FH-OÖ Hagenberg/HSD
SDP3, WS 2019
Übung 3

Name(1):	Daniel Weyrer	Abgabetermin:	
Name(2):	Viktoria Streibl	Punkte:	20,5
Übungsgruppe:	Gruppe 1	korrigiert:	THP
Geschätzter Aufwand in Ph:	12 | 12	Effektiver Aufwand in Ph:	8 | 6

Beispiel 1 (24 Punkte) Verschlüsselung: Entwerfen Sie aus der nachfolgend gegebenen Spezifikation ein Klassendiagramm, instanzieren Sie dieses und implementieren Sie die Funktionalität entsprechend:

Die Firma High Speed Software Engineering (HSE) soll für die beiden Kunden Epcos und Nortel Networks ein Verschlüsselungssystem zur Verfügung stellen.

Es werden 2 Verschlüsselungsalgorithmen unterstützt: Caesar und RSA.

Die Algorithmen sollen zur Laufzeit austauschbar sein. Benützen Sie dafür ein entsprechendes Design Pattern. Da die beiden Kunden unterschiedliche Schnittstellen wünschen, verwenden Sie ein internes Interface und delegieren die Aufrufe der beiden Interfaces mit Hilfe eines geeigneten Design Pattern an die interne Schnittstelle.

Schnittstelle von Epcos:

```
virtual void EncryptRSA(std::string const & fileName) = 0;
virtual void DecryptRSA(std::string const & fileName) = 0;
```

Schnittstelle von Nortel Networks:

```
enum TEncoding {
   eRSA,
   eCaesar
};
virtual void Encipher(TEncoding enc, std::string const & fileName) = 0;
virtual void Decipher(TEncoding enc, std::string const & fileName) = 0;
```

Für `fileName` gilt in allen Fällen vereinfachend:

Bei `fileName` = `Message.txt`:
`Message.txt` ist der Klartext
`Message.txt.Caesar` ist die Caesar-verschlüsselte Datei
`Message.txt.RSA` ist die RSA-verschlüsselte Datei

Die jeweiligen Verschlüsselungs-Parameter (key bei Caesar; n,e,d bei RSA) können intern festgelegt werden. Sie brauchen nicht vom Kunden konfigurierbar sein. Wählen Sie für key selbst einen beliebigen Wert. Für RSA können folgende Schlüssel benützt werden: $n = 187$, $e = 7$, $d = 23$.

Schreiben Sie einen Testtreiber, der verschiedene Nachrichtendateien im ASCII-Format (7 Bit) vom Dateisystem einliest und ver- bzw. entschlüsselt. Verwenden Sie dazu jeweils einen Klienten für Epcos und einen für Nortel, die das für sie zur Verfügung gestellte Interface verwenden. Geben Sie die Ergebnisse (soweit druckbar) aus!

Treffen Sie für alle unzureichenden Angaben sinnvolle Annahmen. Verfassen Sie weiters eine Systemdokumentation (Funktionalität, Klassendiagramm, Schnittstellen der beteiligten Klassen, etc)!

Allgemeine Hinweise: Legen Sie bei der Erstellung Ihrer Übung großen Wert auf eine **saubere Strukturierung** und auf eine **sorgfältige Ausarbeitung!** Dokumentieren Sie alle Schnittstellen und versehen Sie Ihre Algorithmen an entscheidenden Stellen ausführlich mit Kommentaren! Testen Sie ihre Implementierungen ausführlich! Geben Sie den **Testoutput** mit ab!

SDP - Exercise 03

winter semester 2019/20

Viktoria Streibl - S1810306013
Daniel Weyrer - S1820306044

November 15, 2019

Contents

1 Organizational

1.1 Team

- Viktoria Streibl - S1810306013
- Daniel Weyrer - S1820306044

1.2 Roles and responsibilities

1.2.1 Jointly

- Planning
- Documentation
- Systemdocumentation
- Class Diagram

1.2.2 Viktoria Streibl

- Clients
  Client Nortel Networks
  Client Epos
- Interfaces
  IEpos
  INortelNetworks
- Adaptors
  AEpos
  ANorteNetworks
- Testdriver

