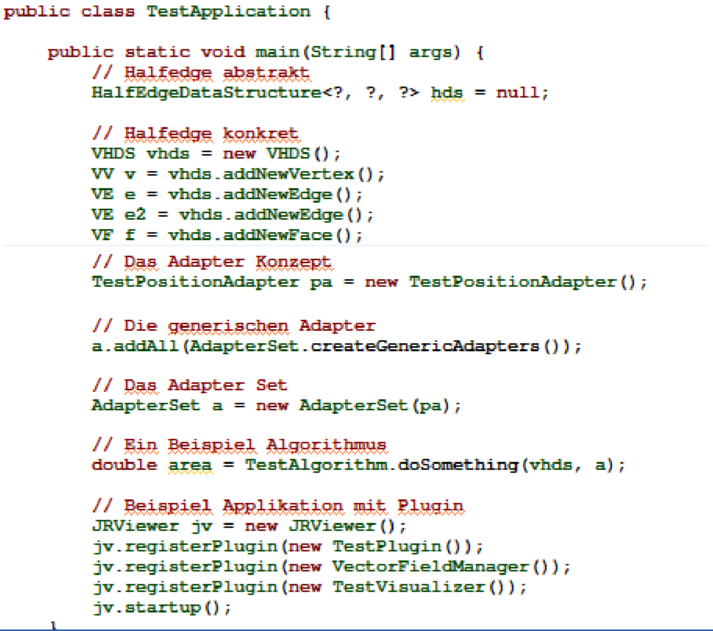
Halfedge!

Stefan Sechelmann

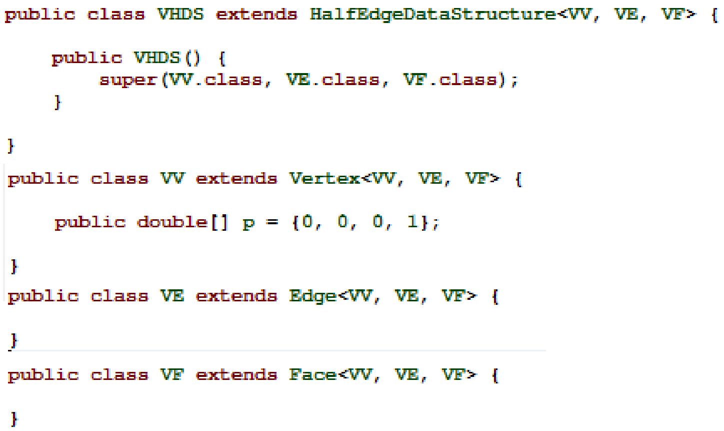
DFG-Forschungszentrum MATHEON
Mathematik für Schlüsseltechnologien

Plan



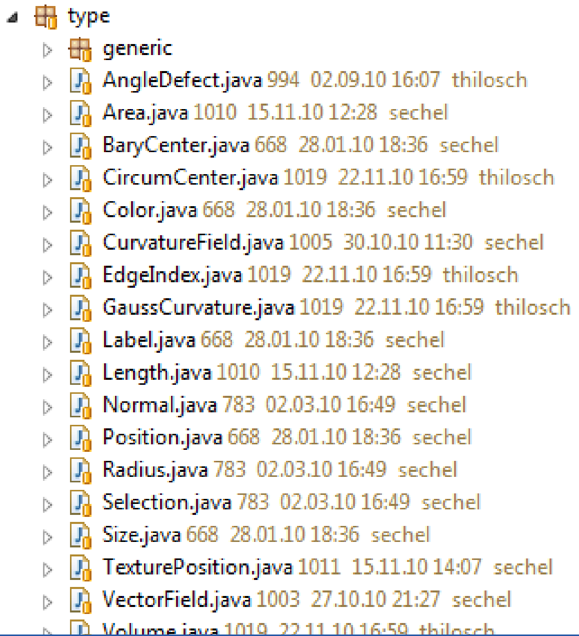
```
public class TestApplication {

    public static void main(String[] args) {
        // Halfedge abstrakt
        HalfEdgeDataStructure<?, ?, ?> hds = null;

        // Halfedge konkret
        VHDS vhds = new VHDS();
        VV v = vhds.addNewVertex();
        VE e = vhds.addNewEdge();
        VE e2 = vhds.addNewEdge();
        VF f = vhds.addNewFace();
        // Das Adapter Konzept
        TestPositionAdapter pa = new TestPositionAdapter();

        // Die generischen Adapter
        a.addAll(AdapterSet.createGenericAdapters());

        // Das Adapter Set
        AdapterSet a = new AdapterSet(pa);

        // Ein Beispiel Algorithmus
        double area = TestAlgorithm.doSomething(vhds, a);

        // Beispiel Applikation mit Plugin
        JRViewer jv = new JRViewer();
        jv.registerPlugin(new TestPlugin());
        jv.registerPlugin(new VectorFieldManager());
        jv.registerPlugin(new TestVisualizer());
        jv.startup();
```

