1 Organisatorisches

1.1 Team

- Reinhard Penn, s1110306019
- Bernhard Selymes, s1110306024

1.2 Aufteilung

- Reinhard Penn
  - Planung
  - Klassendiagramm
  - Implementierung der Klassen SingletonBase, ImageManagement, Circle, EmptyGraphicObjectFactory
  - Testen aller Klassen
- Bernhard Selymes
  - Planung
  - Klassendiagramm
  - Implementierung der Klassen Image, Rectangle, FilledGraphicObjectFactory, GraphicObject
  - Dokumentation

1.3 Zeitaufwand

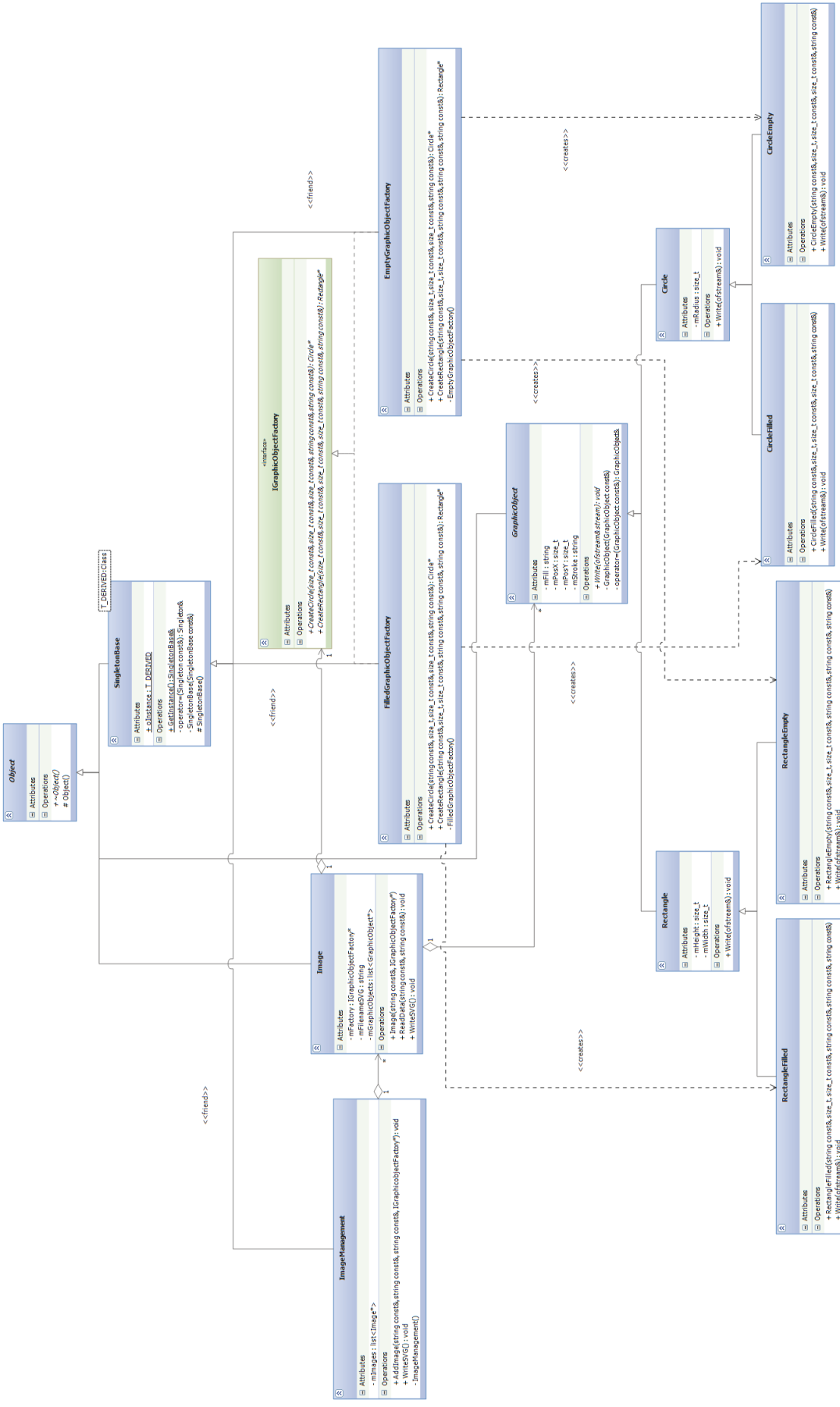
- geschätzte Mh: 20h
- tatsächlich: Reinhard (11h), Bernhard (11h)

2 Systemspezifikation

Es soll eine Software für die Erzeugung und Verwaltung von Bildern mit graphischen Objekten erstellt werden. Die Bilder sind in einer Bildverwaltung abgelegt. SVG-Dateien der Bilder können erzeugt werden. Diese können dann in einem Browser angezeigt werden.