1.2.3 Daniel Weyrer

- Base Class Encryptor
- Derived Classes
  Class RSA
  Class Caesar

1.3 Effort

1.3.1 Viktoria Streibl

- estimated: 12 ph
- actually: 6 ph

1.3.2 Daniel Weyrer

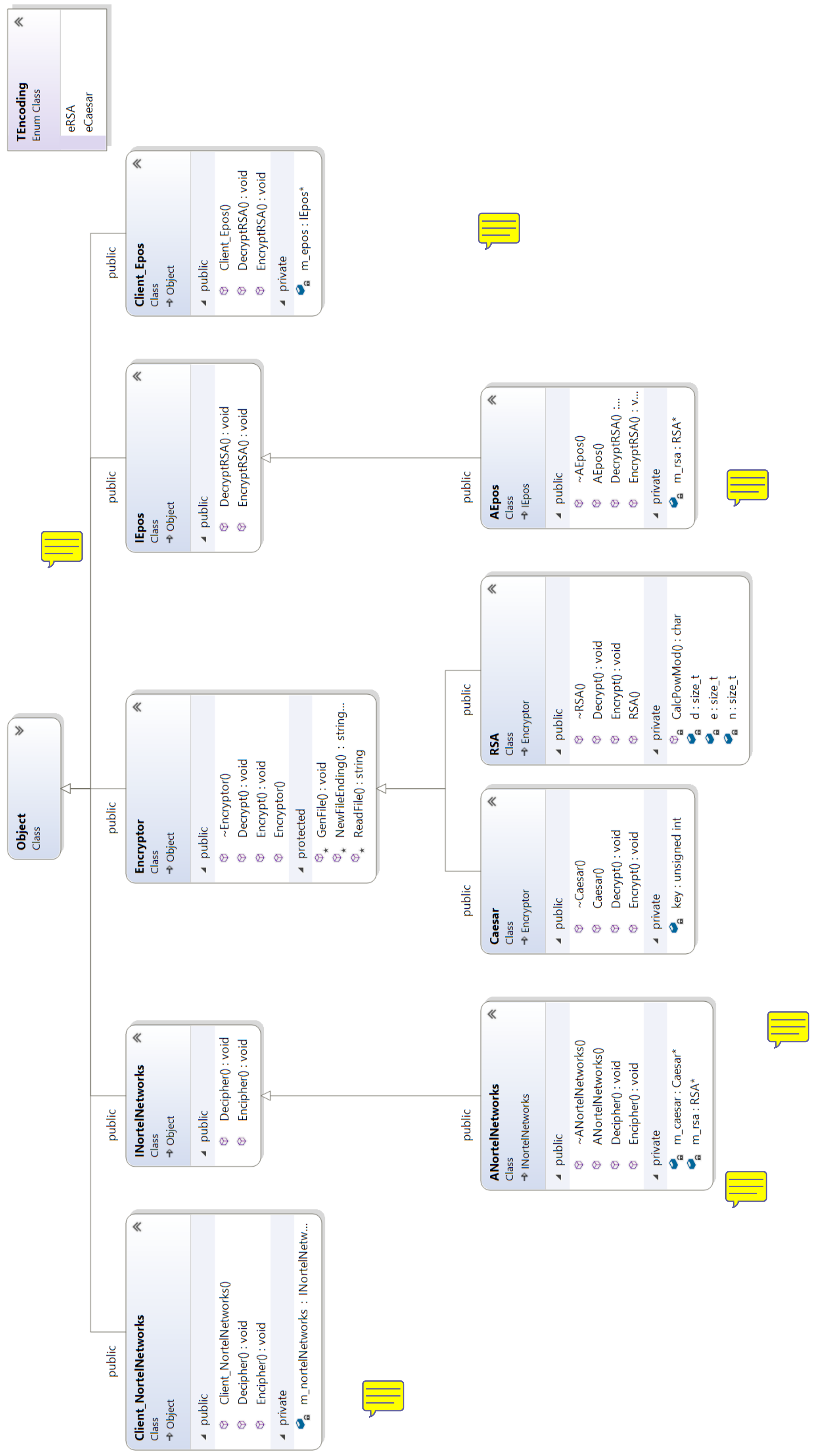
- estimated: 12 ph
- actually: 8 ph

2 Requirenment Definition(System Specification)

The two Companies Epos and Nortel Networks should get access to a new encryption tool via two independent interfaces (which are already given in the information sheet). As the algorithms should be changeable while running the program, we came up with the idea of two adaptors and one base class.

3 System Design

3.1 Classdiagram



3.2 Design Decisions

3.2.1 Reading File into String

The encryption is limited to 7-Bit ASCII-Values, which limits the encryption to plain text files. txt-files in the Megabyte-range are rare and the encryption method is not designed to deal with such many characters (+ it‘s not safe, as we use a small key (RSA) and encrypt character by character).

3.2.2 ASCII-Characters over 127

- Caesar-Encryption: Characters with an ASCII-value over 127 cannot be encrypted due to the modulo division by 127, the Caesar-algorithm uses.

  Encrypting: Characters are written unencrypted into the .Caesar with a warning written in the command-line.

  Decrypting: ASCII-values over 127 in a decrypted file (.caesar) are getting copied over to the decrypted.txt without any decryption. A warning is being delivered to the command-line

- RSA-Encryption

  Encrypting: Characters are being ignored and a warning is being delivered to the cmd, because we found no solution to handle values over 127 when decrypting a message.

  Decrypting: All values are getting decrypted. If a decrypted-value is over 127 a warning is being delivered to the command-line and the decrypted-value is stored in the decrypted.txt

3.2.3 Exception-Handling

- badalloc (thrown by e.g. string::reserve())
- std::exception (thrown by systemfunctions and user)
- unhandled exceptions

All Exceptions are thrown in the base-class (protected, non-public functions) and captured in the derived classes!

3.2.4 Clients

To make the testing easier and avoid code duplication, we decided to not test in the Client-Files this time. We created all tests in the Testdriver and used the client object for testing.

4 Component Design

4.1 Client Epos

This class is the for the company Epos and uses the interface IEpos.

- void EncryptRSA(fileName)

  It calls the method EncryptRSA of the interface.

- void DecryptRSA(fileName)

  It calls the method DecryptRSA of the interface.

4.2 Client Nortel Networks

This class is the for the company Epos and uses the interface INortelNetworks, it contains following functions:

- void Encipher(type, fileName)

  It calls the method Encipher of the interface.

- void Decipher(type, fileName)

  It calls the method Decipher of the interface.

4.3 Interface IEpos

This interface holds the following functions:

- virtual void EncryptRSA(fileName)

  Defines a function for encrypting a file via RSA.

- virtual void DecryptRSA(fileName)

  Defines a function for decrypting a file via RSA.

4.4 Interface INortelNetworks

This interface holds the following functions:

- virtual void Encipher(type, fileName)

  Defines a function for encrypting a file with the algorithm of the specific type.

- virtual void Decipher(type, fileName)

  Defines a function for decrypting a file with the algorithm of the specific type.