Halfedege konkret



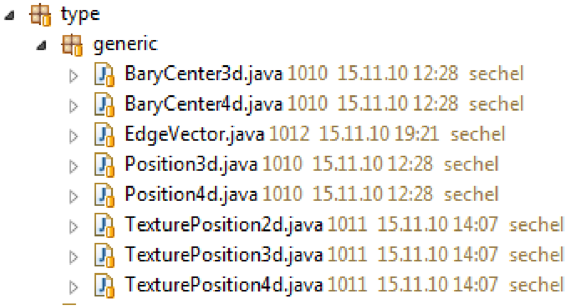
```
public class VHDS extends HalfEdgeDataStructure<VV, VE, VF> {

    public VHDS() {
        super(VV.class, VE.class, VF.class);
    }

}
public class VV extends Vertex<VV, VE, VF> {

    public double[] p = {0, 0, 0, 1};

}
public class VE extends Edge<VV, VE, VF> {

}
public class VF extends Face<VV, VE, VF> {

}
```

Adapter Types



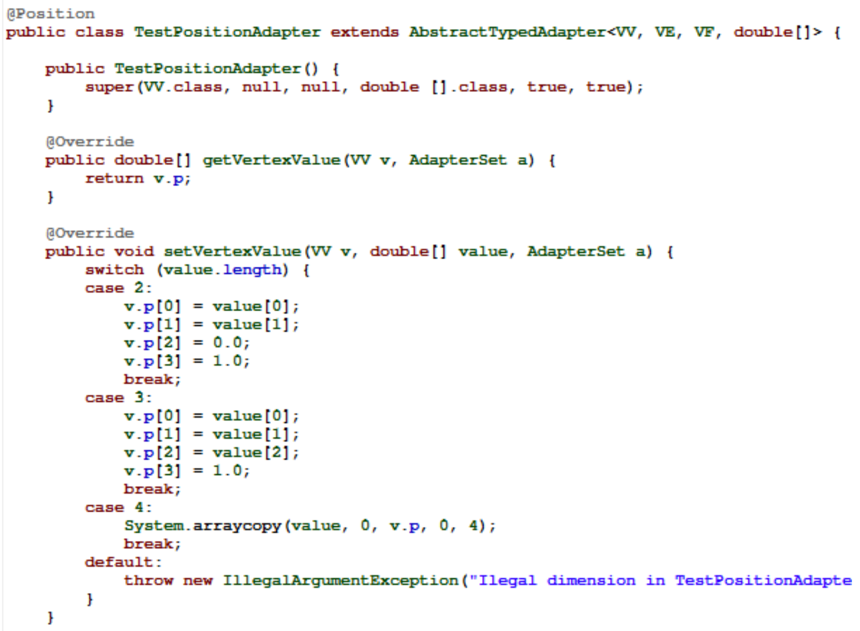
```
type
  generic
  AngleDefect.java 994  02.09.10 16:07  thilosch
  Area.java 1010  15.11.10 12:28  sechel
  BaryCenter.java 668  28.01.10 18:36  sechel
  CircumCenter.java 1019  22.11.10 16:59  thilosch
  Color.java 668  28.01.10 18:36  sechel
  CurvatureField.java 1005  30.10.10 11:30  sechel
  EdgeIndex.java 1019  22.11.10 16:59  thilosch
  GaussCurvature.java 1019  22.11.10 16:59  thilosch
  Label.java 668  28.01.10 18:36  sechel
  Length.java 1010  15.11.10 12:28  sechel
  Normal.java 783  02.03.10 16:49  sechel
  Position.java 668  28.01.10 18:36  sechel
  Radius.java 783  02.03.10 16:49  sechel
  Selection.java 783  02.03.10 16:49  sechel
  Size.java 668  28.01.10 18:36  sechel
  TexturePosition.java 1011  15.11.10 14:07  sechel
  VectorField.java 1003  27.10.10 21:27  sechel
  Volume.java 1019  22.11.10 16:59  thilosch
```

