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 (9h)

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.

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 Source Code

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

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

```
1  ////////////////////////////////////////////////////////////////////////
2  // Workfile : SingletonBase.h
3  // Author : Reinhard Penn, Bernhard Selymes
4  // Date : 18.11.2012
5  // Description : Definition of template class SingletonBase
6  ////////////////////////////////////////////////////////////////////////
7
8  #ifndef SINGLETONBASE_H
9  #define SINGLETONBASE_H
10
11 #include "Object.h"
12
13 //template singleton base class
14 template <typename T_DERIVED>
15 class SingletonBase {
16 public:
17    static T_DERIVED& GetInstance()
18    {
19       static T_DERIVED oInstance;
20       return oInstance;
21    }
22 protected:
23    SingletonBase(){}
24 private:
25    SingletonBase(SingletonBase const& s){};
26    SingletonBase& operator = (SingletonBase const& s){};
27 };
28
29 #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::cout << *(mImageList.begin()) << std::endl;
   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
#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(mFileNameSVG, std::fstream::out);
      if (!file.is_open())
      {
         std::string ex("File couldn't be opened");
         throw(ex);
      }

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

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

      file << endSVG;
      file.close();
   }
   catch(std::string const& ex)
   {
      std::cerr << "Image.cpp::WriteSVG: " << ex << std::endl;
   }
   catch(...)
   {
      std::cerr << "Image.cpp::WriteSVG: Unknown Exception occured" << std
         ::endl;
   }
}

void Image::ReadData(std::string const& filename1,std::string const&
   filename2)
{
   try
   {
      std::ifstream file1(filename1);  //rectangle
```

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

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

```
174     }
175     catch(...)
176     {
177        std::cerr << "Image.cpp::ReadData: Unknown Exception occured" << std
               ::endl;
178     }
179 }
```

```
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, size_t const& width, size_t const& height, std::string const&
          stroke, std::string const& fill) = 0;
18    virtual Circle* CreateCircle(size_t const& posX, size_t const& posY,
          size_t const& radius, std::string const& stroke, std::string const&
          fill) = 0;
19 };
20
21 #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
```

```
1  #include "FilledGraphicObjectFactory.h"
2  #include "RectangleFilled.h"
3  #include "CircleFilled.h"
4  #include "Image.h"
5
6  FilledGraphicObjectFactory::FilledGraphicObjectFactory() {}
7
8  Rectangle* FilledGraphicObjectFactory::CreateRectangle(size_t const& posX,
       size_t const& posY, size_t const& width,
9      size_t const& height, std::string const& stroke, std::string const& fill
           )
10 {
11     return new RectangleFilled(posX, posY, width, height, stroke, fill);
12 }
13
14 Circle* FilledGraphicObjectFactory::CreateCircle(size_t const& posX, size_t
        const& posY, size_t const& radius,
15     std::string const& stroke, std::string const& fill)
16 {
17     return new CircleFilled(posX, posY, radius, stroke, fill);
18 }
```

```
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);
25    Circle* CreateCircle(size_t const& posX, size_t const& posY,
26        size_t const& radius, std::string const& stroke);
27 private:
28    EmptyGraphicObjectFactory();
29 };
30
31 #endif
```

```
1  #include "EmptyGraphicObjectFactory.h"
2  #include "RectangleEmpty.h"
3  #include "CircleEmpty.h"
4  #include "Image.h"
5
6  EmptyGraphicObjectFactory::EmptyGraphicObjectFactory() {}
7
8  Rectangle* EmptyGraphicObjectFactory::CreateRectangle(size_t const& posX,
       size_t const& posY,
9      size_t const& width, size_t const& height, std::string const& stroke)
10 {
11     return new RectangleEmpty(posX, posY, width, height, stroke);
12 }
13
14 Circle* EmptyGraphicObjectFactory::CreateCircle(size_t const& posX, size_t
       const& posY,
15     size_t const& radius, std::string const& stroke)
16 {
17     return new CircleEmpty(posX, posY, radius, stroke);
18 }
```

```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
```

```
1  #include "GraphicObject.h"
2
3  GraphicObject::GraphicObject()
4      : mPosX(0), mPosY(0), mStroke(""), mFill("")
5  {}
```

```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
```

```
1  #include <fstream>
2  #include <iostream>
3  #include "Rectangle.h"
4
5  Rectangle::Rectangle() : mHeight(0), mWidth(0)
6  {}
7
8  void Rectangle::Write(std::ofstream& stream)
9  {
10    try {
11       if (!stream.is_open())
12       {
13          std::string ex("Stream couldn't be opened");
14          throw(ex);
15       }
16
17       stream << begin << rect << space
18              << x << qM << mPosX << qM << space
19              << y << qM << mPosY << qM << space
20              << width << qM << mWidth << qM << space
21              << height << qM << mHeight << qM << space
22              << stroke << qM << mStroke << qM;
23    }
24    catch(std::string const& ex)
25    {
26       std::cerr << "Rectangle.cpp::Write: " << ex << std::endl;
27    }
28    catch(...)
29    {
30       std::cerr << "Rectangle.cpp::Write: Unknown Exception occured" << std
             ::endl;
31    }
32 }
```

```
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
#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;
        stream.close();
    }
    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
#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;
        stream.close();
    }
    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
#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 << "Rectangle.cpp::Write: " << ex << std::endl;
   }
   catch(...)
   {
      std::cerr << "Rectangle.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
#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;
      stream.close();
   }
   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
#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;
      stream.close();
   }
   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>

using namespace std;

#include "SingletonBase.h"
#include "ImageManagment.h"
#include "FilledGraphicObjectFactory.h"
//#include "EmptyGraphicObjectFactory.h"
#include "Image.h"

int main()
{
   std::string filename1 = "Rect.data";
   std::string filename2 = "Circ.data";
   std::string filenameOutput = "test0.svg";

   ImageManagment& imgManager = ImageManagment::GetInstance();
   FilledGraphicObjectFactory& filledGraphicObjectFactory =
      FilledGraphicObjectFactory::GetInstance();
   imgManager.AddImage(filename1, filename2, filenameOutput, &
      filledGraphicObjectFactory);
   imgManager.WriteSVG();

   return 0;
}
```

6 Testausgaben