SINBAD - Quick Reference - JAVA

http://cs.berry.edu/sinbad

Basic Template

```
import core.data.*
...
DataSource ds = DataSource.connect("<URL>");
ds.load();
...
ds.fetch...("<field>");  // see fetch...() methods
```

For Processing

```
import core.data.*;
...
void setup() {
    DataSource.initializeProcessing(this);
    ...
} Then use `connect`, `load`, and `fetch...` as above.
```

Examining Available Data

```
ds.printUsageString()
```

Test if field paths valid:

```
ds.hasFields("...", ...)
```

Array (or list) of available top-level field names:

```
String[] fields = ds.fieldNames();
// or
List<String> fields = ds.fieldNamesList();
```

Other Connection Methods

Specify a data format ("CSV", "XML", "JSON"):

```
DataSource ds = DataSource.connectAs("<FORMAT>", "<URL>");
```

Connect using a data specification file:

```
DataSource ds = DataSource.connectUsing("<URL>");
```

Connection (URL) Parameters

Some data sources may require additional parameters to construct the URL. After the `connect` and before `load`:

```
ds.setParam("<name>", "<value>");
```

Data Format Options

Some data sources provide post-processing options to manipulate the data once it has been downloaded. The available options are format-specific and are listed by enabling verbose usage info:

```
ds.printUsageString(true);
```

Use

```
ds.setOption("<name>", "<value>");
```

For example (with a CSV data source):

```
ds.setOption("header", "ID,Name,Call
sign,Country,Active");
```

Selecting from .zip archive

```
ds.setOption("file-entry", "FACTDATA_MAR2016.TXT");
```

Cache Control

Control frequency of caching (or disable it):

```
ds.setCacheTimeout(<seconds>);
// may use  CacheConstants.NEVER_CACHE
//      or  CacheConstants.NEVER_RELOAD (always caches)
```

Show where files are cached:

```
System.out.println(ds.getCacheDirectory());
```

Clear all cache files (for all data sources):

```
ds.clearENTIRECache();
```

Download Progress Display

(Dots that are printed as files are loaded)

```
DataSource.showDownloadProgress(boolean)
```

Note, this is a global setting and will apply to all data sources that are loaded after this statement has been executed.

View Preferences

```
DataSource.preferences();
```

When preferences are saved, the program will immediately terminate and exit. Comment out or delete the statement above to enable the program to continue running as usual.

Using an Iterator

```
DataSourceIterator iter = ds.iterator();
while (iter.hasData()) {
    String name = iter.fetchString("Name");
    boolean active = iter.fetchBoolean("Active");
    System.out.println(name + ": " + active);
    iter.loadNext();
}
```

Fetching Data

```
        // PRIMITIVE TYPE VALUES
public boolean fetchBoolean(String key);
public byte    fetchByte(String key);
public char    fetchChar(String key);
public double  fetchDouble(String key);
public float   fetchFloat(String key);
public int     fetchInt(String key);
public String  fetchString(String key);
        // ARRAYS
public boolean[] fetchBooleanArray(String key);
public byte[]    fetchByteArray(String key);
public char[]    fetchCharArray(String key);
public double[]  fetchDoubleArray(String key);
public float[]   fetchFloatArray(String key);
public int[]     fetchIntArray(String key);
public String[]  fetchStringArray(String key);
        // LISTS
public ArrayList<Boolean> fetchBooleanList(String key);
public ArrayList<Byte>    fetchByteList(String key);
public ArrayList<Character> fetchCharList(String key);
public ArrayList<Double>  fetchDoubleList(String key);
public ArrayList<Float>   fetchFloatList(String key);
public ArrayList<Integer> fetchIntList(String key);
public ArrayList<String>  fetchStringList(String key);
// OBJECTS (of any class you name - the order of key names
//          should match a constructor of the class)
public <T> T fetch(String clsName, String... keys);
public <T> ArrayList<T> fetchList(String clsName, String... keys);
public <T> T[] fetchArray(String clsName, String... keys);
```