4.5 Adaptor AEpos

This class is an adapter for the Interface IEpos. It contains following methods:

- void EncryptRSA(fileName)

  Implements the function of the Interface. It calls the RSA encrypting algorithmn and encrypts the file.

- void DecryptRSA(fileName)

  Implements the function of the Interface. It calls the RSA decrypting algorithmn and decrypts the file.

4.6 ANortelNetworks

This class is an adapter for the Interface INortelNetworks. It contains following methods:

- void Encipher(type, fileName)

  Implements the function of the Interface. It checks which kind of encoding type it should use and calls the respective algorithmn of encrypting.

- void Decipher(type, fileName)

  Implements the function of the Interface. It checks which kind of encoding type it should use and calls the respective algorithmn of decrypting.

4.7 Class Encryptor

Base Class, contains base-functionality such as:

- void GenFile(fileName, content)

  Creates new File with the given Filename and writes the content into the file

- string ReadFile(fileName)

  Reads of the file with the given File-name into a string and returns the string.

- string NewFileEnding(oldFileName, oldFileEnding, newFileEnding (, appendix)

  Checks for correct file-ending of the file and creates a new one with the new file-extension and optional with an appendix (to create ”filename decrypted.txt”). Throws an exception if the file has the wrong file-extension, returns string with new Filename (and extension) otherwise.

4.8 Class Caesar

Derived class, responsible for the Caesar-Encryption

- Caesar()

  Sets default encryption-key

- Encrypt(fileName)

  Main Encryptionfunction, responsible for the whole encryption process:

  Check file for correct ending

  Read file to a String

  create a encrypted string (character by character)

  create a file with the new file extension ”.caesar”

  Write encrypted string to the newly created file

- Decrypt(fileName)

  Main Decryptionfunction, responsible for the whole decryption process:

  Check for correct file extension

  Read File to a string

create a decrypted string

create a new file with ”-decrypted.txt” ending and extension

write the decrypted string to the newly created file

4.9 Class RSA

Derived class, responsible for the RSA-Encryption

- Caesar()

  Sets default encryption-keys

- Encrypt(fileName)

  Main Encryptionfunction, responsible for the whole encryption process:

  Check file for correct ending

  Read file to a String

  create a encrypted string (character by character)

  create a file with the new file extension ”.RSA”

  Write encrypted string to the newly created file

- Decrypt(fileName)

  Main Decryptionfunction, responsible for the whole decryption process:

  Check for correct file extension

  Read File to a string

  create a decrypted string

  create a new file with ”-decrypted.txt” ending and extension

  write the decrypted string to the newly created file

- CalcPowMod(c, pow, mod)

  Based on the RSA Algorithm, it is needed to calculate $c^{pow} \bmod mod$. This function splits the calculation into pieces to avoid high numbers.

4.10 TestDriver

The Testdriver test alle functions of the clients. It tests the interface for the Epos-Company as well as the NortelNetwork-Company. It encrypt and decrypt several files. It contains also some functions:

- int main()

  It calls all tests.

- void CreateFullTest(subtitle, filename)

  This function calls the function to print the title of the tests. Then i tests the Epos functionality and the Nortel Network functionality.

- void testEPOS(fileName)

  It calls at first the encrypting method of the specific file and than the decrypting method.

- void testNN(type, fileName)

  It calls at first the encrypting method of the specific file tests all encoding types, than it decrypts the outcome.

- void PrintSubheader(subtitle)

  This function outputs the title of the following test.

Following tests are implemented:

- Test alphabet and numbers
- Test special characters
- Testing an email file
- Test if no file is there
- Test if file is empty

It ouputs a error message if there was no successful run.

5 Test Protocol

5.1 Testfiles

5.1.1 alphabet.txt

```
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
a b c d e f g h i j k l m n o p q r s t u v w x y z
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
```

5.1.2 specialCharacters.txt

```
+ - ? : . < > * , / ^ ~ { } | @
```

5.1.3 email.txt

```
FROM: dan.womanswarm@fh.at
TO: every.woman@world.com
Title: I love everyone


Dear women,

i need you! Where are you?
Please call me!!!!
Waiting for ya!
My phone number: 0690 / 6969000

Love all of you
Dan
```

5.2 Decrypted Files

5.2.1 alphabet-decrypted.txt

```
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
a b c d e f g h i j k l m n o p q r s t u v w x y z
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
```

5.2.2 specialCharacters-decrypted.txt

```
+ - ? : . < > * , / ^ ~ { } | @
```

5.2.3 email-decrypted.txt

```
FROM: dan.womanswarm@fh.at
TO: every.woman@world.com
Title: I love everyone


Dear women,

i need you! Where are you?
Please call me!!!!
Waiting for ya!
My phone number: 0690 / 6969000

Love all of you
Dan
```

5.3 Console Output

```
###################################
####  Test alphabet and numbers
###################################
Test epos... ...completed!
Test Nortel Networks Caesar...  completed!
Test Nortel Networks RSA...  completed!

###################################
####  Test special characters
###################################
Test epos... ...completed!
Test Nortel Networks Caesar...  completed!
Test Nortel Networks RSA...  completed!

###################################
####  Testing an email file
###################################
Test epos... completed!
Test Nortel Networks Caesar...  completed!
Test Nortel Networks RSA...  completed!

###################################
####  Test if no file is there
###################################
Test epos...
Error while encrypting RSA"../testFiles/youCannotFindMe.txt": Error Reading File!
Error while decrypting RSA "../testFiles/youCannotFindMe.RSA": Error Reading File!
...completed!
Test Nortel Networks Caesar...
Error while encrypting Caesar"../testFiles/youCannotFindMe-c.txt": Error Reading File!
Error while decrypting Caesar"../testFiles/youCannotFindMe-c.Caesar": Error Reading File!
...completed!
Test Nortel Networks RSA...
Error while encrypting RSA"../testFiles/youCannotFindMe-r.txt": Error Reading File!
Error while decrypting RSA "../testFiles/youCannotFindMe-r.RSA": Error Reading File!
...completed!

###################################
####  Test if the file is empty
###################################
Test epos... completed!
Test Nortel Networks Caesar...  completed!
Test Nortel Networks RSA...  completed!
```

