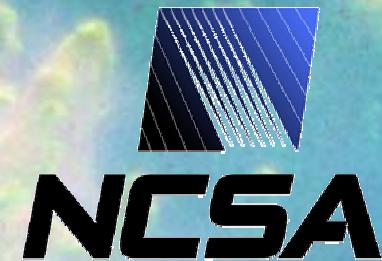
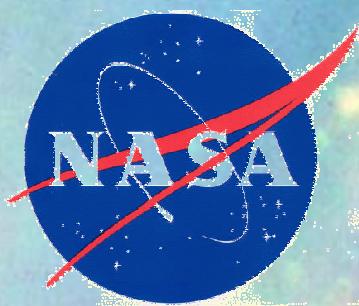




# Modular HDFView

*HDF and HDF-EOS Workshop VII, September 23-25, 2003*

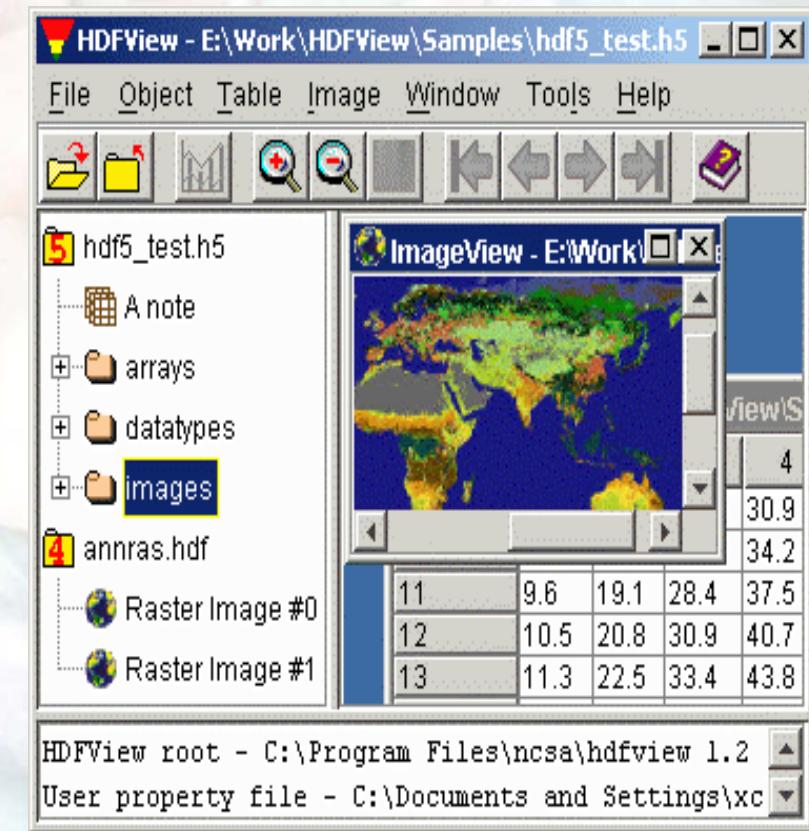


*This work is supported in part by a Cooperative Agreement with the National Aeronautics and Space Agency (NASA) and National Center for Supercomputing Applications (NCSA)*

# What Is HDFView?

**HDFView is a Java-based visual tool to browse and edit HDF4 and HDF5 files.**

**Starting with a tree view of a file hierarchy, you can descend through the hierarchy, navigate among the file's data objects, and open data as standard image, table, or text.**