3 Systementwurf

3.1 Klassendiagramm



3.2 Komponentenübersicht

- Klasse ”Object”:
  Basis aller Basisklassen.
- Klasse ”SingletonBase”:
  Template Basisklasse für Singletons.
- Klasse ”ImageManagement”:
  Verwaltet die Bilder.
- Klasse ”Image”:
  Beinhaltet die einzelnen graphischen Objekte. Kann in eine SVG-Datei geschrieben werden.
- Interface ”IGraphicObjectFactory”:
  Interface für die Fabrik die die Objekte erzeugt.
- Klasse ”FilledGraphicObjectFactory”:
  Erzeugt gefüllte graphische Objekte.
- Klasse ”EmptyGraphicObjectFactory”:
  Erzeugt leere graphische Objekte.
- Klasse ”GraphicObject”:
  Abstrakte Basisklasse für ein graphisches Objekt.
- Klasse ”Rectangle”:
  Abgeleitet von GraphicObject. Basisklasse für ein Viereck.
- Klassen ”RectangleFilled und RectangleEmpty”:
  Abgeleitet von Rectangle. Vierecke die entweder gefüllt oder leer sind.
- Klasse ”Circle”:
  Abgeleitet von GraphicObject. Basisklasse für einen Kreis.
- Klassen ”CircleFilled und CircleEmpty”:
  Abgeleitet von Circle. Kreise die entweder gefüllt oder leer sind.

4 Komponentenentwurf

4.1 Klasse ”Object”

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

4.2 Klasse ”SingletonBase”

Template Basisklasse für die Singletons. Hat einen static Member der auf die Klasse selbst zeigt. Zuweisungsoperator und Copyconstructor sind private. Constructor ist protected.

Methode ”GetInstance”:
Schnittstelle: Rückgabetyp: SingletonBase&.
Static Function die ein Objekt der Klasse erzeugt und einen Zeiger darauf zurückgibt.

4.3 Klasse ”ImageManagement”

Hat eine Liste in der die Bilder gespeichert sind. Ist ein Singleton, Constructor ist private.

Methode ”AddImage”:
Schnittstelle: Rückgabetyp: void.
Parameter: string const& filename1, string const& filename2, string const& SVGFileName, IGraphicObjectFactory* factory
Erzeugt ein neues Bild. Ruft dann die ReadData Funktion des Bildes auf und fügt es zur Liste mit den Bildern hinzu.

Methode ”WriteSVG”:
Schnittstelle:
Rückgabetyp: void.
Ruft für jedes Bild WriteSVG auf.

4.4 Klasse ”Image”

Beinhaltet die Fabrik mit der die Bilder erzeugt werden, den Namen unter der die SVG-Datei gespeichert werden soll und eine Liste mit den graphischen Objekten.

Constructor:
Schnittstelle: Parameter: std::string const& str, IGraphicObjectFactory* factory
Legt die Fabrik und den Dateinamen fest.

Methode ”ReadData”:
Schnittstelle:
Rückgabetyp: void.
Parameter: string const& filename1, string const& filename2
Liest die Daten von der Datei mit den Vierecken und der Datei mit den Kreisen ein, erzeugt die Objekte mit der jeweiligen Fabrik und fügt sie dann der Liste mit den Graphischen Objekten hinzu. Wenn fehlerhafte Daten (bei x,y,radius,width,height) in der Datei sind werden diese durch den stringstream in Zahlen umgewandelt. Fill und Stroke werden nicht überprüft. Wenn

in der Datei noch zusätzliche Elemente sind, werden diese ignoriert.

Methode ”WriteSVG”:
Schnittstelle:
Rückgabetyp: void.
Öffnet die Datei, schreibt den Header hinein, ruft dann die Write Funktionen von den Graphischen Objekten auf und schreibt zum Schluss den Abschluss in die Datei.

4.5 Interface ”IGraphicObjectFactory”

Ist die Schnittstelle für die Fabriken, die die gefüllten oder leeren Graphischen Objekte erzeugen.

Methode ”CreateRectangle”:
Schnittstelle:
Rückgabetyp: Rectangle*.
Parameter: size_t const& posX, size_t const& posY, size_t const& width, size_t const& height, string const& stroke, string const& fill
Pure virtual function.

Methode ”CreateCircle”:
Schnittstelle:
Rückgabetyp: Circle*.
Parameter: size_t const& posX, size_t const& posY, size_t const& radius, string const& stroke, string const& fill
Pure virtual function.

4.6 Klassen ”FilledGraphicObjectFactory” und ”EmptyGraphicObjectFactory”

Implementieren die Funktionen aus dem Interface. Es wird der Constructor für das entsprechende Graphische Objekt aufgerufen. Je nach dem Kreis/Rechteck gefüllt/leer. Beide sind Singletons.

4.7 Klasse ”GraphicObject”

Beinhaltet die allgemeinen Information eines graphischen Objekts: Füllung, x-Position, y-Position, Randfarbe. Copyconstructor und Zuweisungsoperator sind private damit das Objekt nicht zugewiesen oder kopiert werden kann.

4.8 Klasse ”Rectangle”

Hat zusätzlich die Attribute Höhe und Breite des Rechtecks.

Methode ”Write”:
Schnittstelle: Parameter: ofstream& stream.
Rückgabetyp: void.

Daten werden entsprechend der SVG Syntax in den Stream geschrieben.

4.9 Klassen ”RectangleFilled” und ”RectangleEmpty”

Haben Write Funktionen die noch die entsprechenden Daten zum Stream hinzufügen.

4.10 Klasse ”Circle”

Hat zusätzlich das Attribut Radius des Kreises.

Methode ”Write”:
Schnittstelle: Parameter: ofstream& stream.
Rückgabetyp: void.
Daten werden entsprechend der SVG Syntax in den Stream geschrieben.

