result = 1;
   for (int i = 0; i < ed; ++i)
   {
      result = (result * x) % n;
   }
   return result;
}
```

```
1  ////////////////////////////////////////////////////////////////////////
2  // Workfile : INortelNetworks.h
3  // Author : Reinhard Penn, Bernhard Selymes
4  // Date : 12.11.2012
5  // Description : Interface for NortelNetworks
6  ////////////////////////////////////////////////////////////////////////
7
8  #ifndef INORTELNETWORKS_H
9  #define INORTELNETWORKS_H
10
11 #include <string>
12 #include "TEncoding.h"
13
14 class INortelNetworks
15 {
16 public:
17    virtual ~INortelNetworks() {};
18    virtual void Decipher(TEncoding encoding, std::string const& filename) =
          0;
19    virtual void Encipher(TEncoding encoding, std::string const& filename) =
          0;
20 };
21
22 #endif
```

```
1  ////////////////////////////////////////////////////////////////////////
2  // Workfile : NortelNetworksAdapter.h
3  // Author : Reinhard Penn, Bernhard Selymes
4  // Date : 12.11.2012
5  // Description : Header of NortelNetworksAdapter.cpp
6  ////////////////////////////////////////////////////////////////////////
7
8  #ifndef NORTELNETWORKSADAPTER_H
9  #define NORTELNETWORKSADAPTER_H
10
11 #include <string>
12 #include "Object.h"
13 #include "INortelNetworks.h"
14 #include "Cryptographer.h"
15
16 class NortelNetworksAdapter :
17    public Object,
18    public INortelNetworks
19 {
20 public:
21    void Decipher(TEncoding encoding, std::string const& filename);
22    void Encipher(TEncoding encoding, std::string const& filename);
23 };
24
25 #endif
```

```cpp
////////////////////////////////////////////////////////////////////////
// Workfile : NortelNetworksAdapter.cpp
// Author : Reinhard Penn, Bernhard Selymes
// Date : 12.11.2012
// Description : Implementation of the Adapter for NortelNetworks
////////////////////////////////////////////////////////////////////////

#include <string>
#include <iostream>
#include "NortelNetworksAdapter.h"
#include "CryptographerCaeser.h"
#include "CryptographerRSA.h"

using namespace std;

void NortelNetworksAdapter::Decipher(TEncoding encoding, string const& filename)
{
   Cryptographer* cryptographer;

   if (encoding == eCaesar)
   {
      cryptographer = new CryptographerCaeser;
   }
   else
   {
      cryptographer = new CryptographerRSA;
   }

   try {
      cryptographer->ReadFile(filename);
      cryptographer->Decrypt();
      if (encoding == eCaesar)
      {
         cryptographer->WriteFile(filename.substr(0,filename.size()-(extensionCaesar.size()))); //e.g. test.txt
      }
      else
      {
         cryptographer->WriteFile(filename.substr(0,filename.size()-(extensionRSA.size()))); //e.g. test.txt
      }
   }
   catch(std::string const& ex)
   {
      std::cerr << "NortelNetworksAdapter::Decipher: " << ex << std::endl;
   }
   catch(...)
   {
      std::cerr << "NortelNetworksAdapter::Decipher: Unknown Exception occured" << std::endl;
   }

   delete cryptographer;
}

void NortelNetworksAdapter::Encipher(TEncoding encoding, string const& filename)
{
   Cryptographer* cryptographer;
```

```
56
57      if (encoding == eCaesar)
58      {
59          cryptographer = new CryptographerCaeser;
60      }
61      else
62      {
63          cryptographer = new CryptographerRSA;
64      }
65
66      try {
67          cryptographer->ReadFile(filename);
68          cryptographer->Encrypt();
69
70          if (encoding == eCaesar)
71          {
72              cryptographer->WriteFile(filename+extensionCaesar);   //e.g. test.
                    txt.Caesar
73          }
74          else
75          {
76              cryptographer->WriteFile(filename+extensionRSA);   //e.g. test.txt
                    .RSA
77          }
78      }
79      catch(std::string const& ex)
80      {
81          std::cerr << "NortelNetworksAdapter::Encipher: " << ex << std::endl;
82      }
83      catch(...)
84      {
85          std::cerr << "NortelNetworksAdapter::Encipher: Unknown Exception
                occured" << std::endl;
86      }
87
88      delete cryptographer;
89  }
```

```
1  ////////////////////////////////////////////////////////////////////////
2  // Workfile : IEpcos.h
3  // Author : Reinhard Penn, Bernhard Selymes
4  // Date : 12.11.2012
5  // Description : Interface for Epcos
6  ////////////////////////////////////////////////////////////////////////
7
8  #ifndef IEPCOS_H
9  #define IEPCOS_H
10
11 #include <string>
12
13 class EpcosAdapter;
14
15 class IEpcos
16 {
17 public:
18    virtual ~IEpcos() {};
19    virtual void DecryptRSA(std::string const& filename) = 0;
20    virtual void EncryptRSA(std::string const& filename) = 0;
21 };
22
23 #endif
```

