

☐ Gr. 1, J. Karder, MSc.

Name \_\_\_\_\_ Effort in h \_\_\_\_\_

☐ Gr. 2, P. Fleck, MSc.

Points \_\_\_\_\_ Lecturer \_\_\_\_\_

**1. One, to generate them all****(6 Points)**

This time it is your task to implement a mechanism for generating source code for classes. This time, the specification of these classes is given in XML files; for example, the definition of the classes *Person* and *Student* could be given in the following way:

```
<?xml version="1.0" encoding="utf-8" ?>
<model>
  <class name="Person" namespace="GenerativeProgramming">
    <property name="Name" type="string"/>
    <property name="Age" type="int"/>
  </class>
  <class name="Student" namespace="GenerativeProgramming" base="Person">
    <property name="Id" type="string"/>
  </class>
</model>
```

Implement a **T4 template** for generating C# code using the given XML files. The definition of base classes should be optional; e.g., in this example the base class for the class *Student* is *Person*.

In addition to the defined properties the generated classes should have a *ToString()* function. For example, for an instance of the class *Student* the *ToString()* function should return:

```
Name: Ben
Age: 23
Id: 2045402
```

**2. Clone 'Em All!****(6 Points)**

In many frameworks, for example also in HeuristicLab, cloning mechanisms are used for creating deep copies of arbitrary objects in an automated way. Your task is now to implement a **T4 template** which generates a function *IDeepCloneable Clone(Cloner cloner)* as well as a copy constructor for given classes.

Of course, fields of objects that are to be cloned are also cloned:

- Elements that implement the interface *IDeepCloneable* are also cloned using the *Cloner.Clone* function,
- Elements that have a *Clone()* function are copied using this function, and
- all other elements are simply assigned, as in this case no functionality for creating clones is available.

The here used interface *IDeepCloneable* is defined as follows:

```
namespace CloningGenerator {  
    public interface IDeepCloneable : ICloneable {  
        IDeepCloneable Clone(Cloner cloner);  
    }  
}
```

For example, let the classes *A* and *B* be defined as follows:

```
public partial class A : DeepCloneable {  
    public int i;  
    public B b;  
    public A() { }  
}  
public partial class B : DeepCloneable {  
    public string s;  
    public int[] arr;  
    public B() { }  
}
```

For these classes the following functionality is generated using the T4 template that is to be implemented:

```
partial class A {  
    protected A(A original, Cloner cloner)  
        : base(original, cloner) {  
        this.i = original.i;  
        this.b = cloner.Clone(original.b);  
    }  
    public override IDeepCloneable Clone(Cloner cloner) {  
        return new A(this, cloner);  
    }  
}
```

For finding all types that implement *IDeepCloneable* we analyze the assembly that is generated by the compiler.

### 3. SVG Generator

(8 + 4 Points)

*Scalable Vector Graphics* (SVG) is a standard for vector graphics in XML syntax<sup>1</sup>, defined by the W3C. All geometric primitives that are to be drawn are defined as tags in the XML document; using this XML document, a renderer is used for drawing the figure. For example, the following XML source text

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE svg>
<svg xmlns="http://www.w3.org/2000/svg" width="1000" height="600" viewBox="0 0 5 5">
  <rect id="black" fill="#000" width="5" height="3"/>
  <rect id="gray_i" fill="#444" width="5" height="2" y="1"/>
  <rect id="gray_ii" fill="#888" width="5" height="1" y="2"/>
  <rect id="gray_iii" fill="#ccc" width="5" height="1" y="3"/>
  <rect id="white" fill="#fff" width="5" height="1" y="4"/>
</svg>
```

defines the following graphic:



Since all tags of geometric primitives have the same structure, template based approaches can be used for generating code for generating SVG graphics. Your task is to use *FreeMarker* for developing a framework for the generation of SVG graphics that consist of arbitrary shapes.

Implement a class *Diagram* that defines the following essential properties of a diagram:

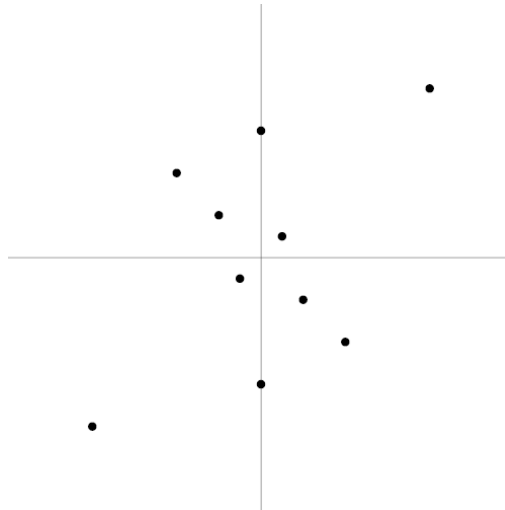
- Width and height of the graphic in pixel (int) (= screen coordinates)
- Minimum and maximum x and y values of the coordinate system for defining the part of the figure that shall be displayed (= user coordinates)
- Standard size of a shape (double)
- Display of the axes (boolean)
- List of shapes that shall be displayed

Furthermore, your task is to implement a FreeMarker-template *diagram.ftl* which is the basic frame of the SVG graphics that are to be generated. All data that is necessary for the generation of the diagram has to be passed to a class *DiagramGenerator* that uses the diagram template for code generation.

---

<sup>1</sup> Further information about the SVG 1.1 specification can be found at <http://www.w3.org/TR/SVG11/>

- a) The first version of your generator should produce simple graphics representing sets of points. Additionally, the optional display of x and y axes shall be possible. The following example can be generated; the corresponding SVG code is given below:



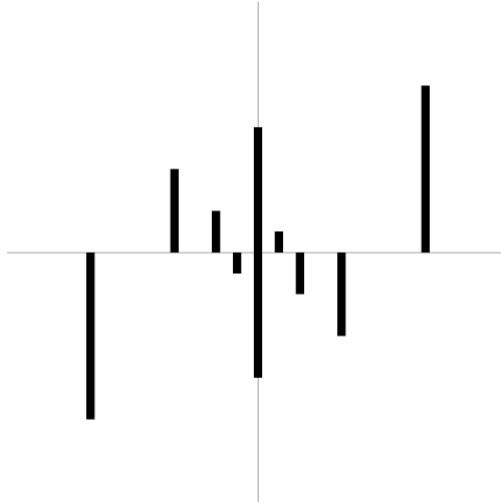
```
<?xml version="1.0" encoding="utf-8"?>
<svg xmlns=http://www.w3.org/2000/svg
      xmlns:xlink=http://www.w3.org/1999/xlink
      xmlns:ev=http://www.w3.org/2001/xml-events
      width="500"
      height="500"
      viewBox="-3.0 -3.0 6.0 6.0"
      preserveAspectRatio="none">

  <line x1="-3.0" y1="0" x2="3.0" y2="0" stroke="#555" stroke-width="0.01"/>
  <line x1="0" y1="-3.0" x2="0" y2="3.0" stroke="#555" stroke-width="0.01"/>

