A Histogramming Package in Java

N. Takashimizu[1] *and T. Sasaki*[2]

[1] Department of Mathematics and Computer Science, Shimane University
[2] Computing division, KEK

Abstract

We have implemented a prototype of software package for histogramming. Histogram objects are stored in the database, and are exchangeable between C++ and Java application. Using this package, one can store and retrieve histograms in and from the database in analysis programs coded in either C++ or Java.

Java is a simple language which is designed to be object oriented. Some of the complicated features in C++ are excluded; in particular the exclusion of pointers frees programmers from memory management. Architecture-neutral bytecodes generated by the Java compiler enable applications to be transported efficiently to any platform. An efficient development in Java is possible even for unskilled programmers.

The package also implements a histogram visualizer, which can access both C++ and Java objects, including the functionality of interactive operations such as addition, subtraction, fitting, changing bin width and so on.

Keywords: histogramming, OO, Java

1 Introduction

Recently software developments in C++ have been making steady progress in the area of high energy physics. However, a great deal of effort might be needed for many programmers to learn C++. On the other hand, more efficient development in Java can be possible for unskilled programmers because it is a simple language compared to C++. Some of the complicated features in C++ are excluded; in particular exclusion of pointers and the garbage collection mechanism free users from complexities of memory management, and lighten a risk of buffer overruns[1]. Architecture-neutral bytecodes generated by the Java compiler enable applications to be transported efficiently to any platform. Furthermore, it provides a variety of GUI components which are independent of the platform, so it is suited to GUI programming on multi platform environment. Regardless of above advantage there seems to be few experiment adopting Java because many projects to develop applications in C++ have started already. However, if one can share objects between C++ and Java applications one can use each language appropriately to suit one's purpose. We have designed a histogram object which is accessible by applications written in both language, and a tool to display it interactively.

2 Interoperability of persistent objects

In order to exchange a C++ object and a Java object which has different internal expression each other, a technique of object request broker, e.g. CORBA, or ODBMS is necessary. We decided to use Objectivity/DB, which is widely used in HEP. If one wants to share an object stored in the database, one must make Java classes and the corresponding C++ classes compatible completely

with the schema of the shared database[2]. Here we show an example of the schema definition in C++ and Java. To define a persistence-capable class one need to define a class derived directly or indirectly from a persistence-capable class provided by Objectivity/DB such as `ooObj`. This definition is done in a DDL (Data Definition Language) file in C++, a normal source file in Java.

```
// Histo1D.ddl
#include "bin.h"

typedef ooRef(Bin) BinR;
declare(ooVArray, BinR);

class Histo1D : public Histogram
{
  private:
    int32 _numberOfBins;
    float64 _xLow;
    float64 _xHigh;
    ooVArray(BinR) _bin;

    ... snipped.
```

```
// Histo1D.java
public class Histo1D extends Histogram
{
  static int SIZE = 100;
  private int _numberOfBins;
  private double _xLow;
  private double _xHigh;
  private Bin[] _bin = new Bin[SIZE];

    ... snipped.
```

There are several limitations and problems in defining a compatible schema. In the current version of Objectivity/DB, variable-length arrays in C++ are mapped to fixed-length arrays in Java. Further, one cannot define a Java class to be compatible with a C++ class that has multiple inheritance since Java does not support it. However, it is pointed out that multiple inheritance should be avoided even in C++ because it introduces complexities in code readability, therefore, this limitation is not important. A problem is that one cannot share the schema file, that is, a couple of files that represent the same schema always exists. When one wants to change schema definition, one must modify both of files independently.

3 Design

We have designed an experimental model of the histogramming package according to the Unified Meothd[3]. The class diagram is shown in Fig.1.

