J9.19

FRAMEWORK FOR A JAVA METEOROLOGICAL CLASS HIERARCHY

Young Yee
Army Research Laboratory
Computational & Information Sciences Directorate
White Sands Missile Range, NM 88002

ABSTRACT

A framework for a Java™ meteorological class hierarchy is proposed to implement object oriented applications in the atmospheric sciences. One of the recurring issues in the management and handling of enormous quantities of meteorological measurements is formatting and processing of data from one application into other applications. For example, data received from various sources may need special formatting or unit conversions before the data can be properly used as input for a mesoscale model. The major components and functionality of the met class hierarchy will be discussed and a specific implementation of data processing using these classes will be demonstrated.

1. INTRODUCTION

The collection and management of enormous amounts of meteorological data, including satellite-based as well as ground-based measurements, heavily taxes computer resources and presents great challenges in the optimal usage of this information. To address these issues, object oriented technology has become increasingly important. To many scientist and engineers the task of retrofitting legacy computer models that are used operationally can be very daunting. In many cases, there is no easy method to convert legacy procedural computer code to object oriented architectures.

With the proliferation of meteorological data formats , the reformatting and processing of data as input into existing models is fast becoming unmanageable. Traditional complex models typically require very precise formatted data where physical units must be exact. This methodology is prone to human error and if the programmer changes the input format, errors can be difficult to trace. A properly designed architecture should take advantage of repeatable patterns. In object oriented terminology, these reusable templates are referred to as "classes".

* *Corresponding author address:* Y. Yee,
Army Research Laboratory,
White Sands Missile Range, NM 88002-5501
E-mail: yyee@arl.army.mil

2. PROBLEM AREAS

In atmospheric modeling development, programming issues that hinder reuse and portability of code are as follows:

- Lack of a standard Met packages for commonly used routines
- Need for simplification of current model codes
- Need for more compact & specific met applications
- Lack of object oriented designs
- Integration of legacy code
- Complex and confusing data formats
- Need for more graphical tools to view and plot met data
- Specialized databases with custom interfaces
- No cohesive top down structure for finding met applications

3. TOP-LEVEL JAVA MET CLASS HIERARCHY

One of the challenges in management and processing of large meteorological data sets is the development of techniques and methodologies that permit code to be reusable and data to be shared and distributed, without dependencies on operating systems or computer hardware. To address these issues of sharing and platform independence, we propose a top level class hierarchy for meteorological applications. Figure 1 illustrates important key java met packages. In many cases, researchers have initiated extensive development in certain object oriented applications (Murray, 2001) which might fall under one or more of these categories. We provide brief descriptions of each package with a summary of the intended functionality.

Upper Level Java Met Classes:

MetModels – a set of classes that address meteorological computer models from the very simple one-dimensional models to complex models but with modularity in the early design.

MetParam - a set of classes that addresses meteorological parameters and their attributes such as physical units, dimensionality (single value or multi-dimensional arrays), accuracy of measurement, upper and lower limits, and application specific usage.