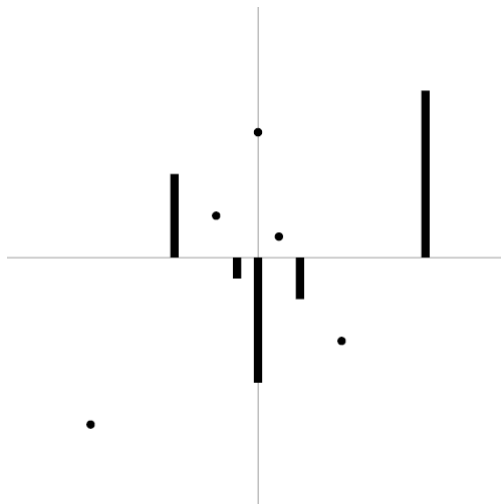
  <circle cx="-2.0" cy="2.0" r="0.05" fill="#000"/>
  <circle cx="-1.0" cy="-1.0" r="0.05" fill="#000"/>
  <circle cx="-0.5" cy="-0.5" r="0.05" fill="#000"/>
  <circle cx="-0.25" cy="0.25" r="0.05" fill="#000"/>
  <circle cx="0.0" cy="-1.5" r="0.05" fill="#000"/>
  <circle cx="0.0" cy="1.5" r="0.05" fill="#000"/>
  <circle cx="0.25" cy="-0.25" r="0.05" fill="#000"/>
  <circle cx="0.5" cy="0.5" r="0.05" fill="#000"/>
  <circle cx="1.0" cy="1.0" r="0.05" fill="#000"/>
  <circle cx="2.0" cy="-2.0" r="0.05" fill="#000"/>
</svg>
```

b) Improve your SVG generator by making it work in a modular way. For each shape a specific generator shall be implemented. One concrete variant of this generator (*PointGenerator*) has already been used implicitly in a); your task now is to implement at least two additional generators, e.g. a *RectangleGenerator* for generating bar charts or a *PolylineGenerator*. Please note: Implement the generators in analogy to the *DiagramGenerator*.

The result could look like the following example:



The combination of these generators within one graphic must also be possible:



Please note:

- Test your implementations at length (at least three test cases).
- Formulate reasonable documentations for your solutions.

## Übung 3

# 1 One, to generate them all

## 1.1 Lösungsidee

Dieser Abschnitt beschäftigt sich mit der Dokumentation der Lösungsidee der Aufgabenstellung *One, to generate them all*. Um von den *XML-Tag*-Namen und dessen Attributen unabhängig zu sein sollen die *XML-Tag*-Namen und dessen Attribute in Konstanten ausgelagert werden und sollen bei der Navigation durch das *XML*-Dokument verwendet werden, um die *XML-Tags* sowie dessen Attribute zu adressieren. Die Unabhängigkeit von der Struktur des *XML*-Dokuments ist aber nicht möglich.

Da die im *XML*-Dokument definierten Klassen sich im selben Namensraum befinden können, sollen die definierten Klassen nach den Namensräumen gruppiert werden, damit die Namensräume nicht mehrfach in den generierten Klassen definiert werden müssen. Um das zu realisieren soll *Linq* verwendet werden. Um sich zu viele *If*-Blöcke im *T4-Template* zu vermeiden soll mit dem *?* Operator gearbeitet werden, um die resultierenden Zeichenketten des *C#*-Quelltextes zu generieren. Dadurch soll das *T4-Template* übersichtlicher werden, da meiner Meinung nach das Verwenden von zu vielen *If*-Anweisungen das *Template* unübersichtlich macht.

Zusätzlich zu den verlangten Attributen der *XML-Tags* wurde noch auf Klassenebene ein *Modifier* eingeführt, womit auch abstrakte Klassen möglich sind.

Die *Tostring* Methode wurde so implementiert, dass wenn es eine Basisklasse gibt, zuerst an diese Basisklasse delegiert wird und dann erst die Ausgabe der konkreten Klasse erfolgt, wobei am Anfang einer jeder *Tostring* Implementierung immer die der Klassennamen ausgegeben wird.

## 1.2 Quelltexte

Folgender Abschnitt enthält die Quelltexte der Aufgabenstellung.

Listing 1: ClassGenerator.tt

```

1 <#@ template debug="true" hostspecific="true" language="C#" #>
2 <#@ output extension=".cs" #>
3 <#@ assembly name="System.Xml" #>
4 <#@ assembly name="System.Core" #>
5 <#@ assembly name="System.Xml.Linq, Version=4.0.0.0, Culture=neutral,
   ↳ PublicKeyToken=b77a5c561934e089" #>
6 <#@ import namespace="System.Collections.Generic" #>
7 <#@ import namespace="System.Linq" #>
8 <#@ import namespace="System.Xml.Linq" #>
9
10 <#
11     // Constants for attribute names of the XElements
12     const string ELEMENT_CLASS = "class";
13     const string ELEMENT_PROPERTY = "property";
14     const string ATTR_CLASS_NAME = "name";
15     const string ATTR_CLASS_MODIFIER = "modifier";
16     const string ATTR_NAMESPACE = "namespace";
17     const string ATTR_BASE_CLASS = "base";
18     const string ATTR_FIELD_NAME = "name";
19     const string ATTR_FIELD_TYPE = "type";
20
21     // Loading elements
22     var modelFile = Host.ResolvePath("Model.xml");
23     var xml = XElement.Load(modelFile);
24     // group by namespaces
25     var namespaceClasses = xml.Elements(ELEMENT_CLASS)

```

# Übung 3

```

26         .GroupBy(el => (el.Attribute(ATTR_NAMESPACE) != null ?
↳   el.Attribute(ATTR_NAMESPACE).Value : null),
27             el => el);
28
29     // Create classes for each namespace
30     foreach (var group in namespaceClasses)
31     {
32
33     #>
34     <#=(group.Key != null) ? ("namespace " + group.Key + " {") : ""#>
35     <#
36         // Create classes of namespace
37         foreach (var element in group)
38         {
39             // Get all all class level relevant attributes
40             var className = element.Attribute(ATTR_CLASS_NAME)?.Value;
41             var baseClass = element.Attribute(ATTR_BASE_CLASS)?.Value;
42             var classModifier = element.Attribute(ATTR_CLASS_MODIFIER);
43             var propertyList = new List<Tuple<string, string>>(element.Elements(ELEMENT_PROPERTY)
44                 .Select(prop => new Tuple<string,
↳   string>(prop.Attribute(ATTR_FIELD_TYPE)?.Value, prop.Attribute(ATTR_FIELD_NAME)?.Value)));
45     #>
46     public <#=(classModifier != null ? classModifier.Value : "")#> class <#=className
47         + (baseClass != null ? (": " + baseClass + "{") : "{")#>
48     <#
49         // Create properties
50         foreach (var propElement in propertyList)
51         {
52     #>
53     public <#=propElement.Item1 + " " + propElement.Item2#> { get; set; }
54     <#
55         }
56
57         // Create Constructor ToString if properties are present
58         if (propertyList.Count > 0)
59         {
60     #>
61     public override string ToString() {
62     <#
63         // Create toString with all attributes
64         for (var i = 0; i < propertyList.Count; i++)
65         {
66             var propItem = propertyList[i];
67             var last = i + 1 == propertyList.Count;
68             var first = i == 0;
69     #>
70     <#=(first ? "return " + ((baseClass != null) ? "base.ToString() + "\\n\\n" + " : ""
71                 + "\"Class: " + className + " \\n\\n" + "
72                 : " + ")
73                 + "\\n" + propItem.Item2 + ": \" + \" + propItem.Item2
74                 + (last ? ";\" : \" + \"\\n\\n\")\"#>
75     <#
76         }
77     #>
78     }
79     <#
80         }
81     #>
82     }
83     <#
84     }
85     #>

```

## Übung 3

```

86 <#=(group.Key != null) ? "}" : "<#>
87 <#
88     }
89 #>

```

Listing 2: Model.xml

```

1  <?xml version="1.0" encoding="utf-8" ?>
2  <model>
3      <class name="Person" namespace="Persons" modifier="abstract">
4          <property name="Name" type="string"/>
5          <property name="Age" type="int"/>
6      </class>
7      <class name="Student" namespace="Persons" base="Person">
8          <property name="Id" type="string"/>
9      </class>
10     <class name="Teacher" namespace="Persons" base="Person">
11         <property name="Id" type="string"/>
12         <property name="Department" type="Department"/>
13     </class>
14     <class name="Department" namespace="Persons" base="Person">
15         <property name="Name" type="string"/>
16     </class>
17
18     <!-- Other classes of another namespace -->
19     <class name="BaseShape" namespace="Shape">
20         <property name="Name" type="string"/>
21         <property name="Coordinate" type="Coordinate"/>
22         <property name="Stroke" type="int"/>
23         <property name="StrokeHexColor" type="string"/>
24         <property name="FillColor" type="string"/>
25     </class>
26     <class name="Rectangular" namespace="Shape" base="BaseShape">
27         <property name="Width" type="double"/>
28         <property name="Height" type="double"/>
29     </class>
30     <class name="Circle" namespace="Shape" base="BaseShape">
31         <property name="Radius" type="double"/>
32     </class>
33
34     <!-- Other classes in no namespace -->
35     <class name="Coordinate">
36         <property name="XCoordinate" type="double"/>
37         <property name="YCoordinate" type="double"/>
38     </class>
39 </model>

```

Listing 3: Program.cs

```

1  using CloningGenerator;
2  using System;
3  using Persons;
4  using Shape;
5
6  namespace ClassGenerator {
7      class Program
8      {
9          static void Main(string[] args)
10         {
11             TestPersons();

```



## Übung 3

```

12         TestShapes();
13
14         // Just to keep console open, which it wouldn't otherwise
15         Console.Read();
16     }
17
18     public static void TestPersons()
19     {
20         Console.WriteLine("##### \n"
21             + "testPersons()\n"
22             + "#####");
23
24         Department department = new Department()
25         {
26             Name = "Software engineering"
27         };
28
29         Teacher teacher = new Teacher()
30         {
31             Age = 50,
32             Department = department,
33             Id = "P00000001",
34             Name = "Dobler"
35         };
36
37         Student student = new Student() {
38             Name = "Thomas",
39             Id = "S1610454013"
40         };
41
42         Console.WriteLine(department + "\n");
43         Console.WriteLine(teacher + "\n");
44         Console.WriteLine(student + "\n");
45
46         Console.WriteLine("#####\n");
47     }
48
49     public static void TestShapes()
50     {
51         Console.WriteLine("##### \n"
52             + "testShapes()\n"
53             + "#####");
54
55         Coordinate coordinate = new Coordinate();
56         coordinate.XCoordinate = 1.0;
57         coordinate.YCoordinate = 2.0;
58
59         Rectangular rectangular = new Rectangular();
60         rectangular.Height = 1.0;
61         rectangular.Width = 2.0;
62         rectangular.Name = "Rectangular";
63         rectangular.Coordinate = coordinate;
64         rectangular.FillColor = "#ffffff";
65         rectangular.StrokeHexColor = "#000000";
66         rectangular.Stroke = 1;
67
68         Circle circle = new Circle()
69         {
70             Coordinate = coordinate,
71             FillColor = "#ffffff",
72             Name = "Circle",
73             Radius = 2.0,
74             Stroke = 1,

```

## Übung 3