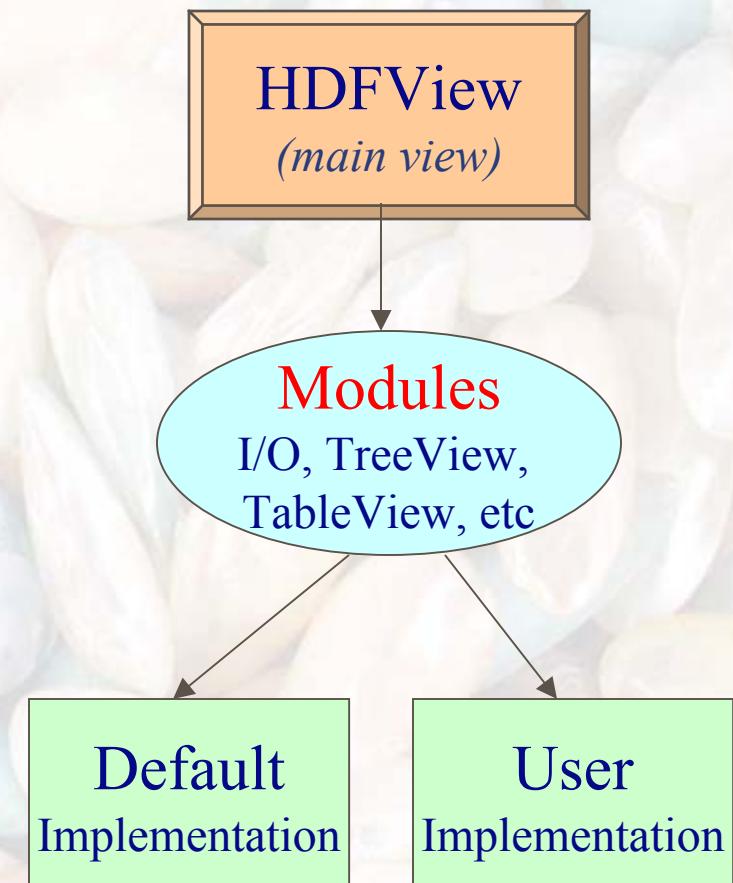
The current HDFView (version 1.3) is built as a integrated tool. It consists of standard TreeView, TableView, ImageView and MetadataView. These components can not be replaced.

# What Is Modular HDFView?

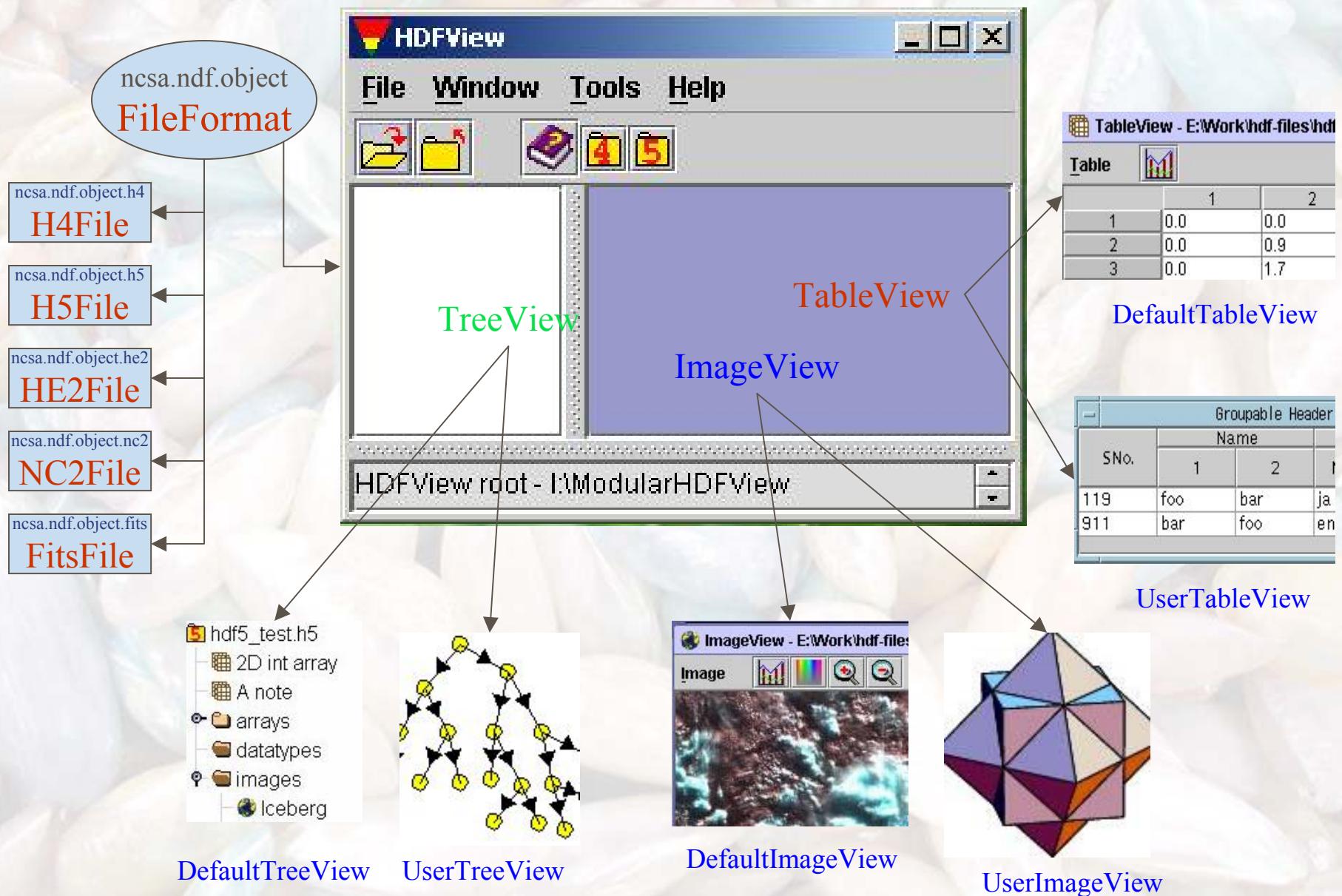
**Modular HDFView** is an improved HDFView where I/O and GUI components are replaceable modules.