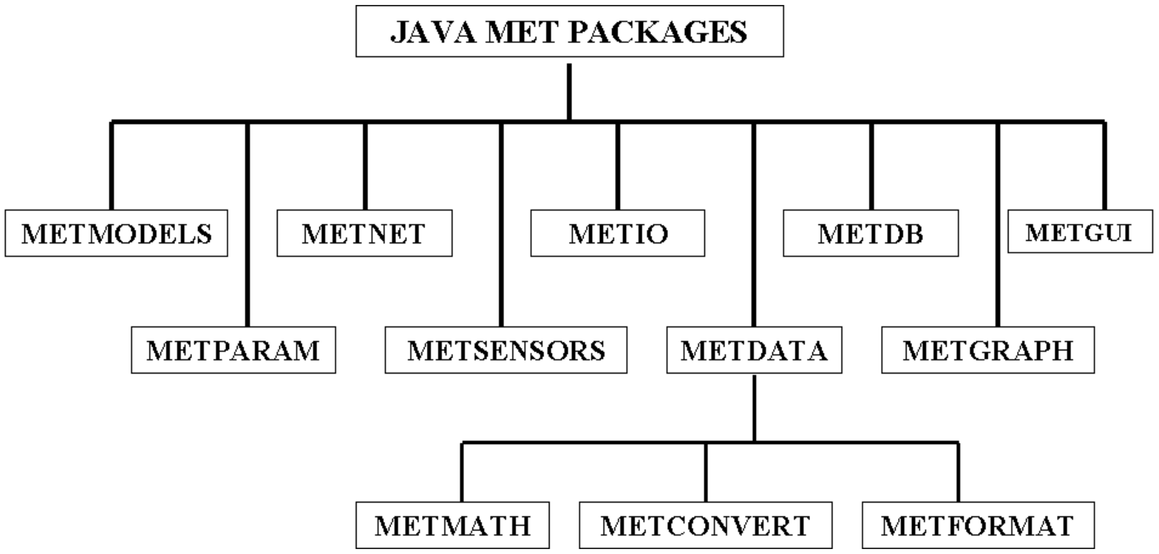

Figure 1. Top Level Java Met Packages

MetNet - a set of classes and interfaces for performing meteorological network operations including acquiring data directly from met sensors or from data sources available on the Internet.

MetSensors - a set of classes that addresses the different meteorological sensors and their characteristics. Calibration methods may be included in these classes.

MetIO - a set of classes that addresses meteorological data inputs and outputs and file handling.

MetData - a set of classes and subclasses to address met data structures and physical units. These classes should have methods to handle different forms of data structures as well as different dimensionality of parameters. Other related classes should include MetMath, MetConvert, and MetFormat to perform data format conversions, interpolations, statistics, unit checking, etc. Compression algorithms may fall under this category or under a MetUtil package.

MetMath - a set of classes and interfaces that contain methods to handle mathematical calculations of meteorological parameters as well as methods to perform linear regression analysis, data statistics, interpolations in time and space, and general math functions

MetFormat - a set of classes and interfaces that addresses the overwhelming data formats that exist for different meteorological applications. Major subclasses should handle imagery data formats, which are computationally expensive to process, and data storage issues.

MetDB - a set of classes that addresses meteorological databases. The classes should have search methods for particular databases, documentations to describe the different databases, and methods to extract and query external databases.

MetGraph - a set of classes and interfaces that addresses meteorological graphical representations, 1D to 4D plots of met information, data formats for specialized graphical software products, and visualization of met data. Visualization of large data sets for quality control of met information is an important issue.

MetGUI - a set of classes and interfaces to provide graphical user interfaces to handle user inputs and requests for specific met data and for model processing options. Other uses could be interactive controls on graphical displays.

An additional package that could be defined is a MetSecurity package to address security access to meteorological information and models. These classes will have methods to discriminate and handle different users such as government personnel versus contractors versus foreign nationals.

4. SAMPLE ILLUSTRATION

To illustrate the applicability of the defined met packages, figure 2 shows different packages that would be used in the processing of a mesoscale model to forecast the weather (Kirby, 2001). In this scenario, NOGAPS (Navy Operational Global Atmospheric Prediction System model) data is used to initialize the mesoscale Battlefield Forecast Model (Henmi, 1996). Note that certain packages are used both before and after the mesoscale model execution. Depending on the graphical tools used, METFORMAT classes may be required to reformat model outputs into graphical inputs. Efficient, modular coding to promote reusability of code is encouraged.

