Creating Features and Feature types

As you can see in the diagram, features, feature types and attribute types are just interfaces.

This means that you cannot instantiate them directly, but also that you can provide your own specialized versions that wrap legacy objects you may have in your own application.

You can create a Feature by hand with the following steps:

- create an array of attribute types, describing the attributes in your feature type, using the AttributeTypeFactory static methods;
- create a FeatureType out of the attribute types using the FeatureTypeFactory static methods;
- create the set of attribute values that describe your specific feature;
- create the Feature object by using the FeatureType as a factory.

The following code snippet illustrates this process:

```
//AttributeType geom =
AttributeTypeFactory.newAttributeType("the_g
eom", Point.class);
AttributeType geom =
AttributeTypeFactory.newAttributeType("the_g
eom", LineString.class);
AttributeType roadWidth =
AttributeTypeFactory.newAttributeType("width
", Float.class);
FeatureType ftRoad =
FeatureTypeFactory.newFeatureType(new
AttributeType[] {geom, roadWidth}, "road");

WKTReader wktReader = new WKTReader();
//Point geometry = (Point)
wktReader.read("POINT (" + lat + " " + lon +
")");
LineString geometry = (LineString)
wktReader.read("LINESTRING (0 0, 10 10)");
Float width = new Float(10);
Feature theRoad = ftRoad.create(new Object[]
{geometry, width}, "myRoad");

System.out.println(theRoad);
```

Of course, you won't usually be involved in feature creation (that's the work of DataStores) but it's nice to know how to do it by hand, to have a better understanding of how things work.

Features are usually accessed from DataStores in a streaming way. Should you need to handle a set of Features in memory, have a look at the FeatureCollection class, which is a specialized Set designed to hold features and provides a convenient FeatureIterator that doesn't require you to use casts when calling the next() method.