Generic Adapter



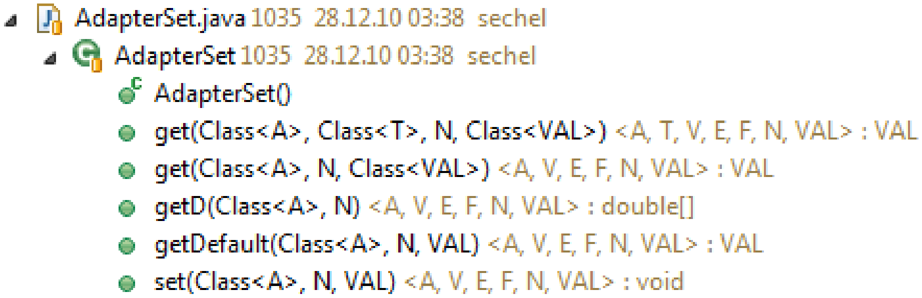
```
type
  generic
    BaryCenter3d.java 1010  15.11.10 12:28  sechel
    BaryCenter4d.java 1010  15.11.10 12:28  sechel
    EdgeVector.java 1012  15.11.10 19:21  sechel
    Position3d.java 1010  15.11.10 12:28  sechel
    Position4d.java 1010  15.11.10 12:28  sechel
    TexturePosition2d.java 1011  15.11.10 14:07  sechel
    TexturePosition3d.java 1011  15.11.10 14:07  sechel
    TexturePosition4d.java 1011  15.11.10 14:07  sechel
```

Position Adapter

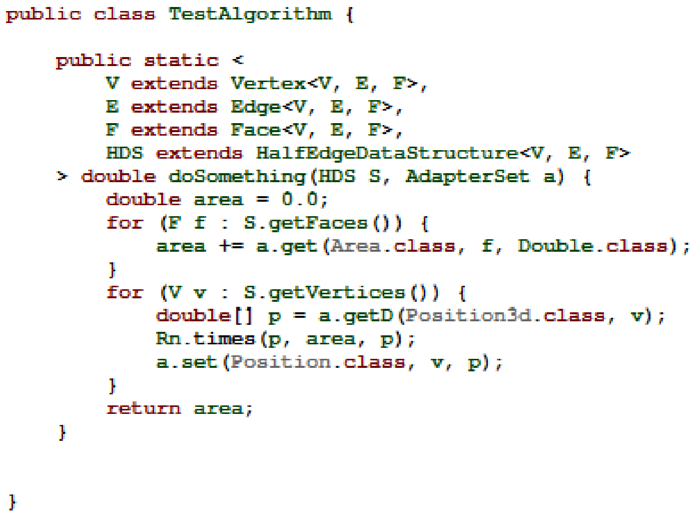


```
@Position
public class TestPositionAdapter extends AbstractTypedAdapter<VV, VE, VF, double[]> {

    public TestPositionAdapter() {
        super(VV.class, null, null, double [].class, true, true);
    }

    @Override
    public double[] getVertexValue(VV v, AdapterSet a) {
        return v.p;
    }

    @Override
    public void setVertexValue(VV v, double[] value, AdapterSet a) {
        switch (value.length) {
        case 2:
            v.p[0] = value[0];
            v.p[1] = value[1];
            v.p[2] = 0.0;
            v.p[3] = 1.0;
            break;
        case 3:
            v.p[0] = value[0];
            v.p[1] = value[1];
            v.p[2] = value[2];
            v.p[3] = 1.0;
            break;
        case 4:
            System.arraycopy(value, 0, v.p, 0, 4);
            break;
        default:
            throw new IllegalArgumentException("Ilegal dimension in TestPositionAdapte
        }
    }
```