**Replaceable modules include:**

- ✓ File I/O (file/data format)
- ✓ Tree view (show file structure)
- ✓ Table view (spreadsheet-like)
- ✓ Text view (view/edit for text dataset)
- ✓ Image view (view/process image)
- ✓ Palette view (view/change palette)
- ✓ Metadata (attribute) view



# Replaceable Modules



# Need for Modularization

- **Reuse source code:** users can extend their classes from common packages and abstract classes for less coding
- **Configurable installation:** users can choose to install HDF4 support or HDF5 support or both
- **Separation of file I/O and data viewer:** GUI components do not depend on file I/O implementation. Adding a new file format does not need to change any GUI components
- **Replaceable GUI modules:** users can implement their GUI components to replace the default TreeView, TabView, ImageView, and etc, which is intended for general purpose.
- **Easy to maintain:** replacing/changing one module does not change the rest of the source code

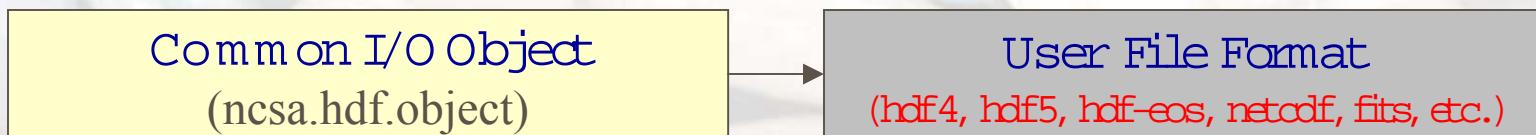
# Desired Features

- **Abstract interfaces/classes along with a default implementation:** the main view will only access to abstract interfaces/classes. It does not depend on user implementation
- **Dynamic loading user's modules:** HDFView automatically detect user's module packed in jar files
- **A mechanism for selecting which module to use:** when multiple modules provided for a data object, users can choose which module to use and set their default choice.
- **Extensive documentation and examples how to implement such a module**