6 Source Code

6.1 ClientEpos

6.1.1 ClientEpos.h

```
/* -----------------------------------------------------------------------
| Workfile : Client_Epos.h
| Description : [HEADER] Client uses the Interface IEpos
| Name : Viktoria Streibl     PKZ : S1810306013
| Date : 08.11.2019
| Remarks : -
| Revision : 0
| ---------------------------------------------------------------------- */
#ifndef CLIENT_EPOS_H
#define CLIENT_EPOS_H

#include "Object.h"
#include "IEpos.h"

class Client_Epos : public Object
{
public:
  //Constructor
  Client_Epos(IEpos& epos);

  //calls the encryption logic
  void EncryptRSA(std::string const& fileName);

  //calls the decryption logic
  void DecryptRSA(std::string const& fileName);

private:
  IEpos* m_epos;
};

#endif //CLIENT_EPOS_H
```

6.1.2 ClientEpos.cpp

```
/* -----------------------------------------------------------------------
| Workfile : Client_Epos.cpp
| Description : [SOURCE] Client uses the Interface IEpos
| Name : Viktoria Streibl     PKZ : S1810306013
| Date : 08.11.2019
| Remarks : -
| Revision : 0
| ---------------------------------------------------------------------- */
#include "Client_Epos.h"

Client_Epos::Client_Epos(IEpos& epos)
{
  m_epos = &epos;
}

void Client_Epos::EncryptRSA(std::string const& fileName)
{
  m_epos->EncryptRSA(fileName);
}

void Client_Epos::DecryptRSA(std::string const& fileName)
{
  m_epos->DecryptRSA(fileName);
}
```

6.2 Client Nortel Networks

6.2.1 ClientNortelNetworks.h

```
/* ----------------------------------------------------------------------
| Workfile : Client_NortelNetworks.h
| Description : [HEADER] Client uses the Interface INortelNetworks
| Name : Viktoria Streibl     PKZ : S1810306013
| Date : 08.11.2019
| Remarks : -
| Revision : 0
| ---------------------------------------------------------------------- */
#ifndef CLIENT_NORTEL_NETWORKS_H
#define CLIENT_NORTEL_NETWORKS_H

#include "Object.h"
#include "INortelNetworks.h"

class Client_NortelNetworks : public Object
{
public:
  //Constructor
  Client_NortelNetworks(INortelNetworks& nortelNetworks);

  //calls the encryption by encoding type
  void Encipher(TEncoding type, std::string const& fileName);
  //calls the decryption by encoding type
  void Decipher(TEncoding type, std::string const& fileName);

private:
  INortelNetworks* m_nortelNetworks;
};
#endif //CLIENT_NORTEL_NETWORKS_H
```

6.2.2 ClientNortelNetworks.h

```
/* ----------------------------------------------------------------------
| Workfile : Client_NortelNetworks.cpp
| Description : [SOURCE] Client uses the Interface INortelNetworks
| Name : Viktoria Streibl     PKZ : S1810306013
| Date : 08.11.2019
| Remarks : -
| Revision : 0
| ---------------------------------------------------------------------- */
#include "Client_NortelNetworks.h"

Client_NortelNetworks::Client_NortelNetworks(INortelNetworks& nortelNetworks)
{
  m_nortelNetworks = &nortelNetworks;
}

void Client_NortelNetworks::Encipher(TEncoding type, std::string const& fileName)
{
  m_nortelNetworks->Encipher(type, fileName);
}

void Client_NortelNetworks::Decipher(TEncoding type, std::string const& fileName)
{
  m_nortelNetworks->Decipher(type, fileName);
}
```

6.3 Interface IEpos

6.3.1 IEpos.h

```
/* ______________________________________________________________________
| Workfile : IEpos.h
| Description : [ Interface ] Interface between Client and RSA and Caesar
| Name : Viktoria Streibl     PKZ : S1810306013
| Date : 08.11.2019
| Remarks : -
| Revision : 0
| _______________________________________________________________________ */
#ifndef IEPOS_H
#define IEPOS_H

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

class IEpos : public Object {
public:
  //Default Constructor
  IEpos() = default;
  //Default Destructor
  ~IEpos() = default;

  //calls the encrpytion method for RSA
  virtual void EncryptRSA(std::string const& fileName) = 0;
  //calls the decrpytion method for RSA
  virtual void DecryptRSA(std::string const& fileName) = 0;
};

#endif //IEPOS_H
```

6.4 Interface INortelNetworks

6.4.1 INortelNetworks.h

```
/* ______________________________________________________________________
| Workfile : INortelNetworks.h
| Description : [ Interface ] Interface between Client and RSA and Caesar
| Name : Viktoria Streibl     PKZ : S1810306013
| Date : 08.11.2019
| Remarks : -
| Revision : 0
| _______________________________________________________________________ */
#ifndef INORTEL_NETWORKS_H
#define INORTEL_NETWORKS_H

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

//enum for the encoding types
enum class TEncoding {
  eRSA,
  eCaesar
};

class INortelNetworks : public Object {
public:
  //Default Constructor
  INortelNetworks() = default;
  //Default Destructor
  ~INortelNetworks() = default;

  //calls the correct encryption method by type
  virtual void Encipher(TEncoding enc, std::string const& fileName) = 0;
  //calls the correct decryption method by type
  virtual void Decipher(TEncoding enc, std::string const& fileName) = 0;
};

#endif //INORTEL_NETWORKS_H
```