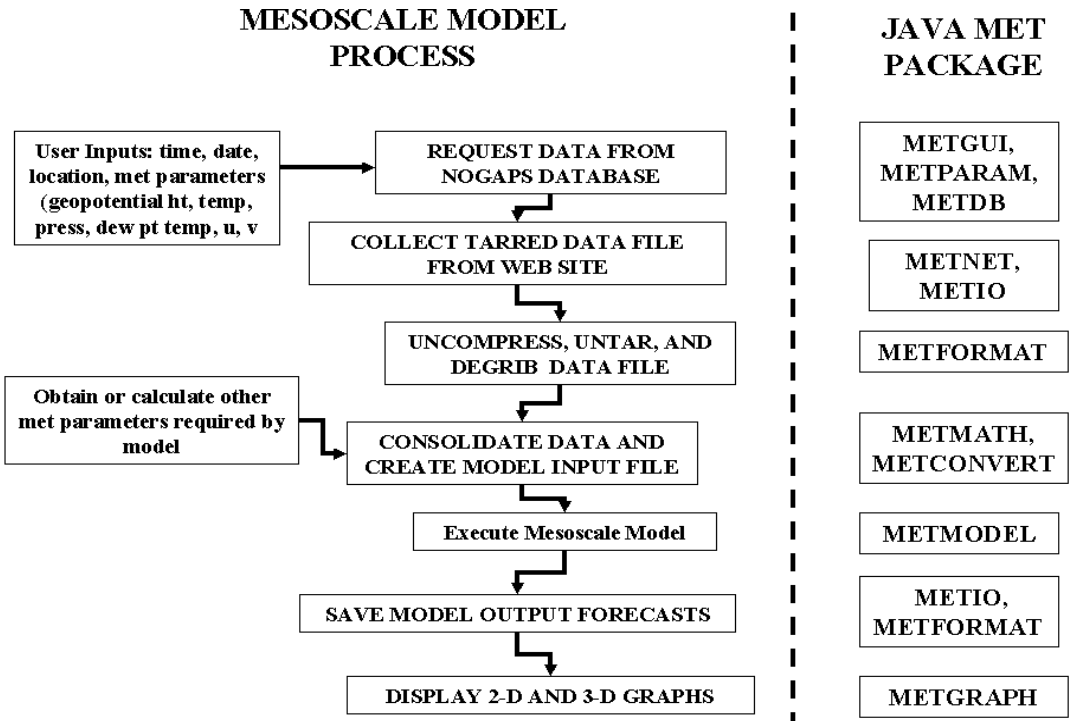
MESOSCALE MODEL PROCESS
JAVA MET PACKAGE
User Inputs: time, date, location, met parameters (geopotential ht, temp, press, dew pt temp, u, v
REQUEST DATA FROM NOGAPS DATABASE
COLLECT TARRED DATA FILE FROM WEB SITE
UNCOMPRESS, UNTAR, AND DEGRIB DATA FILE
Obtain or calculate other met parameters required by model
CONSOLIDATE DATA AND CREATE MODEL INPUT FILE
Execute Mesoscale Model
SAVE MODEL OUTPUT FORECASTS
DISPLAY 2-D AND 3-D GRAPHS
METGUI, METPARAM, METDB
METNET, METIO
METFORMAT
METMATH, METCONVERT
METMODEL
METIO, METFORMAT
METGRAPH

Figure 2. Example of reusable design for mesoscale computer modeling.

5. CLOSING REMARKS

The advantages of defining a top level architecture for Java Met Classes are as follows:
* Applications can be easily located
* Commonality between applications can be identified
* Reuse of modular code can be realized
* Standard terminology can be established especially met parameter nomenclature
* User interfaces can be customized for specific types of met models

To address data discovery issues, the JavaSpaces™ service specification (2000) may provide a distributed persistence and object exchange mechanism for meteorological information and processing. Meteorological objects can be written as entities that provide a typed grouping of relevant meteorological fields. A met client might perform simple operations on a JavaSpaces service to write new meteorological entries, lookup existing met entries, and remove met entries from the space. The idea of persistence is that a collection of data remains intact in the Java space even if its source of origin is no longer attached to the network.

Torres (2001) presents a generic model designed to serve as a blueprint for development of scalable distributed-computing applications. It can be adapted to support various communication protocols. The strength of this basic model lies in the effective use of Java's interface facility. Basically, this model illustrates (polymorphism). Through the interface facility, an application can scale several levels deep and be used to developed sophisticated applications.

6. ACKNOWLEDGEMENTS

The author gratefully acknowledges Brian Malloy, Mario Torres, David Marlin, James Brandt, Teizi Henmi, Steven Kirby, Edward Measure, and James Cogan for their invaluable contributions.

7. REFERENCES

Henmi, T. and R. E. Dumais, “Description of the Battlescale Forecast Model,” US Army Research Laboratory, White Sands Missile Range, NM, ARL-TR-1032, 1996.

JavaSpaces™ Service Specification, Oct 2000,SUNMicrosys,java.sun.com/products/javaspaces.

Kirby, S. F. Y. Yee, P. Haines, T. Henmi, B. A. Malloy, “Exploiting The Internet To Automate The Execution Of Mesoscale Models”, 17th International Conference on Interactive Information Processing Sciences (IIPS), AMS, 14-18 Jan 2001.

Murphy, D., C. Murphy, “Community Software Design: The Unidata Java-Based Metapps Project”, 17th International Conference on Interactive Information Processing Sciences (IIPS), AMS, 14-18 Jan 2001, 367-368.

Torres, Mario A., “Developing Scalable Distributed Applications: A generic model implemented in Java”, Dr. Dobb's Journal, Sept 2001.