```

75         StrokeHexColor = "#000000"
76     };
77
78     Console.WriteLine(coordinate + "\n");
79     Console.WriteLine(rectangular + "\n");
80     Console.WriteLine(circle + "\n");
81
82     Console.WriteLine("#####\n");
83 }
84 }
85 }

```

### 1.3 Tests

Folgender Abschnitt enthält die Tests der Aufgabenstellung in Form der generierten Quelltexte sowie der Ausgaben des Testprogramms.

Listing 4: ClassGenerator.cs

```

1  namespace Persons
2  {
3      public abstract class Person
4      {
5          public string Name { get; set; }
6          public int Age { get; set; }
7
8          public override string ToString()
9          {
10             return "Class: Person \n" + "Name: " + Name + "\n"
11                 + "Age: " + Age;
12         }
13     }
14
15     public class Student : Person
16     {
17         public string Id { get; set; }
18
19         public override string ToString()
20         {
21             return base.ToString() + "\n" + "Class: Student \n" + "Id: " + Id;
22         }
23     }
24
25     public class Teacher : Person
26     {
27         public string Id { get; set; }
28         public Department Department { get; set; }
29
30         public override string ToString()
31         {
32             return base.ToString() + "\n" + "Class: Teacher \n" + "Id: " + Id + "\n"
33                 + "Department: " + Department;
34         }
35     }
36
37     public class Department : Person
38     {
39         public string Name { get; set; }
40
41         public override string ToString()
42         {

```

# Übung 3

```

43         return base.ToString() + "\n" + "Class: Department \n" + "Name: " + Name;
44     }
45 }
46 }
47
48 namespace Shape
49 {
50     public class BaseShape
51     {
52         public string Name { get; set; }
53         public Coordinate Coordinate { get; set; }
54         public int Stroke { get; set; }
55         public string StrokeHexColor { get; set; }
56         public string FillColor { get; set; }
57
58         public override string ToString()
59         {
60             return "Class: BaseShape \n" + "Name: " + Name + "\n"
61                 + "Coordinate: " + Coordinate + "\n"
62                 + "Stroke: " + Stroke + "\n"
63                 + "StrokeHexColor: " + StrokeHexColor + "\n"
64                 + "FillColor: " + FillColor;
65         }
66     }
67
68     public class Rectangular : BaseShape
69     {
70         public double Width { get; set; }
71         public double Height { get; set; }
72
73         public override string ToString()
74         {
75             return base.ToString() + "\n" + "Class: Rectangular \n" + "Width: " + Width + "\n"
76                 + "Height: " + Height;
77         }
78     }
79
80     public class Circle : BaseShape
81     {
82         public double Radius { get; set; }
83
84         public override string ToString()
85         {
86             return base.ToString() + "\n" + "Class: Circle \n" + "Radius: " + Radius;
87         }
88     }
89 }
90
91
92 public class Coordinate
93 {
94     public double XCoordinate { get; set; }
95     public double YCoordinate { get; set; }
96
97     public override string ToString()
98     {
99         return "Class: Coordinate \n" + "XCoordinate: " + XCoordinate + "\n"
100             + "YCoordinate: " + YCoordinate;
101     }
102 }

```

```
#####
testPersons()
#####
Class: Person
Name:
Age: 0
Class: Department
Name: Software engineering

Class: Person
Name: Dobler
Age: 50
Class: Teacher
Id: P00000001
Department: Class: Person
Name:
Age: 0
Class: Department
Name: Software engineering

Class: Person
Name: Thomas
Age: 0
Class: Student
Id: S1610454013

#####

#####
testShapes()
#####
Class: Coordinate
XCoordinate: 1
YCoordinate: 2

Class: BaseShape
Name: Rectangular
Coordinate: Class: Coordinate
XCoordinate: 1
YCoordinate: 2
Stroke: 1
StrokeHexColor: #000000
FillColor: #ffffff
Class: Rectangular
Width: 2
Height: 1

Class: BaseShape
Name: Circle
Coordinate: Class: Coordinate
XCoordinate: 1
YCoordinate: 2
Stroke: 1
StrokeHexColor: #000000
FillColor: #ffffff
Class: Circle
Radius: 2

#####
```

Abbildung 1: *Class* Generator Testausgabe

## Übung 3

## 2 Clone 'Em All

### 2.1 Lösungsidee

Dieser Abschnitt beschäftigt sich mit der Dokumentation der Lösungsidee der Aufgabenstellung *Clone 'Em All*. Diese Aufgabenstellung wurde bereits in der Übung implementiert und nachträglich formatiert und einige Kommentare eingefügt.

### 2.2 Quelltexte

In diesem Abschnitt werden nur die selbst implementierten Quelltexte angeführt.

Listing 5: CloningGenerator.tt

```

1 <#@ template debug="true" language="C#" hostSpecific="true" #>
2 <#@ output extension=".cs" #>
3 <#@ assembly name="System.Core.dll" #>
4 <#@ import namespace="System.Reflection" #>
5 <#@ import namespace="System.IO" #>
6 <#@ import namespace="System.Linq" #>
7 <#
8     string assemblyFile = @"bin\Debug\CloningGenerator.exe";
9     Assembly assembly = LoadProjectAssembly(assemblyFile);
10    var deepClonables = assembly.GetTypes().Where(x => !x.IsInterface && x.Name !=
    ↳ "DeepCloneable" && x.GetInterfaces()
11                                     .Any(y => y.Name == "IDeepCloneable"));
12 #>
13 namespace CloningGenerator {
14 <#
15     // Iterate over each found type
16     foreach (var t in deepClonables)
17     {
18 #>
19 public partial class <# t.Name #> {
20
21     // Clone constructor
22     protected <# t.Name #>(<# t.Name #> original, Cloner cloner): base(original, cloner) {
23 <#
24         // Iterate over each found field of the found type
25         foreach (var fieldInfo in t.GetFields())
26         {
27             var fieldType = fieldInfo.FieldType;
28             // If field type is instance of IDeepCloneable
29             if (fieldType.GetInterfaces().Any(y => y.Name == "IDeepCloneable"))
30             {
31 #>
32 this.<# fieldInfo.Name #> = cloner.Clone(original.<# fieldInfo.Name #>);
33 <#
34             }
35             // If field type is instance of Clonable
36             else if (typeof(IClonable).IsAssignableFrom(fieldType))
37             {
38 #>
39 this.<# fieldInfo.Name #> = (<# fieldType.FullName #>) original.<# fieldInfo.Name #>.Clone();
40 <#
41             }
42             // If field type is primitive one
43             else
44             {
45 #>
46 this.<# fieldInfo.Name #> = original.<# fieldInfo.Name #>;

```

## Übung 3

```

47 <#
48         }
49     }
50 #>
51 }
52
53 public override IDepCloneable Clone(Cloner cloner){
54     return new <#= t.Name #>(this, cloner);
55 }
56
57 }
58 <#
59     }
60 #>
61 }
62 <#
63     // Cleanup after generation
64     Cleanup();
65 #>
66 <#+
67     // Class feature blocks for util methods
68     private string outputDir;
69
70     private Assembly LoadProjectAssembly(string assemblyFile)
71     {
72         if (!Path.IsPathRooted(assemblyFile))
73         {
74             assemblyFile = Path.Combine(Path.GetDirectoryName(Host.TemplateFile), assemblyFile);
75         }
76         outputDir = Path.GetDirectoryName(assemblyFile);
77         if (!File.Exists(assemblyFile))
78         {
79             throw new ArgumentException(string.Format("Project assembly file could not be located
↵ at {0}.",
80                 assemblyFile));
81         }
82         AppDomain.CurrentDomain.AssemblyResolve += ResolveAssembly;
83
84         return Assembly.Load(File.ReadAllBytes(assemblyFile));
85     }
86
87     private void Cleanup()
88     {
89         AppDomain.CurrentDomain.AssemblyResolve -= ResolveAssembly;
90     }
91
92     private Assembly ResolveAssembly(object sender, ResolveEventArgs args)
93     {
94         string dependency = Path.Combine(outputDir, args.Name.Substring(0, args.Name.IndexOf(',')))
↵ + ".dll");
95         if (File.Exists(dependency))
96         {
97             return Assembly.Load(File.ReadAllBytes(dependency));
98         }
99
100         return null;
101     }
102 #>

```

Listing 6: A.cs

## Übung 3