AdapterSet

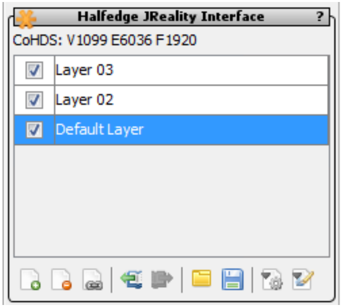

```
AdapterSet.java 1035  28.12.10 03:38  sechel
  AdapterSet 1035  28.12.10 03:38  sechel
    AdapterSet()
    get(Class<A>, Class<T>, N, Class<VAL>) <A, T, V, E, F, N, VAL> : VAL
    get(Class<A>, N, Class<VAL>) <A, V, E, F, N, VAL> : VAL
    getD(Class<A>, N) <A, V, E, F, N, VAL> : double[]
    getDefault(Class<A>, N, VAL) <A, V, E, F, N, VAL> : VAL
    set(Class<A>, N, VAL) <A, V, E, F, N, VAL> : void
```

Algorithm



```
public class TestAlgorithm {

    public static <
        V extends Vertex<V, E, F>,
        E extends Edge<V, E, F>,
        F extends Face<V, E, F>,
        HDS extends HalfEdgeDataStructure<V, E, F>
    > double doSomething(HDS S, AdapterSet a) {
        double area = 0.0;
        for (F f : S.getFaces()) {
            area += a.get(Area.class, f, Double.class);
        }
        for (V v : S.getVertices()) {
            double[] p = a.getD(Position3d.class, v);
            Rn.times(p, area, p);
            a.set(Position.class, v, p);
        }
        return area;
    }


}
```