4.11 Klassen ”CircleFilled” und ”CircleEmpty”

Haben Write Funktionen die noch die entsprechenden Daten zum Stream hinzufügen.

5 Dateibeschreibung

5.1 Datei mit Daten für ”Rectangle”

Dateiname: *Rect.data
Inhalt: Daten für die Vierecke.
EBNF:

```
file = {rectangle}.
rectangle = x y width height stroke [fill].
x = y = width = height = number.
number = digit {digit}.
digit = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9".
stroke = fill = word.
word = letter {letter}.
letter = "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".
```

5.2 Datei mit Daten für ”Circle”

Dateiname: *Circ.data
Inhalt: Daten für die Kreise.
EBNF:

```
file = {circle}.
circle = x y width height stroke [fill].
x = y = radius = number.
number = digit {digit}.
digit = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9".
word = letter {letter}.
letter =  "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".
```

6 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 : SingletonBase.h
// Author : Reinhard Penn, Bernhard Selymes
// Date : 18.11.2012
// Description : Definition of template class SingletonBase
////////////////////////////////////////////////////////////////////////////

#ifndef SINGLETONBASE_H
#define SINGLETONBASE_H

#include "Object.h"

//template singleton base class
template <typename T_DERIVED>
class SingletonBase {
public:
   static T_DERIVED& GetInstance()
   {
      static T_DERIVED oInstance;
      return oInstance;
   }
protected:
   SingletonBase(){};
private:
   SingletonBase(SingletonBase const& s){};
   SingletonBase& operator = (SingletonBase const& s){};
};

#endif
```

```cpp
////////////////////////////////////////////////////////////////////////////
// Workfile : ImageManagment.h
// Author : Reinhard Penn, Bernhard Selymes
// Date : 18.11.2012
// Description : Header for ImageManagment.cpp
////////////////////////////////////////////////////////////////////////////

#ifndef IMAGEMANAGMENT_H
#define IMAGEMANAGMENT_H

#include <string>
#include <list>
#include "SingletonBase.h"
#include "Image.h"

class ImageManagment
   : public SingletonBase<ImageManagment>
{
   friend class SingletonBase<ImageManagment>;
public:
   //virtual Destructor
   virtual ~ImageManagment();

   void AddImage(std::string const& filename1, std::string const& filename2
      ,
      std::string const& SVGFileName, IGraphicObjectFactory* factory);

   void WriteSVG();

private:
   std::list<Image*> mImageList;

   //Default CTor
   ImageManagment();
};

#endif
```

```cpp
////////////////////////////////////////////////////////////////////////
// Workfile : ImageManagment.cpp
// Author : Reinhard Penn, Bernhard Selymes
// Date : 18.11.2012
// Description : Implementation of class ImageManagment
////////////////////////////////////////////////////////////////////////

#include <algorithm>
#include <iterator>
#include "ImageManagment.h"
#include "Image.h"

#include <iostream>

ImageManagment::ImageManagment() {}

ImageManagment::~ImageManagment()
{
   std::for_each(mImageList.begin(),mImageList.end(),[](Image* img)
   {
      delete img;
   });
}

void ImageManagment::AddImage(std::string const& filename1, std::string
    const& filename2,
   std::string const& SVGFileName, IGraphicObjectFactory* factory)
{
   Image* img = new Image(SVGFileName, factory);

   img->ReadData(filename1,filename2);
   mImageList.push_back(img);
}

void ImageManagment::WriteSVG()
{
   std::for_each(mImageList.begin(),mImageList.end(),[](Image* img)
   {
      img->WriteSVG();
   });
}
```

```cpp
////////////////////////////////////////////////////////////////////////
// Workfile : Image.h
// Author : Reinhard Penn, Bernhard Selymes
// Date : 18.11.2012
// Description : Header for Image.cpp
////////////////////////////////////////////////////////////////////////

#ifndef IMAGE_H
#define IMAGE_H

#include <string>
#include <list>
#include "Object.h"
#include "IGraphicObjectFactory.h"
#include "GraphicObject.h"

typedef std::list<GraphicObject*> TGraphicObjects;
typedef TGraphicObjects::const_iterator TGraphicObjectsItor;

std::string const version = "<?xml version=\"1.0\"?>";
std::string const link = "<svg xmlns=\"http://www.w3.org/2000/svg\">";
std::string const endSVG = "</svg>";

class Image
   : public Object
{
public:
   //CTor
   Image(std::string const& str, IGraphicObjectFactory* factory);

   //virtual Destructor
   virtual ~Image();

   void WriteSVG();
   void ReadData(std::string const& filename1,std::string const& filename2)
      ;
private:
   std::string mFileNameSVG;
   IGraphicObjectFactory* mFactory;
   TGraphicObjects mGraphicObjects;
};

#endif
```

```cpp
////////////////////////////////////////////////////////////////////////
// Workfile : Image.cpp
// Author : Reinhard Penn, Bernhard Selymes
// Date : 18.11.2012
// Description : Implementation of class Image
////////////////////////////////////////////////////////////////////////

#include <fstream>
#include <iostream>
#include <sstream>
#include <algorithm>
#include <iterator>
#include "Image.h"

Image::Image(std::string const& str, IGraphicObjectFactory* factory)
{
   mFileNameSVG = str;
   mFactory = factory;
}