```
1  ////////////////////////////////////////////////////////////////////////
2  // Workfile : EpcosAdapter.h
3  // Author : Reinhard Penn, Bernhard Selymes
4  // Date : 12.11.2012
5  // Description : Header of EpcosAdapter.cpp
6  ////////////////////////////////////////////////////////////////////////
7
8  #ifndef EPCOSADAPTER_H
9  #define EPCOSADAPTER_H
10
11 #include <string>
12 #include "Object.h"
13 #include "IEpcos.h"
14 #include "Cryptographer.h"
15
16 class EpcosAdapter :
17    public Object,
18    public IEpcos
19 {
20 public:
21    ~EpcosAdapter();
22    void DecryptRSA(std::string const& filename);
23    void EncryptRSA(std::string const& filename);
24 };
25
26 #endif
```

```cpp
////////////////////////////////////////////////////////////////////////
// Workfile : EpcosAdapter.cpp
// Author : Reinhard Penn, Bernhard Selymes
// Date : 12.11.2012
// Description : Implementation of the Adapter for Epcos
////////////////////////////////////////////////////////////////////////

#include <string>
#include <iostream>
#include "EpcosAdapter.h"
#include "CryptographerRSA.h"

EpcosAdapter::~EpcosAdapter()
{
}

void EpcosAdapter::DecryptRSA(std::string const& filename)
{
   Cryptographer* cryptographer = new CryptographerRSA;
   try {
      cryptographer->ReadFile(filename);              //e.g. test.txt.RSA
      cryptographer->Decrypt();
      cryptographer->WriteFile(filename.substr(0,filename.size()-(
         extensionRSA.size()))); //e.g. test.txt
   }
   catch(std::string const& ex)
   {
      std::cerr << "EpcosAdapter.cpp::DecryptRSA: " << ex << std::endl;
   }
   catch(...)
   {
      std::cerr << "EpcosAdapter::DecryptRSA: Unknown Exception occured" <<
          std::endl;
   }
   delete cryptographer;
}

void EpcosAdapter::EncryptRSA(std::string const& filename)
{
   Cryptographer* cryptographer = new CryptographerRSA;
   try {
      cryptographer->ReadFile(filename);              //e.g. test.txt
      cryptographer->Encrypt();
      cryptographer->WriteFile(filename+extensionRSA);   //e.g. test.txt.
         RSA
   }
   catch(std::string const& ex)
   {
      std::cerr << "EpcosAdapter::EncryptRSA: " << ex << std::endl;
   }
   catch(...)
   {
      std::cerr << "EpcosAdapter::EncryptRSA: Unknown Exception occured" <<
          std::endl;
   }
   delete cryptographer;
}
```

6 Testausgaben

```
Visual Leak Detector Version 2.2.3 installed.
Testcase0: File doesnt exist
Encrypt with Epcos, RSA: testcase0.txt ...
EpcosAdapter::EncryptRSA: File couldn't be opened
Finished
Decrypt with Epcos, RSA: testcase0.txt.RSA ...
EpcosAdapter.cpp::DecryptRSA: File couldn't be opened
Finished
Encipher with NortelNetworks, RSA: testcase0.txt ...
NortelNetworksAdapter::Encipher: File couldn't be opened
Finished
Decipher with NortelNetworks, RSA: testcase0.txt.RSA ...
NortelNetworksAdapter::Decipher: File couldn't be opened
Finished
Encipher with NortelNetworks, Caesar: testcase0.txt ...
NortelNetworksAdapter::Encipher: File couldn't be opened
Finished
Decipher with NortelNetworks, Caesar: testcase0.txt.Caesar ...
NortelNetworksAdapter::Decipher: File couldn't be opened
Finished
Testcase1: File is empty
Encrypt with Epcos, RSA: testcase1.txt ... Finished
Decrypt with Epcos, RSA: testcase1.txt.RSA ... Finished
Encipher with NortelNetworks, RSA: testcase1.txt ... Finished
Decipher with NortelNetworks, RSA: testcase1.txt.RSA ... Finished
Encipher with NortelNetworks, Caesar: testcase1.txt ... Finished
Decipher with NortelNetworks, Caesar: testcase1.txt.Caesar ... Finished
Testcase2: Ascii symbols
Encrypt with Epcos, RSA: testcase2.txt ... Finished
Decrypt with Epcos, RSA: testcase2.txt.RSA ... Finished
Encipher with NortelNetworks, RSA: testcase2.txt ... Finished
Decipher with NortelNetworks, RSA: testcase2.txt.RSA ... Finished
Encipher with NortelNetworks, Caesar: testcase2.txt ... Finished
Decipher with NortelNetworks, Caesar: testcase2.txt.Caesar ... Finished
Testcase3: Normal text
Encrypt with Epcos, RSA: testcase3.txt ... Finished
Decrypt with Epcos, RSA: testcase3.txt.RSA ... Finished
Encipher with NortelNetworks, RSA: testcase3.txt ... Finished
Decipher with NortelNetworks, RSA: testcase3.txt.RSA ... Finished
Encipher with NortelNetworks, Caesar: testcase3.txt ... Finished
Decipher with NortelNetworks, Caesar: testcase3.txt.Caesar ... Finished
No memory leaks detected.
Visual Leak Detector is now exiting.
Drücken Sie eine beliebige Taste . . .
```