A wish list for HDF-Java work

# General areas

## Performance improvement

* Have a more efficient way to handle multiple dimensional datasets. Currently if you pass an 2D+ data buffer to H5.H5Dread(), the performance will be very low because there are a lot of conversions between C array and Java array. To obtain high performance, users have to pass 1D array to H5.H5Dread() . Our goal is to revise the current implementation to improve the performance when a 2D+ buffer is used.
* Support large number of data elements. Java cannot handle arrays that have the number of data points larger than (2^31-1, Interger.MAX\_VALUE) since Java uses “int” type for array index. As 64-bit Java supports large memory, users may need to load a large dataset into memory. We need to figure out a solution for such a problem.
* Handle large dataset – due to the memory limitation of JVM or machine system, hdf-java will fail to open a large dataset when you try to load the whole dataset into memory. It would be better to have a buffering system that allows application to load part of the data into a buffer and automatically update the buffer as needed. This task can b very complicated.

## Better example code

* Revise current JNI example code and make it available at hdfgroup.org website.
* Add more example programs that show users how to use the object package

## Better documentation

* API descriptions and usage.
* Readme files on how to compile and run hdf-java.
* Instructions on how to build applications using hdf-java products.
* Updated user’s guide and online webpages.

## Better testing

* Unit tests for HDF5 1.6 version JNI.
* More testing cases for the object package.
* A standard list of test files.
* A list of regression tests for bug fixes.

# HDF5 1.8 support

## JNI support

The current release (version 2.6) of HDF-Java was built on HDF5 1.8.4 with 1.6 compatibility flag. Although it was built on 1.8, it does not include any new APIs. In order to use the new features or improvements added to HDF5 1.8 in other hdf-java products, the first thing is to add the new APIs in HDF5 JNI. This work was funded by one of our customers. To see the details, please read the RFC at <http://www.hdfgroup.uiuc.edu/RFC/HDF5/hdf-java/RFC_Support_HDF518_in_Java.pdf>

## Object package

Once the new APIs are added to HDF5 JNI, we will need to use the new APIs in the hdf-java package to take the advantage of the HDF5 1.8 library. A full RFC on how to support HDF5 1.8 in the hdf-java object package is in progress. Here we list just a few urgent needs:

* Replace all the deprecated APIs with the new APIs in HDF5 Java objects.
* Use more efficient new APIs. For instance, using the new H5Ocopy() function is more efficient than the current hdf-java copy() method.
* Redesign hdf-java object package to support new objects and new features added to HDF5 1.8.
* Investigate the compatibility issues between different version of HDF5 files and objects.