6.5 Adaptor AEpos

6.5.1 AEpos.h

```
/* ---------------------------------------------------------------------
| Workfile : AEpos.h
| Description : [HEADER] Implements the IEpos interface
| Name : Viktoria Streibl      PKZ : S1810306013
| Date : 08.11.2019
| Remarks : -
| Revision : 0
| --------------------------------------------------------------------- */
#ifndef AEPOS_H
#define AEPOS_H

#include "IEpos.h"
#include "RSA.h"

class AEpos : public IEpos
{
public:
  //Constructor
  AEpos();
  //Default deconstructor
  ~AEpos() = default;

  //encrypt the file with RSA
  void EncryptRSA(std::string const& fileName) override;

  //decrypt the file with RSA
  void DecryptRSA(std::string const& fileName) override;

private:
  RSA* m_rsa;
};

#endif //AEPOS_H
```

6.5.2 AEpos.cpp

```
/* ---------------------------------------------------------------------
| Workfile : AEpos.cpp
| Description : [SOURCE] Implements the IEpos interface
| Name : Viktoria Streibl      PKZ : S1810306013
| Date : 08.11.2019
| Remarks : -
| Revision : 0
| --------------------------------------------------------------------- */
#include "AEpos.h"

AEpos::AEpos() {
  m_rsa = new RSA;
}

void AEpos::EncryptRSA(std::string const& fileName)
{
  m_rsa->Encrypt(fileName);
}

void AEpos::DecryptRSA(std::string const& fileName)
{
  m_rsa->Decrypt(fileName);
}
```

6.6 ANortelNetworks

6.6.1 ANortelNetworks.h

```
/* ----------------------------------------------------------------------
| Workfile : AEpos.h
| Description : [HEADER] Implements the INortelNetwors interface and
        handles which encryption/decryption should be used
| Name : Viktoria Streibl     PKZ : S1810306013
| Date : 08.11.2019
| Remarks : -
| Revision : 0
| ---------------------------------------------------------------------- */
#ifndef ANORTEL_NETWORKS_H
#define ANORTEL_NETWORKS_H

#include "INortelNetworks.h"
#include "RSA.h"
#include "Caesar.h"

class ANortelNetworks : public INortelNetworks
{
public:
  //Constructor
  ANortelNetworks();
  //Default deconstructor
  ~ANortelNetworks() = default;

  //encrypt the file based on the encoding type
  void Encipher(TEncoding enc, std::string const& fileName) override;
  //decrypt the file based on the encoding type
  void Decipher(TEncoding enc, std::string const& fileName) override;

private:
  RSA* m_rsa;
  Caesar* m_caesar;
};

#endif //ANORTEL_NETWORKS_H
```

6.6.2 ANortelNetworks.cpp

```
/* ----------------------------------------------------------------------
| Workfile : AEpos.h
| Description : [SOURCE] Implements the INortelNetwors interface and
        handles which encryption/decryption should be used
| Name : Viktoria Streibl     PKZ : S1810306013
| Date : 08.11.2019
| Remarks : -
| Revision : 0
| ---------------------------------------------------------------------- */
#include "ANortelNetworks.h"

ANortelNetworks::ANortelNetworks() {
  m_caesar = new Caesar;
  m_rsa = new RSA;
}

void ANortelNetworks::Encipher(TEncoding enc, std::string const& fileName)
{
  //check if the encoding type is caesar
  if (enc == TEncoding::eCaesar) {
    m_caesar->Encrypt(fileName);
  }
  else {
    m_rsa->Encrypt(fileName);
  }
}

void ANortelNetworks::Decipher(TEncoding enc, std::string const& fileName)
{
  //check if the encoding type is caesar
  if (enc == TEncoding::eCaesar) {
    m_caesar->Decrypt(fileName);
  }
  else {
    m_rsa->Decrypt(fileName);
  }
}
```

6.7 Class Encryptor

6.7.1 Encryptor.h

```cpp
/* -----------------------------------------------------------------------
| Workfile : Encryptor.h
| Description : [ HEADER ] Base Class for encryptors
| Name : Daniel Weyrer                  PKZ : S1820306044
| Date : 05.11.2019
| Remarks : -
| Revision : 0
| ---------------------------------------------------------------------- */
#ifndef ENCRYPTOR_H
#define ENCRYPTOR_H
#include <fstream>
#include <iostream>
#include <string>
#include <iterator>
#include <algorithm>

#include "Object.h"

class Encryptor : public Object {
public:
  Encryptor() = default;
  virtual ~Encryptor() = default;
  virtual void Encrypt(std::string const& fileName) = 0;
  virtual void Decrypt(std::string const& fileName) = 0;

protected:
  void GenFile(std::string const& fileName, std::string const& content);
  std::string ReadFile(std::string const& fileName);

  std::string NewFileEnding(std::string const& oldFileName, std::string const& oldFileEnding, std::string const& newFileEnding);
  std::string NewFileEnding(std::string const& oldFileName, std::string const& oldFileEnding, std::string const& newFileEnding, std::string const& appendix);

};

#endif // ENCRYPTOR_H
```