```

1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5
6 namespace CloningGenerator
7 {
8     // Test class for cloning generator
9     public partial class A : DeepCloneable
10    {
11        public int i;
12        public B b;
13        public A() { }
14    }
15 }

```

Listing 7: B.cs

```

1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5
6 namespace CloningGenerator
7 {
8     // Test class for cloning generator
9     public partial class B : DeepCloneable
10    {
11        public string s;
12        public int[] arr;
13        public B() { }
14    }
15
16 }

```

Listing 8: Program.cs

```

1 using CloningGenerator;
2 using System;
3
4 namespace ClassGenerator
5 {
6     class Program
7     {
8         static void Main(string[] args)
9         {
10             var a = new A();
11             var b = new B
12             {
13                 s = "Hello world",
14                 arr = new[] { 1, 2, 3 }
15             };
16
17             a.b = b;
18
19             Console.WriteLine("##### \n"
20                             + "Before clone\n"
21                             + "#####");
22             Console.WriteLine("a.i : " + a.i);

```

## Übung 3

```

23 Console.WriteLine("a.b.s : " + a.b.s);
24 Console.WriteLine("a.b.arr: " + a.b.arr[0]);
25 Console.WriteLine("#####\n");
26
27 // clone object
28 var clone = (A)a.Clone();
29
30 // change value
31 a.i = 1337;
32 a.b.s = "bye world";
33 a.b.arr[0] = 0;
34
35 Console.WriteLine("##### \n"
36 + "After clone\n"
37 + "#####");
38 Console.WriteLine("a.i : " + a.i);
39 Console.WriteLine("a.b.s : " + a.b.s);
40 Console.WriteLine("a.b.arr: " + a.b.arr[0] + "\n");
41
42 Console.WriteLine("clone.i : " + clone.i);
43 Console.WriteLine("clone.b.s : " + clone.b.s);
44 Console.WriteLine("clone.b.arr: " + clone.b.arr[0] + "\n");
45 Console.WriteLine("#####\n");
46
47 Console.ReadLine();
48 }
49 }
50 }

```

## 2.3 Tests

Dieser Abschnitt enthält die Tests in Form von den generierten Quelltexten und der Ausgabe des Testprogramms.

Listing 9: CloningGenerator.cs

```

1 namespace CloningGenerator
2 {
3     public partial class A
4     {
5         // Clone constructor
6         protected A(A original, Cloner cloner) : base(original, cloner)
7         {
8             this.i = original.i;
9             this.b = cloner.Clone(original.b);
10        }
11
12        public override IDepCloneable Clone(Cloner cloner)
13        {
14            return new A(this, cloner);
15        }
16    }
17
18    public partial class B
19    {
20        // Clone constructor
21        protected B(B original, Cloner cloner) : base(original, cloner)
22        {
23            this.s = (System.String) original.s.Clone();
24            this.arr = (System.Int32[]) original.arr.Clone();
25        }

```



## Übung 3

```
26  
27     public override IDepCloneable Clone(Cloner cloner)  
28     {  
29         return new B(this, cloner);  
30     }  
31 }  
32 }
```

```
#####  
Before clone  
#####  
a.i      : 0  
a.b.s    : Hello world  
a.b.arr: 1  
#####  
  
#####  
After clone  
#####  
a.i      : 1337  
a.b.s    : bye world  
a.b.arr: 0  
  
clone.i   : 0  
clone.b.s : Hello world  
clone.b.arr: 1  
  
#####
```

Abbildung 2: Clone Generator Testausgabe

### 3 SVG Generator

#### 3.1 Lösungsidee

Dieser Abschnitt beschäftigt sich mit der Dokumentation der Lösungsidee der Aufgabenstellung SVG Generator. Es sollen zwei *Exception*-Klassen implementiert werden, wobei eine *Exception*-Klasse einen durch den Generator verursachten Fehler repräsentiert und die andere *Exception*-Klasse einen durch eine *Shape* ausgelösten Fehler repräsentiert.

Es soll jeweils eine Schnittstelle für die Repräsentation eines *Generators* und eines *Shapes* implementiert werden um beide Aspekte von ihren Implementierungen zu trennen. Es sollen abstrakte Klassen für die Generatoren und die *Shapes* implementiert werden, die deren gemeinsame Funktionalität kapseln. Die implementierten Generatoren und *Shapes* sollen von diesen Klassen erben und die spezifischen Funktionalitäten und Zustände implementieren.

Damit nicht zu viele Klassendateien für die SVG Generatoren angelegt werden sollen die Generatorenklassen als statische innere Klassen in einer Klasse gebündelt werden. Es sollen die folgenden SVG-Tags

- *PointShape*,
- *LineShape*,
- *RectangularShape* und
- *TextShape*

implementiert werden.

## Übung 3

Um sich die *Getter* und *Setter*-Methoden zu ersparen wurde die Bibliothek *lombok* verwendet, die über Annotationen die *Getter* und *Setter*-Methoden generiert, wobei der *Compile* instrumentiert wird. Siehe folgenden Link wie *lombok* verwendet wird und wie die Bibliothek *lombok* in eine *IDE* integriert wird <https://projectlombok.org/download.html>.

Ohne diese Integration von der Bibliothek *lombok* kann das Program nicht in einer IDE gebaut werden.

### 3.2 Quelltexte

Listing 10: GeneratorException.java

```

1 package at.fh.ooe.gp2.template.api.exception;
2
3 /**
4  * @author Thomas Herzog <t.herzog@curecomp.com>
5  * @since 04/23/17
6  */
7 public class GeneratorException extends Exception {
8
9     public GeneratorException() {
10    }
11
12     public GeneratorException(String message) {
13         super(message);
14     }
15
16     public GeneratorException(String message,
17                               Throwable cause) {
18         super(message, cause);
19     }
20
21     public GeneratorException(Throwable cause) {
22         super(cause);
23     }
24
25     public GeneratorException(String message,
26                               Throwable cause,
27                               boolean enableSuppression,
28                               boolean writableStackTrace) {
29         super(message, cause, enableSuppression, writableStackTrace);
30     }
31 }

```

Listing 11: GeneratorException.java

```

1 package at.fh.ooe.gp2.template.api.exception;
2
3 /**
4  * @author Thomas Herzog <t.herzog@curecomp.com>
5  * @since 04/23/17
6  */
7 public class RenderException extends Exception {
8
9     public RenderException() {
10    }
11
12     public RenderException(String message) {
13         super(message);
14     }
15 }

```

## Übung 3

```

16     public RenderException(String message,
17                             Throwable cause) {
18         super(message, cause);
19     }
20
21     public RenderException(Throwable cause) {
22         super(cause);
23     }
24
25     public RenderException(String message,
26                             Throwable cause,
27                             boolean enableSuppression,
28                             boolean writableStackTrace) {
29         super(message, cause, enableSuppression, writableStackTrace);
30     }
31 }

```

Listing 12: Generator.java

```

1  package at.fh.ooe.gp2.template.api.generator;
2
3  import at.fh.ooe.gp2.template.api.exception.GeneratorException;
4  import at.fh.ooe.gp2.template.api.shape.Shape;
5
6  /**
7   * This interface marks a class as a generator.
8   *
9   * @author Thomas Herzog <t.herzog@curecomp.com>
10  * @since 04/21/17
11  */
12  public interface Generator<S extends Shape> {
13
14      /**
15       * Generates the specific representation of the shapes.
16       *
17       * @param shape the shape to generate representation for
18       * @return the specific string representation of the shape
19       * @throws GeneratorException if the generation fails
20       */
21      String generate(S shape) throws GeneratorException;
22  }

```

Listing 13: Shape.java

```

1  package at.fh.ooe.gp2.template.api.shape;
2
3  import at.fh.ooe.gp2.template.api.exception.RenderException;
4
5  /**
6   * This interfaces marks a class as a shape.
7   *
8   * @author Thomas Herzog <t.herzog@curecomp.com>
9   * @since 04/23/17
10  */
11  public interface Shape {
12
13      /**
14       * Renders this shape.
15       *
16       * @return the rendered string representation of this shape

```

## Übung 3

```

17     * @throws RenderException if the shape rendering failed
18     */
19     String render() throws RenderException;
20 }

```

Listing 14: Coordinate.java

```

1  package at.fh.ooe.gp2.template.api;
2
3
4  import lombok.Getter;
5
6  import java.util.Objects;
7
8  /**
9   * An instance of this class represents a coordinate of an shape.
10   *
11   * @author Thomas Herzog <t.herzog@curecomp.com>
12   * @since 04/23/17
13   */
14  public class Coordinate {
15
16      @Getter
17      private final double x;
18      @Getter
19      private final double y;
20
21      public Coordinate(double x,
22                        double y) {
23          this.x = x;
24          this.y = y;
25      }
26
27      @Override
28      public boolean equals(Object o) {
29          if (this == o) return true;
30          if (o == null || getClass() != o.getClass()) return false;
31          Coordinate that = (Coordinate) o;
32          return x == that.x &&
33                 y == that.y;
34      }
35
36      @Override
37      public int hashCode() {
38          return Objects.hash(x, y);
39      }
40 }

```

Listing 15: AbstractShape.java