Image::~Image()
{
   std::for_each(mGraphicObjects.begin(),mGraphicObjects.end(),[](
      GraphicObject* obj)
   {
      delete obj;
   });
}

void Image::WriteSVG()
{
   try {
      std::ofstream file;
      file.open(mFileNameSVG);

      if (!file.is_open())
      {
         std::string ex("File couldn't be opened");
         throw(ex);
      }

      file << version << std::endl << link << std::endl;

      /*for(TGraphicObjectsItor itor = mGraphicObjects.begin(); itor !=
         mGraphicObjects.end(); ++itor)
      {
      (*itor)->Write(file);
      ++itor;
      }*/

      std::for_each(mGraphicObjects.begin(),mGraphicObjects.end(),[&file](
         GraphicObject* obj)
      {
         obj->Write(file);
      });

      file << endSVG;
      file.close();
   }
   catch(std::string const& ex)
```

```
58     {
59        std::cerr << "Image.cpp::WriteSVG: " << ex << std::endl;
60     }
61     catch(...)
62     {
63        std::cerr << "Image.cpp::WriteSVG: Unknown Exception occured" << std
              ::endl;
64     }
65  }
66
67  void Image::ReadData(std::string const& filename1,std::string const&
       filename2)
68  {
69     try
70     {
71        std::ifstream file1(filename1);  //rectangle
72        std::string buffer;
73
74        //help variables
75        size_t pos = 0;   //position of " " in string
76        size_t posX = 0;
77        size_t posY = 0;
78        size_t width = 0;
79        size_t height = 0;
80        size_t radius = 0;
81        std::string stroke = "";
82        std::string fill = "";
83
84        if (!file1.is_open())
85        {
86           std::string ex("File with Rectangle Data couldn't be opened");
87           throw(ex);
88        }
89
90        //rectangle
91        while(!file1.eof())
92        {
93           getline(file1,buffer);
94           // not an empty string
95           if (buffer != "")
96           {
97              //posX
98              pos = buffer.find_first_of(' ');
99              std::stringstream (buffer.substr(0,pos)) >> posX;
100             buffer.erase(0,pos+1);
101
102             //posY
103             pos = buffer.find_first_of(' ');
104             std::stringstream (buffer.substr(0,pos)) >> posY;
105             buffer.erase(0,pos+1);
106
107             //width
108             pos = buffer.find_first_of(' ');
109             std::stringstream (buffer.substr(0,pos)) >> width;
110             buffer.erase(0,pos+1);
111
112             //height
113             pos = buffer.find_first_of(' ');
114             std::stringstream (buffer.substr(0,pos)) >> height;
115             buffer.erase(0,pos+1);
```

```
116
117             //stroke
118             pos = buffer.find_first_of(' ');
119             stroke = buffer.substr(0,pos);
120             buffer.erase(0,pos+1);
121
122             //fill exists or doesnt exist
123             if (buffer == "")
124             {
125                fill = stroke;
126             }
127             else
128             {
129                pos = buffer.find_first_of(' ');
130                fill = buffer.substr(0,pos);
131             }
132             buffer.erase();
133             mGraphicObjects.push_back(mFactory->CreateRectangle(posX,posY,
                   width,height,stroke,fill));
134          }
135       }
136       file1.close();
137
138       //circle
139       std::ifstream file2(filename2);
140       if (!file2.is_open())
141       {
142          std::string ex("File with Circle Data couldn't be opened");
143          throw(ex);
144       }
145
146       //circle
147       while(!file2.eof())
148       {
149          getline(file2,buffer);
150          // not an empty string
151          if (buffer != "")
152          {
153             //posX
154             pos = buffer.find_first_of(' ');
155             std::stringstream (buffer.substr(0,pos)) >> posX;
156             buffer.erase(0,pos+1);
157
158             //posY
159             pos = buffer.find_first_of(' ');
160             std::stringstream (buffer.substr(0,pos)) >> posY;
161             buffer.erase(0,pos+1);
162
163             //radius
164             pos = buffer.find_first_of(' ');
165             std::stringstream (buffer.substr(0,pos)) >> radius;
166             buffer.erase(0,pos+1);
167
168             //stroke
169             pos = buffer.find_first_of(' ');
170             stroke = buffer.substr(0,pos);
171             buffer.erase(0,pos+1);
172
173             //fill exists or doesnt exist
174             if (buffer == "")
```

```
175             {
176                fill = stroke; //if doesnt exist it gets color of stroke
177             }
178             else
179             {
180                pos = buffer.find_first_of(' ');
181                fill = buffer.substr(0,pos);
182             }
183             buffer.erase();
184             mGraphicObjects.push_back(mFactory->CreateCircle(posX,posY,
                   radius,stroke,fill));
185          }
186       }
187       file2.close();
188    }
189    catch(std::string const& ex)
190    {
191       std::cerr << "Image.cpp::ReadData: " << ex << std::endl;
192    }
193    catch(...)
194    {
195       std::cerr << "Image.cpp::ReadData: Unknown Exception occured" << std
          ::endl;
196    }
197 }
```

