hochschule mannheim

Bachelor-Studiengang Informatik
Übungen zur Vorlesung “Grundlagen der Informatik” (GDI), SS 201X/201X

# Übungsblatt ≈10: Interfaces, Klassen, Vererbung

Ein Kunde Ihrer Firma involviert Sie in ein Projekt, das zum Ziel hat ein bestehendes Grafikprogramm mit der Möglichkeit zur Verarbeitung von Vektorgrafiken im SVG-Format zu erweitern. Ihre Aufgabe ist es eine Konsolenanwendung zu erstellen, mit der die Testabteilung der Kundenfirma komfortabel einfache SVG-Grafiken über ein Konsolen-Menü erstellen kann.

Da Sie nicht die einzige Arbeitsgruppe am Projekt sind, der Kunde aber nicht möchte, dass die verschiedenen Teams aneinander vorbei arbeiten, stellt er Ihnen die folgenden Interfaces und Datentypen zur Verfügung (Siehe Anhang), die er bereits in einer Testversion verwendet (D.h. Sie dürfen diese Interfaces unter keinen Umständen verändern, sonst sucht sich der Kunde eine neue Firma!).

Da der das Programm zum Ansehen der SVG-Dateien noch nicht fertiggestellt ist, empfiehlt er Ihnen während der Entwicklung auf einen SVG-fähigen Browser zurückzugreifen. Außerdem gibt er Ihnen den Hinweis sich auf http://www.w3schools.com/svg/svg_examples.asp Beispiele von SVG-Dateien anzueignen.

## Aufgabe 1:

Um Ihrer Abteilung die Problemstellung klar zu machen, fassen Sie in ca. 10 Sätzen zusammen, um was es sich bei dem SVG-Format genau handelt und auf welches Datenformat es aufbaut, bzw. wie es seine Bildinformationen speichert. Beschreiben Sie auch die Unterschiede zu Ihnen bereits bekannten Bildformaten.

## Aufgabe 2:

Der Kunde ist sich noch nicht sicher, welche Funktionen das „ISVGElement“-Interface benötigt (deshalb ist es noch leer) und legt die Denkarbeit in die Hände der verschiedenen Entwicklungsteams. Überlegen Sie, ob und welche Interface-Funktionen für eine Lösung Ihrer Aufgabe nötig sind. Der Kunde empfiehlt Ihnen das Interface schon zu benutzen, egal welche Entscheidung Sie treffen.

*Hinweis: Das ISVGElement-Interface erbt bereits vom IXMLElement-Interface.*

## Aufgabe 3:

Stellen Sie den Kunden zufrieden, indem Sie eine menüorientierte Konsolenanwendung implementieren, mit der er folgende Aufgaben erledigen kann:

- komfortabel Kreise und Rechtecke durch Konsoleneingabe zu einem neuen SVG-Dokument hinzufügen kann.
    - Die Auswahl ob Kreis oder Rechteck, soll in einem Untermenü auswählbar sein.
- sich alle im Dokument existierenden Elemente anzeigen lassen kann.
- existierende Elemente aus dem Dokument löschen kann.
- das Dokument unter einem anzugebenden Dateinamen speichern kann.

*Hinweis: Das Laden einer SVG-Datei ist nicht gefordert!*

## Aufgabe 4:

Ihr Kunde möchte nun auch noch Linien mit mehreren Punkten („polyline“) mit Ihrer Anwendung erstellen. Überlegen Sie ob es sinnvoll ist, aus den 3 Klassen(Rechteck, Kreis, Linie), die nun ISVGElement implementieren, eine Oberklasse zu extrahieren um doppelten Quellcode (falls vorhanden) zu vermeiden.

## Aufgabe 5:

Ein anderes Entwicklungsteam, das an einem graphischen Editor arbeitet, hat beim Kunden folgende Funktionen für das ISVGElement-Interface durchgesetzt, damit es komfortabel den Style eines Elementes ändern kann.

```
/**
 * @param pStyleString For example: <code>&lt;polygon points="..."<br/>
 *       style="<u><b>fill:#cccccc; stroke:#000000;stroke-width:1</b></u>"/&gt;</code>
 */
public void setStyle(final String pStyleString);

/**
 * @return the styleString previously set by
 * {@link ISVGElement#setStyle(String)} or <code>""</code> as default.
 */
public String getStyle();
```

Fügen Sie diese Funktionen in Ihr bestehendes Interface ein und führen Sie alle nötigen Änderungen durch.

Erweitern Sie ihr Programm so, dass der Benutzer beim Erstellen auf der Konsole einen beliebigen Style angeben kann. Fügen Sie außerdem einen neuen Menüpunkt an, mit dem sich der Style eines Objektes im Nachhinein, überschreiben lassen kann.

## Interfaces

```
public interface IXMLElement {
        /**
         * @return Valid XML.
         * I.e.: <u><b><code>&lt;node attribute="value"&gt;something&lt;/node&gt;</code></b></u>.
         */
        public String toXML();

        /**
         * @return the name of the XMLElement,
         * I.e.: <code>&lt;<u><b>node</b></u> attribute="value"&gt;something&lt;/node&gt;</code>.
         */
        public String getTagName();

        /**
         * @return an Array of the XML-Attributes,
         * I.e.: <code>&lt;node <u><b>attribute="value"</b></u>&gt;something&lt;/node&gt;</code>.
         */
        public XMLAttribute[] getAttributes();
}
```

```
import java.util.List;

public interface ISVGDocument {
        /**
         * @return a list of all Elements.
         */
        public abstract List<ISVGElement> getElements();

        /**
         * @param pSVGElement a new element to be added to the list of existing elements.
         */
        public abstract void addElement(final ISVGElement pSVGElement);
}
```

```
public interface ISVGElement extends IXMLElement {

}
```

```
public interface IXMLAttribute {

        /**
         * @param the name of this attribute.
         */
        public String getName();

        /**
         * @param the value of this attribute.
         */
        public String getValue();

        /**
         * @param pValue save a new value to this attribute..
         */
        public void setValue(final String pValue);
}
```