```

1  package at.fh.ooe.gp2.template.api.shape;
2
3  import at.fh.ooe.gp2.template.api.*;
4  import at.fh.ooe.gp2.template.api.exception.GeneratorException;
5  import at.fh.ooe.gp2.template.api.exception.RenderException;
6  import at.fh.ooe.gp2.template.api.generator.Generator;
7  import lombok.Getter;
8
9  import java.awt.*;
10 import java.util.Objects;

```

# Übung 3

```

11
12 /**
13  * @author Thomas Herzog <t.herzog@curecomp.com>
14  * @since 04/21/17
15  */
16 public abstract class AbstractShape<S extends Shape> implements Shape {
17
18     @Getter
19     private final Shape parent;
20     @Getter
21     private final Generator<S> generator;
22     @Getter
23     private final Coordinate origin;
24     @Getter
25     private final double strokeWidth;
26     private final Color strokeColor;
27     private final Color fillColor;
28
29     public AbstractShape(final Generator<S> generator,
30                         final Coordinate origin) {
31         this(null, generator, origin, null, null, 1);
32     }
33
34     public AbstractShape(final at.fh.ooe.gp2.template.api.shape.Shape parent,
35                         final Generator<S> generator,
36                         final Coordinate origin) {
37         this(parent, generator, origin, null, null, 1);
38     }
39
40     public AbstractShape(final at.fh.ooe.gp2.template.api.shape.Shape parent,
41                         final Generator<S> generator,
42                         final Coordinate origin,
43                         final Color strokeColor,
44                         final Color fillColor,
45                         final double strokeWidth) {
46         Objects.requireNonNull(generator, "Shape must hold an generator");
47
48         this.parent = parent;
49         this.generator = generator;
50         this.origin = origin;
51         this.strokeColor = strokeColor;
52         this.fillColor = fillColor;
53         this.strokeWidth = strokeWidth;
54     }
55
56     public String render() throws RenderException {
57         try {
58             return generator.generate((S) this);
59         } catch (GeneratorException e) {
60             throw new RenderException("Shape execution failed", e);
61         }
62     }
63
64     protected String colorToHexString(final Color color) {
65         String hex = null;
66         if (color != null) {
67             hex = "#" + Integer.toHexString(color.getRGB()).substring(2).toUpperCase();
68         }
69         return hex;
70     }
71
72     public String getStrokeColor() {
73         return (strokeColor != null) ? colorToHexString(strokeColor) : null;

```

# Übung 3

```

74     }
75
76     public String getFillColor() {
77         return (fillColor != null) ? colorToHexString(fillColor) : null;
78     }
79 }

```

Listing 16: Diagram.java

```

1  package at.fh.ooe.gp2.template.api.shape;
2
3  import at.fh.ooe.gp2.template.impl.generator.FreeMarkerGenerators;
4  import lombok.Getter;
5
6  import java.util.LinkedList;
7  import java.util.List;
8
9  public class Diagram extends AbstractShape<Diagram> {
10
11     @Getter
12     private final int width;
13     @Getter
14     private final int height;
15     @Getter
16     private final Double minX;
17     @Getter
18     private final Double maxX;
19     @Getter
20     private final Double minY;
21     @Getter
22     private final Double maxY;
23     @Getter
24     private final boolean showAxis;
25     @Getter
26     private final List<AbstractShape> shapes = new LinkedList<>();
27
28     public Diagram(final FreeMarkerGenerators.DiagramGenerator generator,
29                   final int width,
30                   final int height,
31                   final boolean showAxis) {
32         this(generator, width, height, null, null, null, null, showAxis);
33     }
34
35     public Diagram(final FreeMarkerGenerators.DiagramGenerator generator,
36                   final int width,
37                   final int height,
38                   final Double minX,
39                   final Double maxX,
40                   final Double minY,
41                   final Double maxY,
42                   final boolean showAxis) {
43         super(generator, null);
44         this.width = width;
45         this.height = height;
46         this.minX = minX;
47         this.maxX = maxX;
48         this.minY = minY;
49         this.maxY = maxY;
50         this.showAxis = showAxis;
51     }
52
53     public void addShape(AbstractShape s) {

```

## Übung 3

```

54     shapes.add(s);
55 }
56
57 public boolean isViewBoxEnabled(){
58     return (minX != null) && (maxX!= null) && (minY!=null) && (maxY!=null);
59 }
60 }

```

Listing 17: LineShape.java

```

1  package at.fh.ooe.gp2.template.api.shape;
2
3  import at.fh.ooe.gp2.template.impl.generator.FreemarkerGenerators;
4  import at.fh.ooe.gp2.template.api.Coordinate;
5  import lombok.Getter;
6
7  import java.awt.*;
8
9  /**
10   * @author Thomas Herzog <t.herzog@curecomp.com>
11   * @since 04/23/17
12   */
13  public class LineShape extends AbstractShape<LineShape> {
14
15      @Getter
16      private final Coordinate end;
17
18      public LineShape(final Shape parent,
19                      final FreemarkerGenerators.LineGenerator generator,
20                      final Coordinate origin,
21                      final Coordinate end) {
22          this(parent, generator, origin, end, null, 1);
23      }
24
25      public LineShape(final Shape diagram,
26                      final FreemarkerGenerators.LineGenerator generator,
27                      final Coordinate origin,
28                      final Coordinate end,
29                      final Color stroke,
30                      final double strokeWidth) {
31          super(diagram, generator, origin, stroke, null, strokeWidth);
32          this.end = end;
33      }
34
35  }

```

Listing 18: RectangularShape.java

```

1  package at.fh.ooe.gp2.template.api.shape;
2
3
4  import at.fh.ooe.gp2.template.api.*;
5  import at.fh.ooe.gp2.template.impl.generator.FreemarkerGenerators;
6  import lombok.Getter;
7
8  import java.awt.*;
9
10 /**
11  * @author Thomas Herzog <t.herzog@curecomp.com>
12  * @since 04/23/17

```

## Übung 3

```

13  */
14  public class RectangularShape extends AbstractShape<RectangularShape> {
15
16      @Getter
17      private final double height;
18      @Getter
19      private final double width;
20
21      public RectangularShape(final Shape diagram,
22                              final FreemarkerGenerators.RectangularGenerator generator,
23                              final Coordinate origin,
24                              final Color stroke,
25                              final Color fill,
26                              double height,
27                              final double width,
28                              final double strokeWidth) {
29          super(diagram, generator, origin, stroke, fill, strokeWidth);
30          this.height = height;
31          this.width = width;
32      }
33  }

```

Listing 19: TextShape.java

```

1  package at.fh.ooe.gp2.template.api.shape;
2
3  import at.fh.ooe.gp2.template.impl.generator.FreemarkerGenerators;
4  import at.fh.ooe.gp2.template.api.Coordinate;
5  import lombok.Getter;
6
7  import java.awt.*;
8
9  /**
10   * @author Thomas Herzog <t.hertzog@curecomp.com>
11   * @since 04/23/17
12   */
13  public class TextShape extends AbstractShape<TextShape> {
14
15      @Getter
16      private final String text;
17      @Getter
18      private final String fontFamily;
19      @Getter
20      private final double fontSize;
21
22      public TextShape(final Shape diagram,
23                      final FreemarkerGenerators.TextGenerator generator,
24                      final Coordinate coordinate,
25                      final String text) {
26          this(diagram, generator, coordinate, null, null, text, "Verdana", 10, 1);
27      }
28
29      public TextShape(final Shape diagram,
30                      final FreemarkerGenerators.TextGenerator generator,
31                      final Coordinate origin,
32                      final Color stroke,
33                      final Color fill,
34                      final String text,
35                      final String fontFamily,
36                      final double fontSize,
37                      final double strokeWidth) {
38          super(diagram, generator, origin, stroke, fill, strokeWidth);

```



## Übung 3

```

39     this.text = text;
40     this.fontFamily = fontFamily;
41     this.fontSize = fontSize;
42 }
43 }

```

Listing 20: AbstractFreemarkerGenerator.java