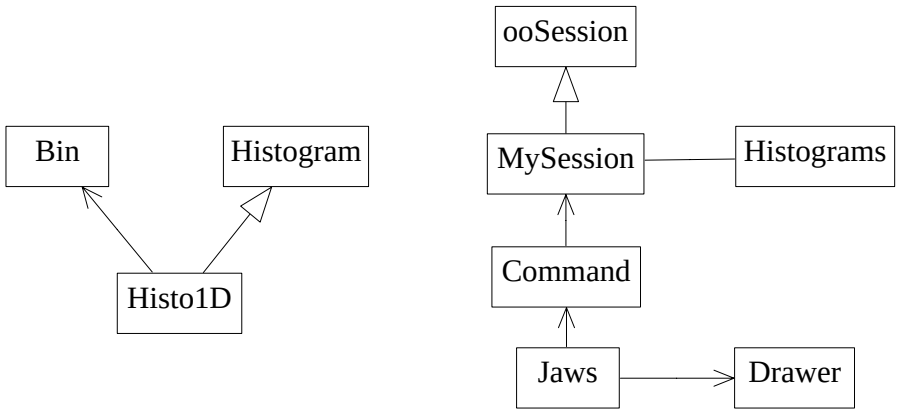

Figure 1: A class diagram of the histogramming package

3.1 A histogramming package

`Histogram` objects contain an collection of `Bin` objects as a member. The collection of the bin is implemented using variable-length array of Objectivity/DB (fixed-length array in Java side). Each `Bin` object has informations of number of entries and errors. `Histograms` object is a container of `Histogram` objects. Objects within a container are physically clustered together in memory pages and on disk, so access to a series of objects in a container is very efficient. A new histogram is created by calling create method in the container object, and clustered into that container. The histogram is equipped with methods of filling and retrieving specified object within the container. `ooSession` is the class for all purpose to establish a connection of an application with the database file and control transactions. `MySession` is the extended class of `ooSession` for accessing persistent Histogram object. The details of the transaction are hidden from the user's application by this class.

3.2 A histogram visualizer

A histogram visualizer is also designed and implemented using classes in the previous section. This package is provisionally named JAWS (Java Analysis WorkShop). It can access both of C++ and Java histograms and is equipped with the functionality of interactive operations such as addition, subtraction, fitting[4], changing bin width and so on. `Jaws` class is the core of the visualizer, which manages users command issued from the command line or GUI.

This package uses functionality provided only by Java except for ODBMS. Since Objectivity/DB is working on Windows and many UNIX's including Linux, the package can be smoothly transported to any platform.

4 Summary

We have implemented a prototype of software package for histogramming efficiently in a short period using Java. It was proved that objects were able to be shared by C++ and Java application without much difficulty.

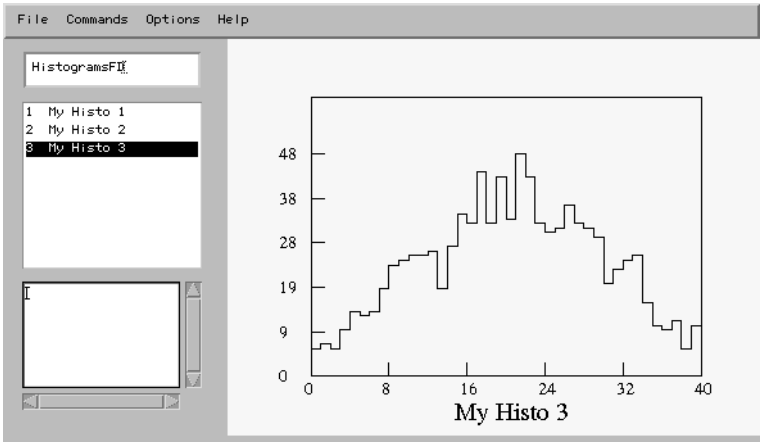

Figure 2: Plotting a histogram

References

1 H. Kumagaya, “Comparison between Java vs. C++”, Java World, pp.48-58 , December 1999.

2 Objectivity for Java Guide, 1998

3 I. Jacobson, G. Booch, J. Rumbaugh, “The Unified Software Development Process”, ISBN 0-201-57169-2.

4 Hiroko Okazawa, ”Development of numerical library software in Java”, submitted to Computing in High Energy Physics 2000, Padova, Italy, Februray, 2000