```
1  ////////////////////////////////////////////////////////////////////////
2  // Workfile : IGraphicObjectFactory.h
3  // Author : Reinhard Penn, Bernhard Selymes
4  // Date : 18.11.2012
5  // Description : GraphicObjectFactory interface
6  ////////////////////////////////////////////////////////////////////////
7
8  #ifndef IGRAPHICOBJECTFACTORY_H
9  #define IGRAPHICOBJECTFACTORY_H
10
11 #include "Circle.h"
12 #include "Rectangle.h"
13
14 class IGraphicObjectFactory
15 {
16 public:
17    virtual Rectangle* CreateRectangle(size_t const& posX, size_t const&
          posY,
18       size_t const& width, size_t const& height, std::string const& stroke,
              std::string const& fill) = 0;
19    virtual Circle* CreateCircle(size_t const& posX, size_t const& posY,
20       size_t const& radius, std::string const& stroke, std::string const&
             fill) = 0;
21 };
22
23 #endif
```

```cpp
////////////////////////////////////////////////////////////////////////
// Workfile : FilledGraphicObjectFactory.h
// Author : Reinhard Penn, Bernhard Selymes
// Date : 18.11.2012
// Description : Header for FilledGraphicObjectFactory.cpp
////////////////////////////////////////////////////////////////////////

#ifndef FILLEDGRAPHICOBJECTFACTORY_H
#define FILLEDGRAPHICOBJECTFACTORY_H

#include <string>
#include "Circle.h"
#include "Rectangle.h"
#include "SingletonBase.h"
#include "IGraphicObjectFactory.h"

class FilledGraphicObjectFactory :
   public SingletonBase<FilledGraphicObjectFactory>,
   public IGraphicObjectFactory
{
   friend class SingletonBase<FilledGraphicObjectFactory>;
public:
   Rectangle* CreateRectangle(size_t const& posX, size_t const& posY,
      size_t const& width,
      size_t const& height, std::string const& stroke, std::string const&
         fill);
   Circle* CreateCircle(size_t const& posX, size_t const& posY, size_t
      const& radius,
      std::string const& stroke, std::string const& fill);
private:
   FilledGraphicObjectFactory();
};

#endif
```

```cpp
////////////////////////////////////////////////////////////////////////
// Workfile : FilledGraphicObjectFactory.cpp
// Author : Reinhard Penn, Bernhard Selymes
// Date : 18.11.2012
// Description : Implementation of class FilledGraphicObjectFactory
////////////////////////////////////////////////////////////////////////

#include "FilledGraphicObjectFactory.h"
#include "RectangleFilled.h"
#include "CircleFilled.h"
#include "Image.h"

FilledGraphicObjectFactory::FilledGraphicObjectFactory() {}

Rectangle* FilledGraphicObjectFactory::CreateRectangle(size_t const& posX,
    size_t const& posY, size_t const& width,
    size_t const& height, std::string const& stroke, std::string const& fill
        )
{
    return new RectangleFilled(posX, posY, width, height, stroke, fill);
}

Circle* FilledGraphicObjectFactory::CreateCircle(size_t const& posX, size_t
     const& posY, size_t const& radius,
    std::string const& stroke, std::string const& fill)
{
    return new CircleFilled(posX, posY, radius, stroke, fill);
}
```

```
1  ////////////////////////////////////////////////////////////////////////
2  // Workfile : EmptyGraphicObjectFactory.h
3  // Author : Reinhard Penn, Bernhard Selymes
4  // Date : 18.11.2012
5  // Description : Header for EmptyGraphicObjectFactory.cpp
6  ////////////////////////////////////////////////////////////////////////
7
8  #ifndef EMPTYGRAPHICOBJECTFACTORY_H
9  #define EMPTYGRAPHICOBJECTFACTORY_H
10
11 #include <string>
12 #include "Circle.h"
13 #include "Rectangle.h"
14 #include "SingletonBase.h"
15 #include "IGraphicObjectFactory.h"
16
17 class EmptyGraphicObjectFactory :
18    public SingletonBase<EmptyGraphicObjectFactory>,
19    public IGraphicObjectFactory
20 {
21    friend class SingletonBase<EmptyGraphicObjectFactory>;
22 public:
23    Rectangle* CreateRectangle(size_t const& posX, size_t const& posY,
          size_t const& width,
24       size_t const& height, std::string const& stroke, std::string const&
             fill);
25    Circle* CreateCircle(size_t const& posX, size_t const& posY,
26       size_t const& radius, std::string const& stroke, std::string const&
             fill);
27 private:
28    EmptyGraphicObjectFactory();
29 };
30
31 #endif
```

```cpp
////////////////////////////////////////////////////////////////////////////
// Workfile : EmptyGraphicObjectFactory.cpp
// Author : Reinhard Penn, Bernhard Selymes
// Date : 18.11.2012
// Description : Implementation of class EmptyGraphicObjectFactory
////////////////////////////////////////////////////////////////////////////

#include "EmptyGraphicObjectFactory.h"
#include "RectangleEmpty.h"
#include "CircleEmpty.h"
#include "Image.h"

EmptyGraphicObjectFactory::EmptyGraphicObjectFactory() {}

Rectangle* EmptyGraphicObjectFactory::CreateRectangle(size_t const& posX, size_t const& posY,
   size_t const& width, size_t const& height, std::string const& stroke, std::string const& fill)
{
   return new RectangleEmpty(posX, posY, width, height, stroke);
}

Circle* EmptyGraphicObjectFactory::CreateCircle(size_t const& posX, size_t const& posY,
   size_t const& radius, std::string const& stroke, std::string const& fill)
{
   return new CircleEmpty(posX, posY, radius, stroke);
}
```

