Graph Serialization and Export/Import

The default graph implementations provided by JGraphT are serializable as long as you choose vertex and edge types which are themselves serializable.

Serialization is a convenient way to store a graph instance as binary data, but the format is not human-readable, and we don't make any guarantee of serialization compatibility across JGraphT versions. (In other words, if you serialize a graph with version X, and then attempt to deserialize it with version X+1, an exception may be thrown.)

To address this, JGraphT provides package org.jgrapht.io for exporting and importing graphs in a variety of standard formats. These can also be used for data interchange with other applications.

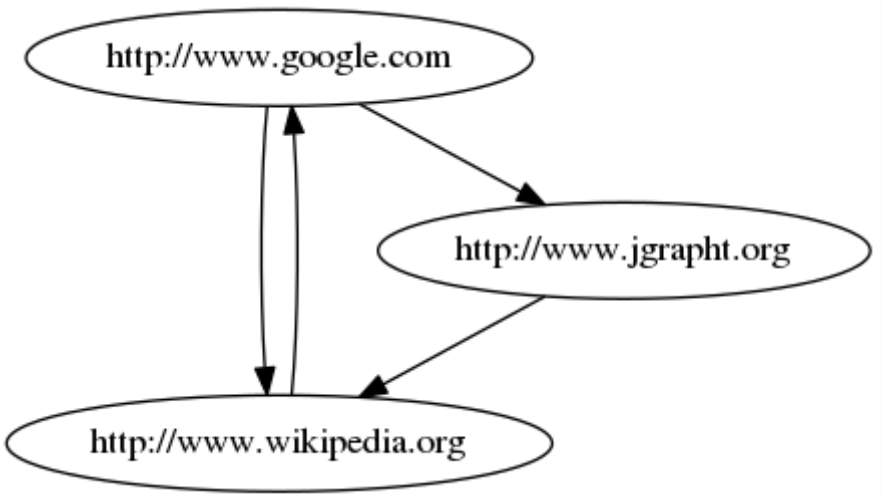
Continuing our HelloJGraphT example, here's how to export a graph in GraphViz .dot format:

```
        DOTExporter<URI, DefaultEdge> exporter =
            new DOTExporter<>(v -> v.getHost().replace('.', '_'));
        exporter.setVertexAttributeProvider((v) -> {
            Map<String, Attribute> map = new LinkedHashMap<>();
            map.put("label", DefaultAttribute.createAttribute(v.toString()));
            return map;
        });
        Writer writer = new StringWriter();
        exporter.exportGraph(hrefGraph, writer);
        System.out.println(writer.toString());
```

with expected output

```
strict digraph G {
  www_google_com [ label="http://www.google.com" ];
  www_wikipedia_org [ label="http://www.wikipedia.org" ];
  www_jgrapht_org [ label="http://www.jgrapht.org" ];
  www_jgrapht_org -> www_wikipedia_org;
  www_google_com -> www_jgrapht_org;
  www_google_com -> www_wikipedia_org;
  www_wikipedia_org -> www_google_com;
}
```

which GraphViz renders as:



If you just want a quick dump of the structure of a small graph, you can also use the `toString` method; here's another example from the HelloJGraphT demo:

```
        System.out.println(stringGraph.toString());
```

which produces

```
([v1, v2, v3, v4], [{v1,v2}, {v2,v3}, {v3,v4}, {v4,v1}])
```

First comes the vertex set, followed by the edge set. Directed edges are rendered with round brackets, whereas undirected edges are rendered with curly brackets. Custom edge attributes are not rendered. If you want a nicer rendering, you can override toStringFromSets in your graph implementation, but you're probably better off using one of the exporters instead.