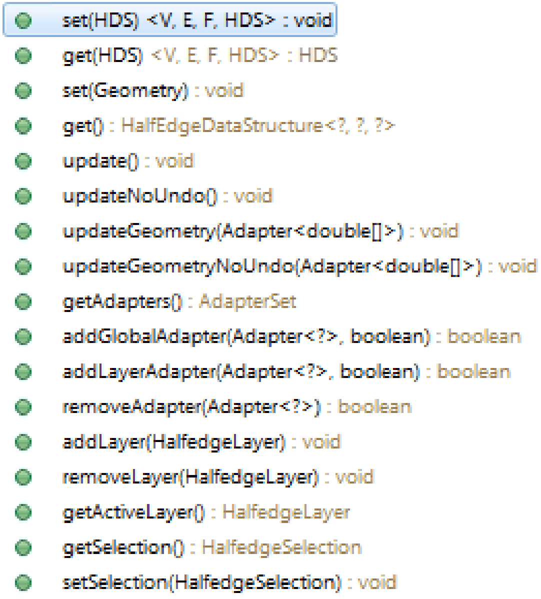
The Halfedge Interface



Halfedge JReality Interface

CoHDS: V1099 E6036 F1920

- ☑ Layer 03
- ☑ Layer 02
- ☑ Default Layer

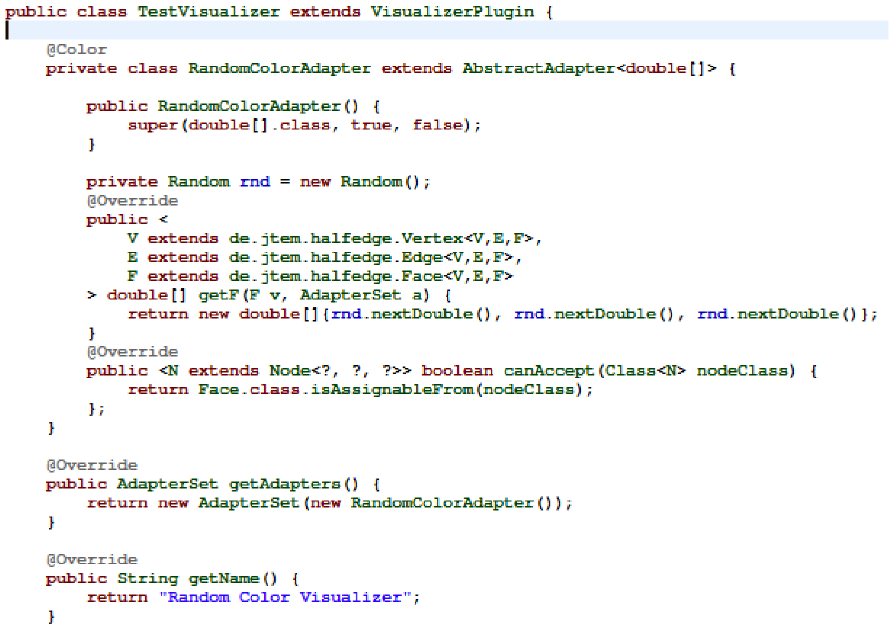


```
set(HDS) <V, E, F, HDS> : void
get(HDS) <V, E, F, HDS> : HDS
set(Geometry) : void
get() : HalfEdgeDataStructure<?, ?, ?>
update() : void
updateNoUndo() : void
updateGeometry(Adapter<double[]>) : void
updateGeometryNoUndo(Adapter<double[]>) : void
getAdapters() : AdapterSet
addGlobalAdapter(Adapter<?>, boolean) : boolean
addLayerAdapter(Adapter<?>, boolean) : boolean
removeAdapter(Adapter<?>) : boolean
addLayer(HalfedgeLayer) : void
removeLayer(HalfedgeLayer) : void
getActiveLayer() : HalfedgeLayer
getSelection() : HalfedgeSelection
setSelection(HalfedgeSelection) : void
```

A Halfedge Plugin

```
public class TestPlugin extends ShrinkPanelPlugin implements ActionListener {

    private HalfedgeInterface
        hif = null;
    private JButton
        button = new JButton("Go");

    public TestPlugin() {
        shrinkPanel.add(button);
        button.addActionListener(this);
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        VHDS hds = hif.get(new VHDS());
        System.out.println("Got data structure:\n" + hds);
        double area = TestAlgorithm.doSomething(hds, hif.getAdapters());
        System.out.println("Area is " + area);
        hif.addGlobalAdapter(new TestVectorField(), false);
        hif.update();
    }

    @Override
    public void install(Controller c) throws Exception {
        super.install(c);
        hif = c.getPlugin(HalfedgeInterface.class);
        hif.addGlobalAdapter(new TestPositionAdapter(), true);
    }

    @Override
```

A Visualizer



```
public class TestVisualizer extends VisualizerPlugin {

    @Color
    private class RandomColorAdapter extends AbstractAdapter<double[]> {

        public RandomColorAdapter() {
            super(double[].class, true, false);
        }

        private Random rnd = new Random();
        @Override
        public <
            V extends de.jtem.halfedge.Vertex<V,E,F>,
            E extends de.jtem.halfedge.Edge<V,E,F>,
            F extends de.jtem.halfedge.Face<V,E,F>
        > double[] getF(F v, AdapterSet a) {
            return new double[]{rnd.nextDouble(), rnd.nextDouble(), rnd.nextDouble()};
        }
        @Override
        public <N extends Node<?, ?, ?>> boolean canAccept(Class<N> nodeClass) {
            return Face.class.isAssignableFrom(nodeClass);
        };
    }

    @Override
    public AdapterSet getAdapters() {
        return new AdapterSet(new RandomColorAdapter());
    }

    @Override
    public String getName() {
        return "Random Color Visualizer";
    }
```

A Vector Field

```java
@VectorField
public class TestVectorField extends AbstractTypedAdapter<VV, VE, VF, double[]>

    public TestVectorField() {
        super(VV.class, null, null, double[].class, true, false);
    }

    @Override
    public double[] getVertexValue(VV v, AdapterSet a) {
        return a.getD(Position3d.class, v);
    }

}
```

A Scalar Function

```
public class TestScalarFunction extends AbstractAdapter<Double> {

    private Random
        rnd = new Random();

    public TestScalarFunction() {
        super(Double.class, true, false);
    }

    @Override
    public <
        V extends Vertex<V, E, F>,
        E extends Edge<V, E, F>,
        F extends Face<V, E, F>
    > Double getF(F f, AdapterSet a) {
        return rnd.nextGaussian();
    }

    @Override
    public <N extends Node<?, ?, ?>> boolean canAccept(Class<N> nodeClass) {
        return Face.class.isAssignableFrom(nodeClass);
    }

}
```