```cpp
////////////////////////////////////////////////////////////////////////////
// Workfile : GraphicObject.h
// Author : Reinhard Penn, Bernhard Selymes
// Date : 18.11.2012
// Description : Header for GraphicObject.cpp
////////////////////////////////////////////////////////////////////////////

#ifndef GRAPHICOBJECT_H
#define GRAPHICOBJECT_H

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

std::string const begin = "<";
std::string const stroke = "stroke=";
std::string const fill = "fill=";
std::string const end = "/>";
std::string const space = " ";
std::string const qM = "\"";  //Quotation Mark
std::string const empty("none");

class GraphicObject
   : public Object
{
public:
   //Default CTor
   GraphicObject();

   virtual void Write(std::ofstream& stream) = 0;
protected:
   std::string mFill;
   std::string mStroke;
   size_t mPosX;
   size_t mPosY;

private:
   GraphicObject(GraphicObject const& s);
   GraphicObject& operator = (GraphicObject const& s);
};

#endif
```

```cpp
////////////////////////////////////////////////////////////////////////////
// Workfile : GraphicObject.cpp
// Author : Reinhard Penn, Bernhard Selymes
// Date : 18.11.2012
// Description : Implementation of class GraphicObject
////////////////////////////////////////////////////////////////////////////

#include "GraphicObject.h"

GraphicObject::GraphicObject()
   : mPosX(0), mPosY(0), mStroke(""), mFill("")
{}
```

```cpp
////////////////////////////////////////////////////////////////////////
// Workfile : Rectangle.h
// Author : Reinhard Penn, Bernhard Selymes
// Date : 18.11.2012
// Description : Header for Rectangle.cpp
////////////////////////////////////////////////////////////////////////

#ifndef RECTANGLE_H
#define RECTANGLE_H

#include <string>
#include <fstream>
#include "GraphicObject.h"

std::string const rect = "rect";
std::string const x = "x=";
std::string const y = "y=";
std::string const width = "width=";
std::string const height = "height=";

class Rectangle
   : public GraphicObject
{
public:
   //Default CTor
   Rectangle();

   virtual void Write(std::ofstream& stream);
protected:
   size_t mHeight;
   size_t mWidth;
};

#endif   //RECTANGLE_H
```

```cpp
////////////////////////////////////////////////////////////////////////////
// Workfile : Rectangle.cpp
// Author : Reinhard Penn, Bernhard Selymes
// Date : 18.11.2012
// Description : Implementation of class Rectangle
////////////////////////////////////////////////////////////////////////////

#include <fstream>
#include <iostream>
#include "Rectangle.h"

Rectangle::Rectangle() : mHeight(0), mWidth(0)
{}

void Rectangle::Write(std::ofstream& stream)
{
   try {
      if (!stream.is_open())
      {
         std::string ex("Stream couldn't be opened");
         throw(ex);
      }

      stream << begin << rect << space
            << x << qM << mPosX << qM << space
            << y << qM << mPosY << qM << space
            << width << qM << mWidth << qM << space
            << height << qM << mHeight << qM << space
            << stroke << qM << mStroke << qM;
   }
   catch(std::string const& ex)
   {
      std::cerr << "Rectangle.cpp::Write: " << ex << std::endl;
   }
   catch(...)
   {
      std::cerr << "Rectangle.cpp::Write: Unknown Exception occured" << std
         ::endl;
   }
}
```

```
1  ////////////////////////////////////////////////////////////////////////
2  // Workfile : RectangleFilled.h
3  // Author : Reinhard Penn, Bernhard Selymes
4  // Date : 18.11.2012
5  // Description : Header for RectangleFilled.cpp
6  ////////////////////////////////////////////////////////////////////////
7
8  #ifndef RECTANGLEFILLED_H
9  #define RECTANGLEFILLED_H
10
11 #include <string>
12 #include <fstream>
13 #include "Rectangle.h"
14
15 class RectangleFilled
16    : public Rectangle
17 {
18 public:
19    //CTor
20    RectangleFilled(size_t const& posX, size_t const& posY, size_t const&
       height,
21       size_t const& width, std::string const& stroke, std::string const&
          fill);
22
23    void Write(std::ofstream& stream);
24 };
25
26 #endif
```

```cpp
////////////////////////////////////////////////////////////////////////
// Workfile : RectangleFilled.cpp
// Author : Reinhard Penn, Bernhard Selymes
// Date : 18.11.2012
// Description : Implementation of class RectangleFilled
////////////////////////////////////////////////////////////////////////

#include <fstream>
#include <iostream>
#include "RectangleFilled.h"

RectangleFilled::RectangleFilled(size_t const& posX, size_t const& posY,
    size_t const& width,
    size_t const& height, std::string const& stroke, std::string const& fill
        )
{
   mPosX = posX;
   mPosY = posY;
   mWidth = width;
   mHeight = height;
   mStroke = stroke;
   mFill = fill;
}

void RectangleFilled::Write(std::ofstream& stream)
{
   try {
      Rectangle::Write(stream);
      if (!stream.is_open())
      {
         std::string ex("Stream couldn't be opened");
         throw(ex);
      }
      stream << space << fill << qM << mFill << qM << end << std::endl;
   }
   catch(std::string const& ex)
   {
      std::cerr << "RectangleFilled.cpp::Write: " << ex << std::endl;
   }
   catch(...)
   {
      std::cerr << "RectangleFilled.cpp::Write: Unknown Exception occured"
         << std::endl;
   }
}
```