```

1  package at.fh.ooe.gp2.template.impl.generator;
2
3  import at.fh.ooe.gp2.template.api.generator.Generator;
4  import at.fh.ooe.gp2.template.api.exception.GeneratorException;
5  import at.fh.ooe.gp2.template.api.shape.Shape;
6  import freemarker.template.Configuration;
7  import freemarker.template.Template;
8  import freemarker.template.TemplateExceptionHandler;
9
10 import java.io.StringWriter;
11 import java.io.Writer;
12 import java.util.Locale;
13 import java.util.Objects;
14
15 /**
16  * @author Thomas Herzog <t.herzog@curecomp.com>
17  * @since 04/23/17
18  */
19 public abstract class AbstractFreemarkerGenerator<S extends Shape> implements Generator<S> {
20
21     private final Template template;
22
23     private static final Configuration config;
24
25     static {
26         config = new Configuration(Configuration.VERSION_2_3_24);
27         config.setClassForTemplateLoading(AbstractFreemarkerGenerator.class,
↵      "/templates/shapes/svg");
28         config.setDefaultEncoding("UTF-8");
29         config.setLocale(Locale.US);
30         config.setTemplateExceptionHandler(TemplateExceptionHandler.RETHROW_HANDLER);
31     }
32
33     public AbstractFreemarkerGenerator(final String templateLocation) throws Exception {
34         Objects.requireNonNull(templateLocation, "Template root location must not be null");
35
36         template = config.getTemplate(templateLocation);
37     }
38
39     @Override
40     public String generate(final S shape) throws GeneratorException {
41         Objects.requireNonNull(shape, "Shape for generation must not be null");
42
43         final String result;
44         try (final Writer writer = new StringWriter();) {
45             template.process(shape, writer);
46             writer.flush();
47             result = writer.toString();
48         } catch (Exception e) {
49             throw new GeneratorException("Generation failed", e);
50         }
51
52         return result;
53     }

```

# Übung 3

54 }

Listing 21: FreemarkerGenerators.java

```

1 package at.fh.ooe.gp2.template.impl.generator;
2
3 import at.fh.ooe.gp2.template.api.shape.*;
4
5 /**
6  * @author Thomas Herzog <t.hertzog@curecomp.com>
7  * @since 04/30/17
8  */
9 public class FreemarkerGenerators {
10
11     private FreemarkerGenerators() {
12     }
13
14     public static class DiagramGenerator extends AbstractFreemarkerGenerator<Diagram> {
15
16         public DiagramGenerator() throws Exception {
17             super("diagram.ftl");
18         }
19     }
20
21     public static class PointGenerator extends AbstractFreemarkerGenerator<PointShape> {
22
23         public PointGenerator() throws Exception {
24             super("point.ftl");
25         }
26     }
27
28     public static class LineGenerator extends AbstractFreemarkerGenerator<LineShape> {
29
30         public LineGenerator() throws Exception {
31             super("line.ftl");
32         }
33     }
34
35     public static class RectangularGenerator extends AbstractFreemarkerGenerator<RectangularShape>
36     ↪ {
37
38         public RectangularGenerator() throws Exception {
39             super("rect.ftl");
40         }
41     }
42
43     public static class TextGenerator extends AbstractFreemarkerGenerator<TextShape> {
44
45         public TextGenerator() throws Exception {
46             super("text.ftl");
47         }
48     }
49 }

```

Listing 22: Main.java

```

1 package at.fh.ooe.gp2.template;
2
3 import at.fh.ooe.gp2.template.api.Coordinate;
4 import at.fh.ooe.gp2.template.api.generator.Generator;

```

# Übung 3

```

5 import at.fh.ooe.gp2.template.api.shape.*;
6 import at.fh.ooe.gp2.template.impl.generator.FreemarkerGenerators;
7
8 import java.awt.*;
9 import java.io.File;
10 import java.io.FileWriter;
11 import java.io.Writer;
12
13 public class Main {
14
15     public static Diagram createDiagramBoxes() throws Exception {
16         final FreemarkerGenerators.RectangularGenerator rectGenerator = new
↪ FreemarkerGenerators.RectangularGenerator();
17         final Diagram diagram = new Diagram(new FreemarkerGenerators.DiagramGenerator(), 500, 400,
↪ false);
18
19         diagram.addShape(new RectangularShape(diagram, rectGenerator, new Coordinate(0, 0), null,
↪ Color.black, 100, 500, 1));
20         diagram.addShape(new RectangularShape(diagram, rectGenerator, new Coordinate(0, 100),
↪ null, Color.decode("#444444"), 100, 500, 1));
21         diagram.addShape(new RectangularShape(diagram, rectGenerator, new Coordinate(0, 200),
↪ null, Color.decode("#888888"), 100, 500, 1));
22         diagram.addShape(new RectangularShape(diagram, rectGenerator, new Coordinate(0, 300),
↪ null, Color.decode("#cccccc"), 100, 500, 1));
23
24         return diagram;
25     }
26
27
28     public static Diagram createDiagramPoint() throws Exception {
29         final FreemarkerGenerators.PointGenerator pointGenerator = new
↪ FreemarkerGenerators.PointGenerator();
30         final Diagram diagram = new Diagram(new FreemarkerGenerators.DiagramGenerator(), 500, 400,
↪ -3.0, 3.0, -3.0, 3.0, true);
31
32         diagram.addShape(new PointShape(diagram, pointGenerator, new Coordinate(-2.0, 2.0), null,
↪ Color.black, 0.05, 0.01));
33         diagram.addShape(new PointShape(diagram, pointGenerator, new Coordinate(-1.0, -1.0), null,
↪ Color.black, 0.05, 0.01));
34         diagram.addShape(new PointShape(diagram, pointGenerator, new Coordinate(-0.5, -0.5), null,
↪ Color.black, 0.05, 0.01));
35         diagram.addShape(new PointShape(diagram, pointGenerator, new Coordinate(-0.25, 0.25),
↪ null, Color.black, 0.05, 0.01));
36         diagram.addShape(new PointShape(diagram, pointGenerator, new Coordinate(0.0, -1.5), null,
↪ Color.black, 0.05, 0.01));
37         diagram.addShape(new PointShape(diagram, pointGenerator, new Coordinate(0.0, 1.5), null,
↪ Color.black, 0.05, 0.01));
38         diagram.addShape(new PointShape(diagram, pointGenerator, new Coordinate(0.25, -0.25),
↪ null, Color.black, 0.05, 0.01));
39         diagram.addShape(new PointShape(diagram, pointGenerator, new Coordinate(0.5, 0.5), null,
↪ Color.black, 0.05, 0.01));
40         diagram.addShape(new PointShape(diagram, pointGenerator, new Coordinate(1.0, 1.0), null,
↪ Color.black, 0.05, 0.01));
41         diagram.addShape(new PointShape(diagram, pointGenerator, new Coordinate(2.0, -2.0), null,
↪ Color.black, 0.05, 0.01));
42
43         return diagram;
44     }
45
46     public static Diagram createDiagramLine() throws Exception {
47         final FreemarkerGenerators.LineGenerator lineGenerator = new
↪ FreemarkerGenerators.LineGenerator();
48         final Diagram diagram = new Diagram(new FreemarkerGenerators.DiagramGenerator(), 500, 400,
↪ false);

```

# Übung 3

```

49         diagram.addShape(new LineShape(diagram, lineGenerator, new Coordinate(10,0), new
50 ↪ Coordinate(10, 400), Color.BLACK, 4));
51         diagram.addShape(new LineShape(diagram, lineGenerator, new Coordinate(14,0), new
52 ↪ Coordinate(14, 500), Color.RED, 2));
53         diagram.addShape(new LineShape(diagram, lineGenerator, new Coordinate(0,50), new
54 ↪ Coordinate(495, 50), Color.BLUE, 5));
55         diagram.addShape(new LineShape(diagram, lineGenerator, new Coordinate(500,0), new
56 ↪ Coordinate(0, 400), Color.YELLOW, 10));
57
58         return diagram;
59     }
60
61     public static Diagram createDiagramText() throws Exception {
62         final FreemarkerGenerators.TextGenerator textGenerator = new
63 ↪ FreemarkerGenerators.TextGenerator();
64         final Diagram diagram = new Diagram(new FreemarkerGenerators.DiagramGenerator(), 500, 400,
65 ↪ false);
66
67         diagram.addShape(new TextShape(diagram,textGenerator, new Coordinate(0,40), Color.black,
68 ↪ Color.RED, "My Text", "Verdana", 40, 1));
69         diagram.addShape(new TextShape(diagram,textGenerator, new Coordinate(100,80), Color.black,
70 ↪ Color.BLUE, "is in", "Verdana", 40, 1));
71         diagram.addShape(new TextShape(diagram,textGenerator, new Coordinate(170,120),
72 ↪ Color.black, Color.GREEN, "the Font Verdana", "Verdana", 40, 3));
73
74         return diagram;
75     }
76
77     public static Diagram createDiagramAll() throws Exception {
78         final FreemarkerGenerators.PointGenerator pointGenerator = new
79 ↪ FreemarkerGenerators.PointGenerator();
80         final FreemarkerGenerators.RectangularGenerator rectGenerator = new
81 ↪ FreemarkerGenerators.RectangularGenerator();
82         final FreemarkerGenerators.TextGenerator textGenerator = new
83 ↪ FreemarkerGenerators.TextGenerator();
84         final FreemarkerGenerators.LineGenerator lineGenerator = new
85 ↪ FreemarkerGenerators.LineGenerator();
86         final Diagram diagram = new Diagram(new FreemarkerGenerators.DiagramGenerator(), 500, 400,
87 ↪ -3.0, 3.0, -3.0, 3.0, true);
88
89         // left positive quadrant
90         diagram.addShape(new RectangularShape(diagram, rectGenerator, new Coordinate(-2.8, -3),
91 ↪ Color.red, Color.black, 3, 0.1, 0.01));
92         diagram.addShape(new RectangularShape(diagram, rectGenerator, new Coordinate(-2, -1.5),
93 ↪ Color.green, Color.yellow, 1.5, 0.1, 0.01));
94         diagram.addShape(new RectangularShape(diagram, rectGenerator, new Coordinate(-1, -1.2),
95 ↪ Color.black, Color.black, 1.2, 0.1, 0.01));
96         diagram.addShape(new RectangularShape(diagram, rectGenerator, new Coordinate(-0.5, -0.8),
97 ↪ Color.blue, Color.black, 0.8, 0.1,0.01));
98
99         // left negative quadrant
100        diagram.addShape(new PointShape(diagram, pointGenerator, new Coordinate(-0.5, 0.5), null,
101 ↪ Color.black, 0.05, 0.01));
102        diagram.addShape(new PointShape(diagram, pointGenerator, new Coordinate(-1.0, 1.0), null,
103 ↪ Color.red, 0.05, 0.01));
104        diagram.addShape(new PointShape(diagram, pointGenerator, new Coordinate(-2.0, 1.0), null,
105 ↪ Color.green, 0.05, 0.01));
106        diagram.addShape(new PointShape(diagram, pointGenerator, new Coordinate(-2.5, 2.8), null,
107 ↪ Color.yellow, 0.05, 0.01));
108        diagram.addShape(new PointShape(diagram, pointGenerator, new Coordinate(-3.0, 3.0), null,
109 ↪ Color.black, 0.05, 0.01));

```

# Übung 3