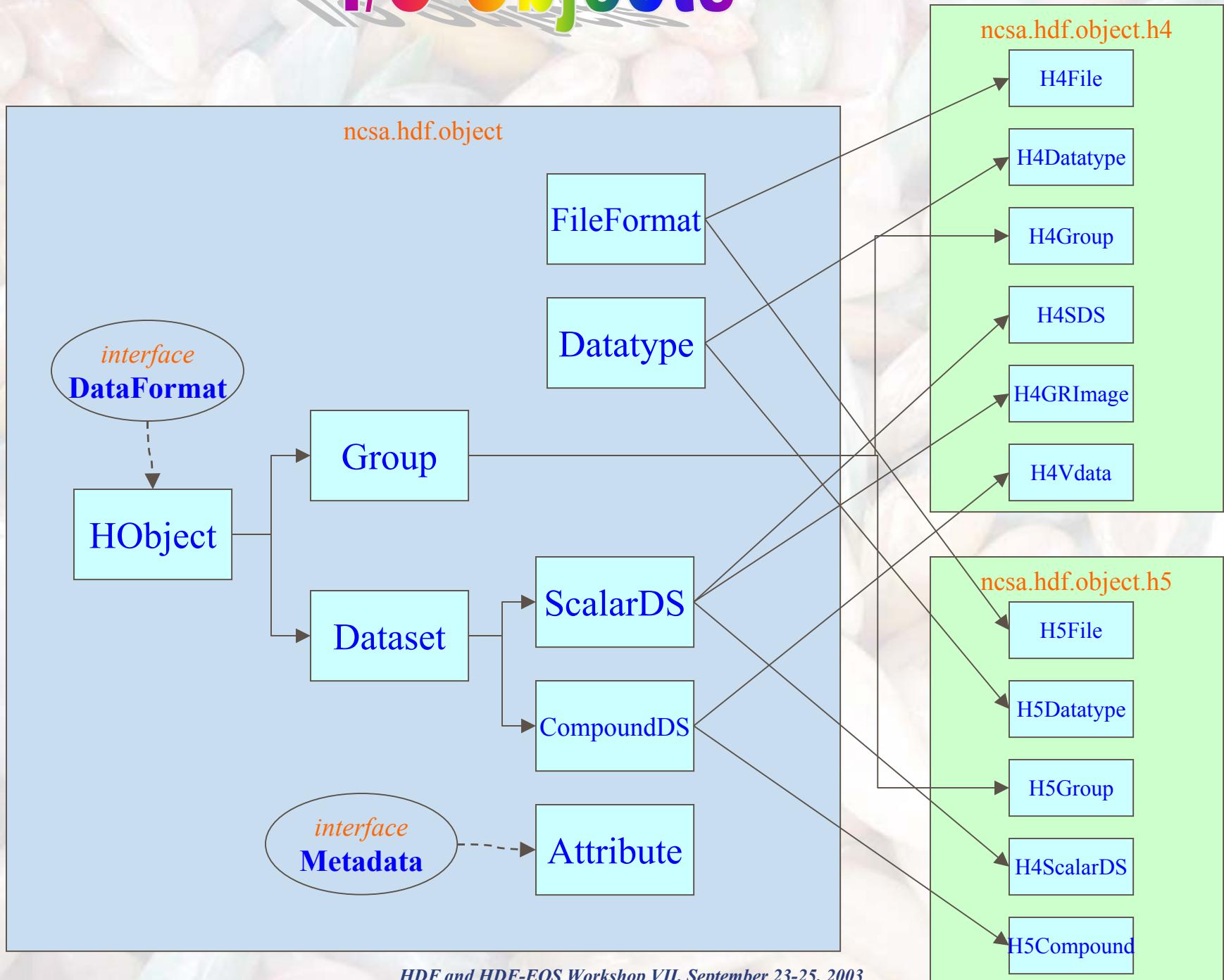
# I/O Module

The common I/O object layer, *ncsa.hdf.object*, defines basic abstract classes for data access such as read/write data. Such abstract I/O layer serves for two purposes:

- easily add new file/data format
- application software depend only on the abstract I/O layer not the implementation