```
1  ////////////////////////////////////////////////////////////////////////
2  // Workfile : RectangleEmpty.h
3  // Author : Reinhard Penn, Bernhard Selymes
4  // Date : 18.11.2012
5  // Description : Header for RectangleEmpty.cpp
6  ////////////////////////////////////////////////////////////////////////
7
8  #ifndef RECTANGLEEMPTY_H
9  #define RECTANGLEEMPTY_H
10
11 #include <string>
12 #include <fstream>
13 #include "Rectangle.h"
14
15 class RectangleEmpty
16    : public Rectangle
17 {
18 public:
19    //CTor
20    RectangleEmpty(size_t const& posX, size_t const& posY, size_t const&
       height,
21       size_t const& width, std::string const& stroke);
22
23    virtual void Write(std::ofstream& stream);
24 };
25
26 #endif
```

```cpp
////////////////////////////////////////////////////////////////////////
// Workfile : RectangleEmpty.cpp
// Author : Reinhard Penn, Bernhard Selymes
// Date : 18.11.2012
// Description : Implementation of class RectangleEmpty
////////////////////////////////////////////////////////////////////////

#include <fstream>
#include <iostream>
#include "RectangleEmpty.h"

RectangleEmpty::RectangleEmpty(size_t const& posX, size_t const& posY,
    size_t const& width,
    size_t const& height, std::string const& stroke)
{
   mPosX = posX;
   mPosY = posY;
   mWidth = width;
   mHeight = height;
   mStroke = stroke;
   mFill = empty;
}

void RectangleEmpty::Write(std::ofstream& stream)
{
   try {
      Rectangle::Write(stream);
      if (!stream.is_open())
      {
         std::string ex("Stream couldn't be opened");
         throw(ex);
      }
      stream << space << fill << qM << mFill << qM << end << std::endl;
   }
   catch(std::string const& ex)
   {
      std::cerr << "RectangleEmpty.cpp::Write: " << ex << std::endl;
   }
   catch(...)
   {
      std::cerr << "RectangleEmpty.cpp::Write: Unknown Exception occured"
         << std::endl;
   }
}
```

```cpp
////////////////////////////////////////////////////////////////////////////
// Workfile : Circle.h
// Author : Reinhard Penn, Bernhard Selymes
// Date : 18.11.2012
// Description : Header for Circle.cpp
////////////////////////////////////////////////////////////////////////////

#ifndef CIRCLE_H
#define CIRCLE_H

#include <string>
#include <fstream>
#include "GraphicObject.h"

std::string const circle = "circle";
std::string const cx = "cx=";
std::string const cy = "cy=";
std::string const radius = "r=";

class Circle
   : public GraphicObject
{
public:
   //Default CTor
   Circle();

   virtual void Write(std::ofstream& stream);
protected:
   size_t mRadius;
};

#endif
```

```cpp
////////////////////////////////////////////////////////////////////////////
// Workfile : Circle.cpp
// Author : Reinhard Penn, Bernhard Selymes
// Date : 18.11.2012
// Description : Implementation of class Circle
////////////////////////////////////////////////////////////////////////////

#include <fstream>
#include <iostream>
#include "Circle.h"

Circle::Circle() : mRadius(0)
{}

void Circle::Write(std::ofstream& stream)
{
   try {
      if (!stream.is_open())
      {
         std::string ex("Stream couldn't be opened");
         throw(ex);
      }

      stream << begin << circle << space
            << cx << qM << mPosX << qM << space
            << cy << qM << mPosY << qM << space
            << radius << qM << mRadius << qM << space
            << stroke << qM << mStroke << qM;
   }
   catch(std::string const& ex)
   {
      std::cerr << "Circle.cpp::Write: " << ex << std::endl;
   }
   catch(...)
   {
      std::cerr << "Circle.cpp::Write: Unknown Exception occured" << std::
         endl;
   }
}
```

```
1  ////////////////////////////////////////////////////////////////////////
2  // Workfile : CircleFilled.h
3  // Author : Reinhard Penn, Bernhard Selymes
4  // Date : 18.11.2012
5  // Description : Header for CircleFilled.cpp
6  ////////////////////////////////////////////////////////////////////////
7
8  #ifndef CircleFilled_H
9  #define CircleFilled_H
10
11 #include <string>
12 #include <fstream>
13 #include "Circle.h"
14
15 class CircleFilled
16    : public Circle
17 {
18 public:
19    //CTor
20    CircleFilled(size_t const& posX, size_t const& posY, size_t const&
          radius,
21       std::string const& stroke, std::string const& fill);
22
23    virtual void Write(std::ofstream& stream);
24 };
25
26 #endif
```

```cpp
////////////////////////////////////////////////////////////////////////////
// Workfile : CircleFilled.cpp
// Author : Reinhard Penn, Bernhard Selymes
// Date : 18.11.2012
// Description : Implementation of class CircleFilled
////////////////////////////////////////////////////////////////////////////