6.7.2 Encryptor.cpp

```cpp
/* -----------------------------------------------------------------------
| Workfile : Encryptor.cpp
| Description : [ SOURCE] Base Class for encryptors
| Name : Daniel Weyrer                  PKZ : S1820306044
| Date : 05.11.2019
| Remarks : -
| Revision : 0
| ---------------------------------------------------------------------- */
#include "Encryptor.h"

//Generate file with given FileName and store the given content in it
void Encryptor::GenFile(std::string const& fileName, std::string const& content) {
    //Create File
    std::ofstream outFile{ fileName, std::ios::binary };

    //Check created file; throw exception in case of a fault
    if (!outFile.good() || outFile.fail()) {
      outFile.close();
      throw std::exception("Error creating new File");
    }
    //write content into created file and close it afterwards
    outFile << content;
    outFile.close();
}
//Check Ending of File and add new ending if valid
std::string Encryptor::NewFileEnding(std::string const& oldFileName, std::string const& oldFileEnding, std::string const& newFileEnding) {

  //check for correct file_ending
  std::string::const_iterator it = std::search(oldFileName.cbegin(), oldFileName.cend(),
      oldFileEnding.cbegin(), oldFileEnding.cend());

  if (it == oldFileName.cend()) {
    throw std::exception("wrong file-ending! Check filename.");
  }

  //create new Filename with new FileEnding
  std::string newFileName;

  newFileName.assign(oldFileName.cbegin(), it);
  newFileName += newFileEnding;

  return newFileName;
}

std::string Encryptor::NewFileEnding(std::string const& oldFileName, std::string const&
    oldFileEnding, std::string const& newFileEnding, std::string const& appendix)
{
  std::string tmpFileEnding = appendix + newFileEnding;
  return NewFileEnding(oldFileName, oldFileEnding, tmpFileEnding);
}

std::string Encryptor::ReadFile(std::string const& fileName) {
  std::string tmp;
    std::ifstream inFile{ fileName, std::ios::binary};
    if (inFile.eof() || inFile.fail() || !inFile.good()) {
      inFile.close();
      throw std::exception("Error Reading File!");
    }

    //Seek end, to calc the size of inFile
    inFile.seekg(std::ios::end);
    //need to check flags before new seek, as it deletes the file-flags
    if (inFile.fail()) {
      throw std::exception("Error while calculating size of file!");
    }
    //Reserve the size of inFile in tmp --> more Efficent when copying larger files!
    tmp.reserve(inFile.tellg());
    //Seek beginning to start reading
    inFile.seekg(std::ios::beg);
    if (inFile.fail()) {
      throw std::exception("Error while calculating size of file!");
    }
    //assigning content of inFile to tmp string
    tmp.assign(std::istreambuf_iterator<char>(inFile), std::istreambuf_iterator<char>());
    inFile.close();

    return tmp;
}
```

6.8 Class Caesar

6.8.1 Caesar.h

```cpp
/* ----------------------------------------------------------------------
| Workfile : Caesar.h
| Description : [ SOURCE ] Derived Class to encrypt via Caeser-procedure
| Name : Daniel Weyrer                PKZ : S1820306044
| Date : 05.11.2019
| Remarks : -
| Revision : 0
| ---------------------------------------------------------------------- */
#ifndef CAESAR_H
#define CAESAR_H

#include <algorithm>

#include "Encryptor.h"
static const unsigned int encryptionKey = 98;

class Caesar : public Encryptor {
public:
  Caesar() : key{ encryptionKey } {}
  virtual ~Caesar() override = default;

  // Inherited via Encryptor
  virtual void Encrypt(std::string const& fileName) override;
  virtual void Decrypt(std::string const& fileName) override;


private:
  unsigned int key;
  //helper methods

};

#endif //CAESAR_H
```

6.8.2 Caesar.cpp

```cpp
/* ----------------------------------------------------------------------
| Workfile : Caesar.cpp
| Description : [ SOURCE ] Derived Class to encrypt via Caeser-procedure
| Name : Daniel Weyrer                PKZ : S1820306044
| Date : 05.11.2019
| Remarks : -
| Revision : 0
| ---------------------------------------------------------------------- */
#include "Caesar.h"

static const unsigned int maxNumberASCII = 127;
static const std::string fileEndingCaesar = ".Caesar";
static const std::string fileEndingUnencrypted = ".txt";
static const std::string decryptedFileAppendix = "_decrypted";


//Reads content of given File, encrypts and saves it into a new File with a new FileEnding
void Caesar::Encrypt(std::string const& fileName) {
  try {

    std::string newFileName = Encryptor::NewFileEnding(fileName, fileEndingUnencrypted,
        fileEndingCaesar);

    //Read content of File
    std::string unencrypted = ReadFile(fileName);

    //Lambda Function to Encrypt a single Char
    auto EncryptSingleChar = [this, fileName](char const c) {
      if (c > maxNumberASCII) {
        std::cerr << "Character " << c << " in  file " << fileName << " is not a standard-ASCII
            value!" << std::endl;
        return c;
      }
      char encryptedChar = ((c + key) % maxNumberASCII);
      return encryptedChar;
    };

    std::string encrypted;

    //Iterate through unencrypted string, encrypt every single char and save it into "encrypted"
    std::transform(unencrypted.cbegin(), unencrypted.cend(), std::back_inserter(encrypted),
        EncryptSingleChar);

    //Generate File with encrypted content
    GenFile(newFileName, encrypted);
  }
  catch (std::bad_alloc const& ex) {
    std::cerr << "Memory Allocation Error: " << ex.what() << std::endl;
  }
  catch (std::exception const& ex) {
    std::cerr << "Error while encrypting Caesar" << '"' << fileName << '"' << ": "
      << ex.what() << std::endl;
  }
  catch (...) {
    std::cerr << "Unhandled Exception!" << std::endl;
  }
}

//Decrypts the content of the given file and saves it into a new file
void Caesar::Decrypt(std::string const& fileName) {
  try {
    std::string newFileName = Encryptor::NewFileEnding(fileName, fileEndingCaesar,
        fileEndingUnencrypted, decryptedFileAppendix);

    //read file to String
    std::string encrypted = ReadFile(fileName);


    //Lambdafunction for decrypting a single char
    auto DecryptSingleChar = [this, fileName](char const c) {
      if (c > maxNumberASCII) {
        std::cerr << "Character " << c << " in  file " << fileName << " is not a standard-ASCII
            value!" << std::endl;
        return c;
      }
      char decryptedChar = (((c - key) + maxNumberASCII) % maxNumberASCII);
      return decryptedChar;
    };

    std::string decrypted;

    //Iterate through encrypted string, decrypt every single char and save it into "decrypted"
    std::transform(encrypted.cbegin(), encrypted.cend(), std::back_inserter(decrypted),
        DecryptSingleChar);

    Encryptor::GenFile(newFileName, decrypted);
  }
  catch (std::bad_alloc const& ex) {
    std::cerr << "Memory Allocation Error: " << ex.what() << std::endl;
  }
  catch (std::exception const& ex) {
    std::cerr << "Error while decrypting Caesar" << '"' << fileName << '"' << ": "
      << ex.what() << std::endl;
  }
  catch (...) {
    std::cerr << "Unhandled Exception!" << std::endl;
  }
}
```