# I/O Objects



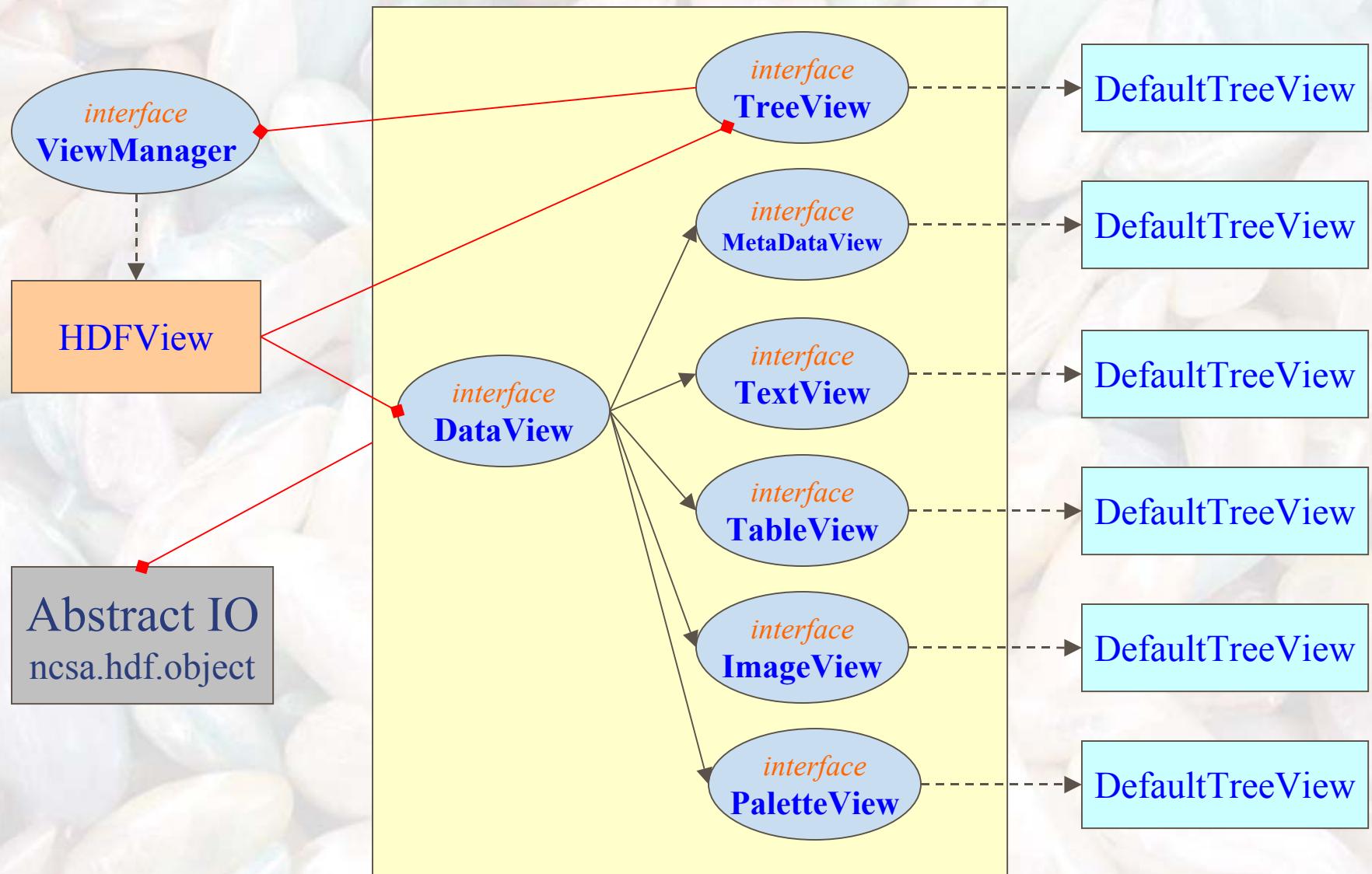
# GUI Modules

GUI components are defined as interfaces. The main view, *ncsa.hdf.view.HDFView*, accesses only to the interfaces, and is separated from implementation

- Adding a new module will not affect the rest
- Users can select module for displaying data