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

CircleFilled::CircleFilled(size_t const& posX, size_t const& posY, size_t const& radius, std::string const& stroke, std::string const& fill)
{
   mPosX = posX;
   mPosY = posY;
   mRadius = radius;
   mStroke = stroke;
   mFill = fill;
}

void CircleFilled::Write(std::ofstream& stream)
{
   try {
      Circle::Write(stream);
      if (!stream.is_open())
      {
         std::string ex("Stream couldn't be opened");
         throw(ex);
      }
      stream << space << fill << qM << mFill << qM << end << std::endl;
   }
   catch(std::string const& ex)
   {
      std::cerr << "CircleFilled.cpp::Write: " << ex << std::endl;
   }
   catch(...)
   {
      std::cerr << "CircleFilled.cpp::Write: Unknown Exception occured" << std::endl;
   }
}
```

```
1  ////////////////////////////////////////////////////////////////////////
2  // Workfile : CircleEmpty.h
3  // Author : Reinhard Penn, Bernhard Selymes
4  // Date : 18.11.2012
5  // Description : Header for CircleEmpty.cpp
6  ////////////////////////////////////////////////////////////////////////
7
8  #ifndef CIRCLEEMPTY_H
9  #define CIRCLEEMPTY_H
10
11 #include <string>
12 #include <fstream>
13 #include "Circle.h"
14
15 class CircleEmpty
16    : public Circle
17 {
18 public:
19    //CTor
20    CircleEmpty(size_t const& posX, size_t const& posY, size_t const& radius
       ,
21       std::string const& stroke);
22
23    virtual void Write(std::ofstream& stream);
24 };
25
26 #endif
```

```cpp
////////////////////////////////////////////////////////////////////////////
// Workfile : CircleEmpty.cpp
// Author : Reinhard Penn, Bernhard Selymes
// Date : 18.11.2012
// Description : Implementation of class CircleEmpty
////////////////////////////////////////////////////////////////////////////

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

CircleEmpty::CircleEmpty(size_t const& posX, size_t const& posY, size_t
    const& radius, std::string const& stroke)
{
   mPosX = posX;
   mPosY = posY;
   mRadius = radius;
   mStroke = stroke;
   mFill = empty;
}

void CircleEmpty::Write(std::ofstream& stream)
{
   try {
      Circle::Write(stream);
      if (!stream.is_open())
      {
         std::string ex("Stream couldn't be opened");
         throw(ex);
      }
      stream << space << fill << qM << mFill << qM << end << std::endl;
   }
   catch(std::string const& ex)
   {
      std::cerr << "CircleEmpty.cpp::Write: " << ex << std::endl;
   }
   catch(...)
   {
      std::cerr << "CircleEmpty.cpp::Write: Unknown Exception occured" <<
         std::endl;
   }
}
```

```cpp
////////////////////////////////////////////////////////////////////////
// Workfile : main.cpp
// Author : Reinhard Penn, Bernhard Selymes
// Date : 18.11.2012
// Description : Testdriver for whole program
////////////////////////////////////////////////////////////////////////

#include <iostream>
#include "SingletonBase.h"
#include "ImageManagment.h"
#include "IGraphicObjectFactory.h"
#include "FilledGraphicObjectFactory.h"
#include "EmptyGraphicObjectFactory.h"

using namespace std;


void testfunction(string filename1, string filename2, string SVGFilename,
    string Header)
{
   ImageManagment& imgManager = ImageManagment::GetInstance();
   FilledGraphicObjectFactory& filledGraphicObjectFactory =
      FilledGraphicObjectFactory::GetInstance();
   EmptyGraphicObjectFactory& emptyGraphicObjectFactory =
      EmptyGraphicObjectFactory::GetInstance();

   cout << Header << endl;

   cout << "AddImage(Filled) ... ";
   imgManager.AddImage(filename1, filename2, "Filled_"+SVGFilename, &
      filledGraphicObjectFactory);
   cout << "done" << endl;

   cout << "AddImage(Empty) ... ";
   imgManager.AddImage(filename1, filename2, "Empty_"+SVGFilename, &
      emptyGraphicObjectFactory);
   cout << "done" << endl;

   cout << "WriteSVG ... ";
   imgManager.WriteSVG();
   cout << "done" << endl;

   cout << endl << endl;
}

int main()
{
   testfunction("NULL","NULL","Testcase0_NotHere.svg",
      "Testcase0: Not existing files");

   testfunction("EmptyRect.data","EmptyCircle.data","Testcase1_EmptyFile.
      svg",
      "Testcase1: Empty files");

   testfunction("Rect.data","Circle.data","Testcase2_FilledFile.svg",
      "Testcase2: Files with correct data");

   testfunction("Rect_Wrong.data","Circle_Wrong.data","
      Testcase3_Filled_File_Wrong_Data.svg",
      "Testcase3: Files with wrong data");
```

```
54
55      return 0;
56  }
```

7 Testausgaben

```
Testcase0: Not existing files
AddImage(Filled) ... Image.cpp::ReadData:
File with Rectangle Data couldn't be opened
done
AddImage(Empty) ... Image.cpp::ReadData:
File with Rectangle Data couldn't be opened
done
WriteSVG ... done

Testcase1: Empty files
AddImage(Filled) ... done
AddImage(Empty) ... done
WriteSVG ... done

Testcase2: Files with correct data
AddImage(Filled) ... done
AddImage(Empty) ... done
WriteSVG ... done

Testcase3: Files with wrong data
AddImage(Filled) ... done
AddImage(Empty) ... done
WriteSVG ... done
```