```

89      // right positive quadrant
90      diagram.addShape(new TextShape(diagram,textGenerator, new Coordinate(0,-2), Color.black,
↪ Color.lightGray, "My Text", "Verdana", 0.5, 0.01));
91      diagram.addShape(new TextShape(diagram,textGenerator, new Coordinate(0,-1), Color.black,
↪ Color.lightGray, "is here", "Verdana", 0.5, 0.01));
92
93      // right negative quadrant
94      diagram.addShape(new LineShape(diagram, lineGenerator, new Coordinate(0,0), new
↪ Coordinate(2, 2), Color.BLACK, 0.1));
95      diagram.addShape(new LineShape(diagram, lineGenerator, new Coordinate(1,0), new
↪ Coordinate(1, 2), Color.RED, 0.05));
96      return diagram;
97  }
98
99  private static final void createSvgFile(final Diagram diagram,
100                                         final Generator<Diagram> generator,
101                                         final String filename) throws Exception {
102      String svgContent = generator.generate(diagram);
103      File svgFile = new File("output/" + filename + ".svg");
104      Writer fileWriter = new FileWriter(svgFile);
105      fileWriter.write(svgContent);
106      fileWriter.close();
107
108      System.out.println("svg generated to " + svgFile.getAbsolutePath());
109  }
110
111  public static void main(String[] args) throws Exception {
112      final Generator<Diagram> generator = new FreemarkerGenerators.DiagramGenerator();
113
114      createSvgFile(createDiagramBoxes(), generator, "boxes");
115      createSvgFile(createDiagramPoint(), generator, "points");
116      createSvgFile(createDiagramLine(), generator, "lines");
117      createSvgFile(createDiagramText(), generator, "texts");
118      createSvgFile(createDiagramAll(), generator, "all");
119  }
120 }

```

Listing 23: diagram.ftl

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <!DOCTYPE svg>
3  <svg xmlns="http://www.w3.org/2000/svg"
4      width="${width}"
5      height="${height}"
6      <#if viewBoxEnabled>viewBox="${minX} ${minY} ${maxX - minX} ${maxY - minY}"
7      preserveAspectRatio="none"</#if>>
8
9  <#if showAxis>
10     <line x1="${minX}" x2="${maxX}" y1="${0}" y2="${0}" stroke="#555" stroke-width="0.01" />
11     <line x1="${0}" x2="${0}" y1="${minY}" y2="${maxY}" stroke="#555" stroke-width="0.01" />
12 </#if>
13
14 <#list shapes as shape>
15     ${shape.render()}
16 </#list>
17
18 </svg>

```

Listing 24: point.ftl

## Übung 3

```

1 <circle cx="${origin.x}"
2       cy="${origin.y}"
3       r="${radius}"
4       stroke-width="${strokeWidth}"
5       <#if strokeColor??>stroke="${strokeColor}"</#if>
6       <#if fillColor??>fill="${fillColor}"</#if>/>

```

Listing 25: line.ftl

```

1 <line x1="${origin.x}"
2      y1="${origin.y}"
3      x2="${end.x}"
4      y2="${end.y}"
5      stroke-width="${strokeWidth}"
6      <#if strokeColor??>stroke="${strokeColor}"</#if>/>

```

Listing 26: rect.ftl

```

1 <rect x="${origin.x}"
2      y="${origin.y}"
3      width="${width}"
4      height="${height}"
5      stroke-width="${strokeWidth}"
6      <#if strokeColor??>stroke="${strokeColor}"</#if>
7      <#if fillColor??>fill="${fillColor}"</#if>/>

```

Listing 27: text.ftl

```

1 <text x="${origin.x}"
2      y="${origin.y}"
3      font-family="${fontFamily}"
4      font-size="${fontSize}"
5      stroke-width="${strokeWidth}"
6      <#if fillColor??>fill="${fillColor}"</#if>
7      <#if strokeColor??>stroke="${strokeColor}"</#if>>
8   ${text}
9 </text>

```

### 3.3 Tests

Dieser Abschnitt beinhaltet die generierten SVG-Dateien und die dazugehörigen Bilder.

Listing 28: boxes.svg

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <!DOCTYPE svg>
3 <svg xmlns="http://www.w3.org/2000/svg"
4     width="500"
5     height="400"
6     >
7
8
9     <rect x="0"
10        y="0"

```

## Übung 3

```
11     width="500"  
12     height="100"  
13     stroke-width="1"  
14  
15     fill="#000000"/>  
16 <rect x="0"  
17     y="100"  
18     width="500"  
19     height="100"  
20     stroke-width="1"  
21  
22     fill="#444444"/>  
23 <rect x="0"  
24     y="200"  
25     width="500"  
26     height="100"  
27     stroke-width="1"  
28  
29     fill="#888888"/>  
30 <rect x="0"  
31     y="300"  
32     width="500"  
33     height="100"  
34     stroke-width="1"  
35  
36     fill="#CCCCC"/>  
37  
38 </svg>
```



Abbildung 3: Boxes

Listing 29: points.svg

## Übung 3

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <!DOCTYPE svg>
3  <svg xmlns="http://www.w3.org/2000/svg"
4      width="500"
5      height="400"
6      viewBox="-3 -3 6 6"
7      preserveAspectRatio="none">
8
9      <line x1="-3" x2="3" y1="0" y2="0" stroke="#555" stroke-width="0.01" />
10     <line x1="0" x2="0" y1="-3" y2="3" stroke="#555" stroke-width="0.01" />
11
12     <circle cx="-2"
13           cy="2"
14           r="0.05"
15           stroke-width="0.01"
16
17           fill="#000000"/>
18     <circle cx="-1"
19           cy="-1"
20           r="0.05"
21           stroke-width="0.01"
22
23           fill="#000000"/>
24     <circle cx="-0.5"
25           cy="-0.5"
26           r="0.05"
27           stroke-width="0.01"
28
29           fill="#000000"/>
30     <circle cx="-0.25"
31           cy="0.25"
32           r="0.05"
33           stroke-width="0.01"
34
35           fill="#000000"/>
36     <circle cx="0"
37           cy="-1.5"
38           r="0.05"
39           stroke-width="0.01"
40
41           fill="#000000"/>
42     <circle cx="0"
43           cy="1.5"
44           r="0.05"
45           stroke-width="0.01"
46
47           fill="#000000"/>
48     <circle cx="0.25"
49           cy="-0.25"
50           r="0.05"
51           stroke-width="0.01"
52
53           fill="#000000"/>
54     <circle cx="0.5"
55           cy="0.5"
56           r="0.05"
57           stroke-width="0.01"
58
59           fill="#000000"/>
60     <circle cx="1"
61           cy="1"
62           r="0.05"
```



## Übung 3

```

63     stroke-width="0.01"
64
65     fill="#000000"/>
66 <circle cx="2"
67       cy="-2"
68       r="0.05"
69       stroke-width="0.01"
70
71       fill="#000000"/>
72
73 </svg>