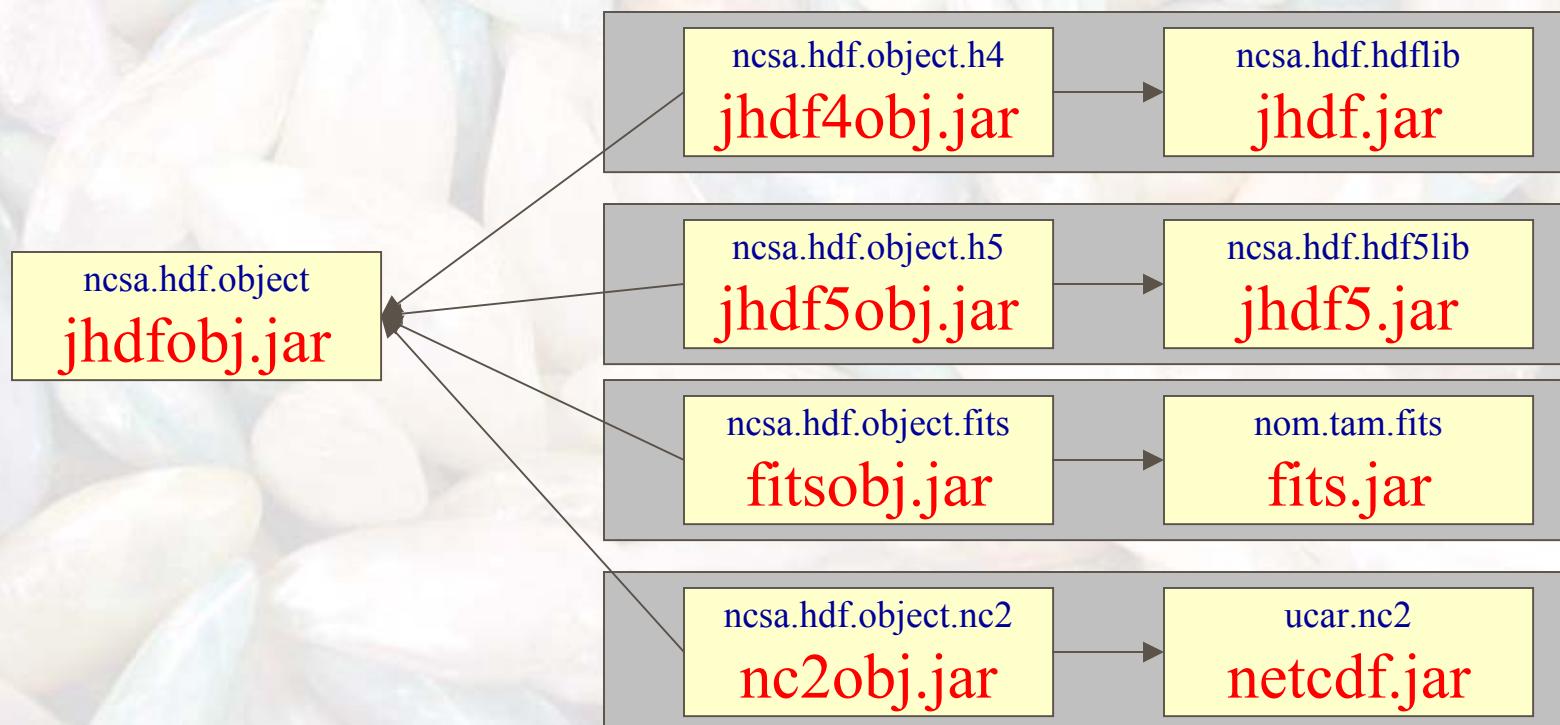


# GUI Modules



# Compiling and Packing I/O

- Required package, *jhdfobj.jar*
- I/O module must be packed in jar file and put it under lib/ext/
- Register I/O module from either property file or FileFormat.addFileFormat()



# Compiling and Packing GUI

- Required package, *j hdfview.jar*
- GUI module must be packed in jar file and put it under *lib/ext/*
- HDFView will automatically detect new module



# For Your Information

- Completed the first round of defining and coding
- The first release is scheduled for December 2003
- Platforms to support
  - Solaris
  - SGI IRIX 6.5
  - Linux
  - Windows 95/98/2000/NT
  - Mac OS X
- For more information visit website  
<http://hdf.ncsa.uiuc.edu/hdf-java-html/hdfview>