6.9 Class RSA

6.9.1 RSA.h

```
/* -----------------------------------------------------------------------
| Workfile : RSA.h
| Description : [ HEADER ] Derived Class to encrypt via RSA technique
| Name : Daniel Weyrer                PKZ : S1820306044
| Date : 05.11.2019
| Remarks : -
| Revision : 0
| ----------------------------------------------------------------------- */
#ifndef RSA_H
#define RSA_H
#include <algorithm>
#include "Encryptor.h"

static const size_t n_default = 187;
static const size_t e_default = 7;
static const size_t d_default = 23;

class RSA : public Encryptor {
public:
  RSA() : n{ n_default }, e{ e_default }, d{ d_default }{}
  virtual ~RSA() override = default;

  // Inherited via Encryptor
  virtual void Encrypt(std::string const& fileName) override;
  virtual void Decrypt(std::string const& fileName) override;

private:
  size_t n;
  size_t e;
  size_t d;

  char CalcPowMod(char const c, unsigned int const pow, unsigned int const mod);
};

#endif //RSA_H
```

6.9.2 RSA.cpp

```
/* -----------------------------------------------------------------------
| Workfile : RSA.cpp
| Description : [ SOURCE ] Derived Class to encrypt via RSA-procedure
| Name : Daniel Weyrer                  PKZ : S1820306044
| Date : 05.11.2019
| Remarks : -
| Revision : 0
| ----------------------------------------------------------------------- */

#include "RSA.h"

static const std::string fileEndingRSA = ".RSA";
static const std::string fileEndingUnencrypted = ".txt";
static const std::string decryptedFileAppendix = "_decrypted";

static const unsigned int maxNumberASCII = 127;

//Reads content of given File, encrypts and saves it into a new File with a new FileEnding
void RSA::Encrypt(std::string const& fileName) {
  try {

    std::string newFileName = Encryptor::NewFileEnding(fileName, fileEndingUnencrypted,
        fileEndingRSA);

    //Read unencrypted file and save it into unencrypted
    std::string unencrypted = ReadFile(fileName);


    auto EncryptSingleChar = [this, &fileName](char const c) {
      if (c > maxNumberASCII) {
```

```cpp
        std::cerr << "Character " << c << " in  file " << fileName << " is not a standard-ASCII value!" << std::endl;
      }
      return CalcPowMod(c, e, n);
    };

    std::string encrypted;

    //iterate through unencrypted string, encrypt every single char and save it into encrypted
    std::transform(unencrypted.cbegin(), unencrypted.cend(), std::back_inserter(encrypted), EncryptSingleChar);

    Encryptor::GenFile(newFileName, encrypted);
  }
  catch (std::bad_alloc const& ex) {
    std::cerr << "Memory Allocation Error: " << ex.what() << std::endl;
  }
  catch (std::exception const& ex) {
    std::cerr << "Error while encrypting RSA" << '"' << fileName << '"' << ": "
          << ex.what() << std::endl;
  }
  catch (...) {
    std::cerr << "Unhandled Exception!" << std::endl;
  }
}

//Decrypts the content of the given file and saves it into a new file
void RSA::Decrypt(std::string const& fileName) {
  try {
    std::string newFileName = Encryptor::NewFileEnding(fileName, fileEndingRSA, fileEndingUnencrypted, decryptedFileAppendix);

    //read content of encrypted file into encrypted
    std::string encrypted = ReadFile(fileName);

    //Lambda for decrypting a single char
    auto DecryptSingleChar = [this, &fileName](char const c) {
      char tmp = CalcPowMod(c, d, n);
      if (tmp > maxNumberASCII) {
        std::cerr << "Character " << c << " in  file " << fileName << " is not a standard-ASCII value!" << std::endl;
      }
      return tmp;
    };

    std::string decrypted;

    //iterate through encrypted string, decrypt it char by char and save it into decrypted
    std::transform(encrypted.cbegin(), encrypted.cend(), std::back_inserter(decrypted), DecryptSingleChar);

    Encryptor::GenFile(newFileName, decrypted);
  }
  catch (std::bad_alloc const& ex) {
    std::cerr << "Memory Allocation Error: " << ex.what() << std::endl;
  }
  catch (std::exception const& ex) {
    std::cerr << "Error while decrypting RSA " << '"' << fileName << '"' << ": ";
    std::cerr<< ex.what() << std::endl;
  }
  catch (...) {
    std::cerr << "Unhandled Exception!" << std::endl;
  }
}

//Calculates (c^pow) % mod; Avoids high numbers by doing it step by step
char RSA::CalcPowMod(char const c, unsigned int const pow, unsigned int const mod) {

  unsigned int tmp, origChar;

  //necessary cast to do the calculation!
  tmp = origChar = static_cast<unsigned char>(c);

  for (size_t i = 1; i < pow; i++) {
    tmp = (tmp * origChar) % mod;
  }
  return static_cast<char>(tmp);
}
```