```

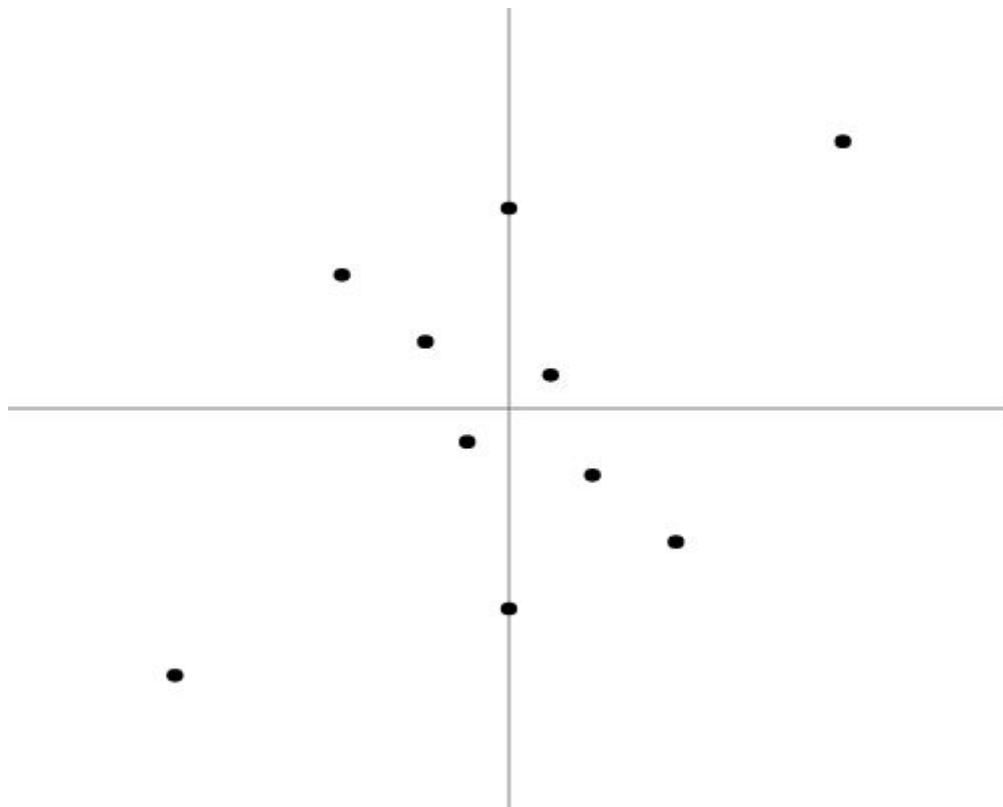


Abbildung 4: *Points*

Listing 30: lines.svg

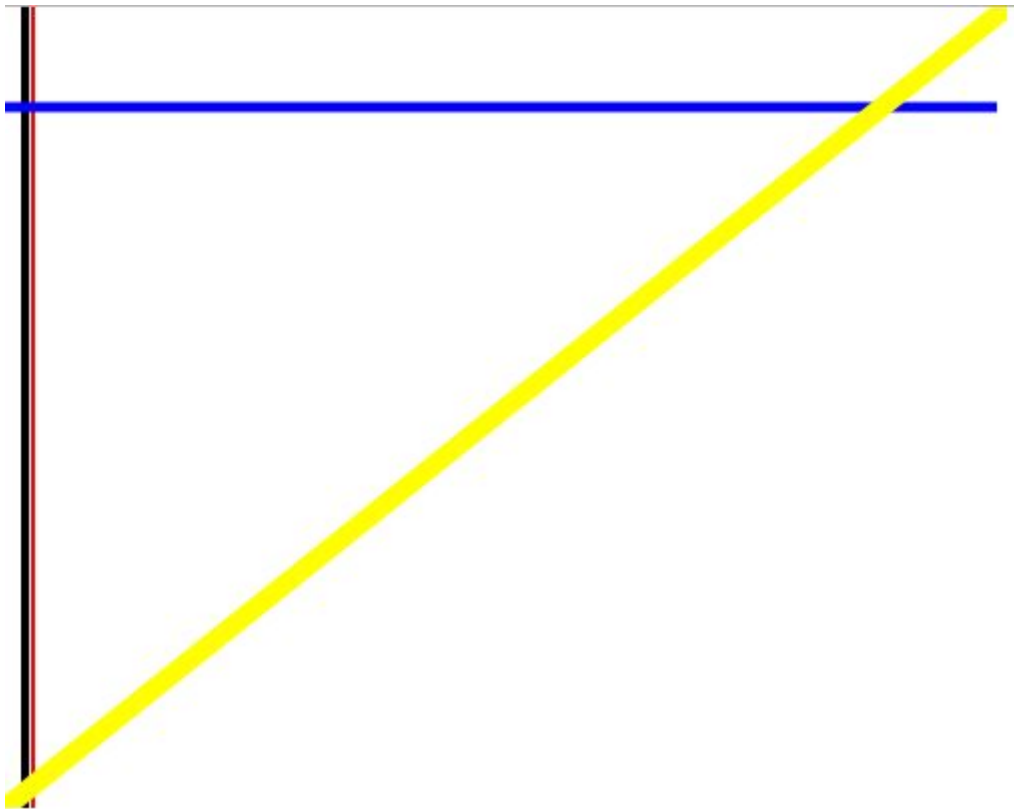
```

1  <?xml version="1.0" encoding="utf-8"?>
2  <!DOCTYPE svg>
3  <svg xmlns="http://www.w3.org/2000/svg"
4      width="500"
5      height="400"
6      >
7
8
9  <line x1="10"
10      y1="0"
11      x2="10"
12      y2="400"
13      stroke-width="4"
14      stroke="#000000"/>
15  <line x1="14"
16      y1="0"

```

## Übung 3

```
17     x2="14"  
18     y2="500"  
19     stroke-width="2"  
20     stroke="#FF0000"/>  
21 <line x1="0"  
22     y1="50"  
23     x2="495"  
24     y2="50"  
25     stroke-width="5"  
26     stroke="#0000FF"/>  
27 <line x1="500"  
28     y1="0"  
29     x2="0"  
30     y2="400"  
31     stroke-width="10"  
32     stroke="#FFFF00"/>  
33  
34 </svg>
```

Abbildung 5: *Lines*

Listing 31: all.svg

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <!DOCTYPE svg>
3  <svg xmlns="http://www.w3.org/2000/svg"
4      width="500"
5      height="400"
6      viewBox="-3 -3 6 6"
7      preserveAspectRatio="none">
8
9      <line x1="-3" x2="3" y1="0" y2="0" stroke="#555" stroke-width="0.01" />
10     <line x1="0" x2="0" y1="-3" y2="3" stroke="#555" stroke-width="0.01" />
11
12     <rect x="-2.8"
13         y="-3"
14         width="0.1"
15         height="3"
16         stroke-width="0.01"
17         stroke="#FF0000"
18         fill="#000000"/>
19     <rect x="-2"
20         y="-1.5"
21         width="0.1"
22         height="1.5"
23         stroke-width="0.01"
24         stroke="#00FF00"
25         fill="#FFFF00"/>
26     <rect x="-1"
27         y="-1.2"
28         width="0.1"
29         height="1.2"
30         stroke-width="0.01"
31         stroke="#000000"
32         fill="#000000"/>
33     <rect x="-0.5"
34         y="-0.8"
35         width="0.1"
36         height="0.8"
37         stroke-width="0.01"
38         stroke="#0000FF"
39         fill="#000000"/>
40     <circle cx="-0.5"
41         cy="0.5"
42         r="0.05"
43         stroke-width="0.01"
44         fill="#000000"/>
45     <circle cx="-1"
46         cy="1"
47         r="0.05"
48         stroke-width="0.01"
49         fill="#FF0000"/>
50     <circle cx="-2"
51         cy="1"
52         r="0.05"
53         stroke-width="0.01"
54         fill="#00FF00"/>
55     <circle cx="-2.5"
56         cy="2.8"
57         r="0.05"
58         stroke-width="0.01"
59         fill="#00FF00"/>
60     <circle cx="-2.5"
61         cy="2.8"
62         r="0.05"
63         stroke-width="0.01"
64         fill="#00FF00"/>
```

## Übung 3

```
61         stroke-width="0.01"
62
63         fill="#FFFF00"/>
64 <circle cx="-3"
65         cy="3"
66         r="0.05"
67         stroke-width="0.01"
68
69         fill="#000000"/>
70 <text x="0"
71     y="-2"
72     font-family="Verdana"
73     font-size="0.5"
74     stroke-width="0.01"
75     fill="#C0C0C0"
76     stroke="#000000">
77 My Text
78 </text>
79 <text x="0"
80     y="-1"
81     font-family="Verdana"
82     font-size="0.5"
83     stroke-width="0.01"
84     fill="#C0C0C0"
85     stroke="#000000">
86 is here
87 </text>
88 <line x1="0"
89     y1="0"
90     x2="2"
91     y2="2"
92     stroke-width="0.1"
93     stroke="#000000"/>
94 <line x1="1"
95     y1="0"
96     x2="1"
97     y2="2"
98     stroke-width="0.05"
99     stroke="#FF0000"/>
100
101 </svg>
```

Abbildung 6: *All*