6.10 TestDriver

6.10.1 TestDriver.h

```cpp
/* -----------------------------------------------------------------------
| Workfile : TestDriver.h
| Description : Tests all functions and interfaces
| Name : Viktoria Streibl     PKZ : S1810306013
| Date : 13.11.2019
| Remarks : -
| Revision : 0
| ----------------------------------------------------------------------- */
#include <iostream>
#include <windows.h>  //used for error-handling: colors

#include "Client_Epos.h"
#include "Client_NortelNetworks.h"

#include "AEpos.h"
#include "ANortelNetworks.h"

AEpos epos;
ANortelNetworks aNN;
Client_Epos c_epos(epos);
Client_NortelNetworks c_nortelNetworks(aNN);

//calls the epos and NN test and also print the subtitle
void CreateFullTest(std::string subtitle, std::string filename);

//tests all function of the epos interface
void test_EPOS(std::string filename);

//tests all functions of the nortel networks interface
void test_NN(TEncoding type, std::string filename);

//print the subtitle of the test
void PrintSubheader(std::string subtitle);
```

6.10.2 TestDriver.cpp

```cpp
/* -----------------------------------------------------------------------
| Workfile : TestDriver.cpp
| Description : Tests all functions and interfaces
| Name : Viktoria Streibl     PKZ : S1810306013
| Date : 13.11.2019
| Remarks : -
| Revision : 0
| ----------------------------------------------------------------------- */
#include "Testdriver.h"

int main()
{
  ///////////////////////////////////////////////
  //   Test alphabet and numbers
  ///////////////////////////////////////////////
  CreateFullTest("Test alphabet and numbers", "alphabet");

  ///////////////////////////////////////////////
  //   Test special characters
  ///////////////////////////////////////////////
  CreateFullTest("Test special characters", "specialCharacters");

  ///////////////////////////////////////////////
  //   Testing an email file
  ///////////////////////////////////////////////
  CreateFullTest("Testing an email file", "email");

  ///////////////////////////////////////////////
  //   Test if no file is there
  ///////////////////////////////////////////////
  CreateFullTest("Test if no file is there", "youCannotFindMe");

  ////////////////////////////////////////////////
  //   Test if file is empty
  ////////////////////////////////////////////////
  CreateFullTest("Test if the file is empty", "emptyFile");

  return 0;
}

void test_EPOS(std::string filename) {
  //call encryt and decrypt of a specific file
  c_epos.EncryptRSA(filename + ".txt");
  c_epos.DecryptRSA(filename + ".RSA");
}

void test_NN(TEncoding type, std::string filename) {
  //checks if it is ceasar of RSA and pass the correct file
  if (type == TEncoding::eCaesar) {
    c_nortelNetworks.Encipher(TEncoding::eCaesar, filename + ".txt");
    c_nortelNetworks.Decipher(TEncoding::eCaesar, filename + ".Caesar");
  }
  else {
    c_nortelNetworks.Encipher(TEncoding::eRSA, filename + ".txt");
    c_nortelNetworks.Decipher(TEncoding::eRSA, filename + ".RSA");
  }
}

void PrintSubheader(std::string subtitle) {
  //print fancy subtitle
  std::cout << "####################################" << std::endl;
  std::cout << "####   " << subtitle << std::endl;
  std::cout << "####################################" << std::endl;
}

void CreateFullTest(std::string subtitle, std::string filename) {

  //is used to output all errors in red
  HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);

  //Tests the epos interface
  PrintSubheader(subtitle);
  std::cout << "Test epos... ";
  SetConsoleTextAttribute(hConsole, 4); //error-handling: set color red
  test_EPOS("../testFiles/"+ filename);
  SetConsoleTextAttribute(hConsole, 15);  //error-handling: set color white
  std::cout << "...completed!" << std::endl;

  //Tests the Caesar functions of the NN interface
  std::cout << "Test Nortel Networks Caesar... ";
  SetConsoleTextAttribute(hConsole, 4); //error-handling: set color red
  test_NN(TEncoding::eCaesar, "../testFiles/"+ filename + "-c");
  SetConsoleTextAttribute(hConsole, 15);  //error-handling: set color white
  std::cout << " completed!" << std::endl;

  //Tests the RSA functions of the NN interface
  std::cout << "Test Nortel Networks RSA... ";
  SetConsoleTextAttribute(hConsole, 4); //error-handling: set color red
  test_NN(TEncoding::eRSA, "../testFiles/" + filename + "-r");
  SetConsoleTextAttribute(hConsole, 15);  //error-handling: set color white
  std::cout << " completed!" << std::endl;
  std::cout << std::endl;
}
```