JavaRoot: An Automatic Java to Root Interface Generator

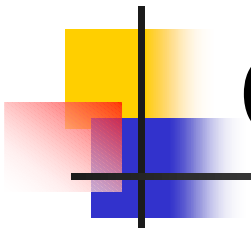
Root Workshop, 2002

CERN, Geneva

Subir Sarkar: INFN/Roma

Rene Brun/CERN

Fons Rademakers/CERN



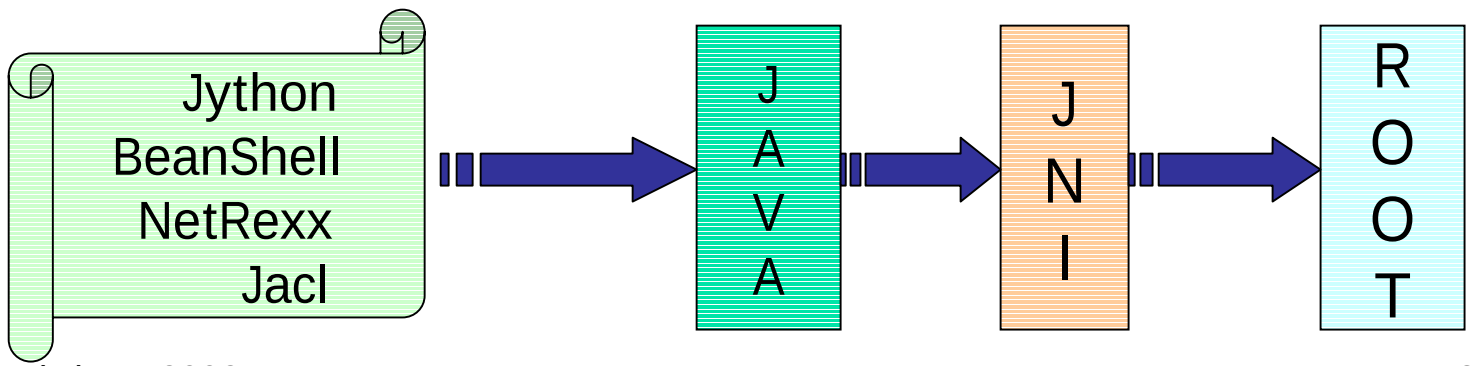
Outline of the Talk

- Motivation
- Different approaches
- Root RTTI based approach
  - Design issues
  - Implementation details
  - Examples and benchmark results
  - Hurdles
- Future plans and conclusion

Motivation

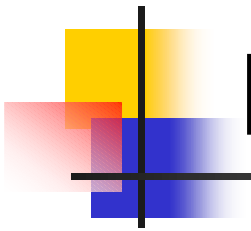
Extend Java with Root libraries

- Java gets a matured Histogramming API, Fitting and Physics analysis classes, a HEP specific Socket programming API
- Root reaches an even wider audience, finds a number of interpreter and scripting environments that are (re)implemented in Java (Jython, BeanShell etc.)



Different Approaches

- Manual Java to Root interface impracticable
  - Root has a large number of classes and a short development cycle
    - impossible to cope up with development
    - interface may go out-of-sync and require coding again
  - So far only a tiny part (e.g Histogramming, Fitting) has been ported, prompted by need
  - Lacks a general approach to the problem
- General purpose tools like SWIG have problems in parsing Root classes as is
  - Wrapping all the Root data members and methods neither needed nor wise
  - SWIG must be augmented with a filtering tool



Root RTTI Based Approach

- Root introspection API to the rescue!
- Load Root classes in a running Root program,
  - Explore type information (method signatures, enums) of the loaded class at runtime (using TClass, TMethod, TMethodArg etc.)
  - Generate code (the hard part)
- For each Root class (e.g TFile), 3 files generated
  - Java proxy class (root/base/TFile.java)
  - JNI header file (base/inc/root_base_TFile.h)
    - Header file name is a JNI convention
  - JNI source file (base/src/root_base_TFile.cxx)

Libraries

- Shared libraries from JNI code

```
-rwxr-xr-x 1 sarkar sarkar 1307186 Sep 19 15:59 libJBase.so
-rwxr-xr-x 1 sarkar sarkar  386833 Sep 19 16:01 libJCont.so
-rwxr-xr-x 1 sarkar sarkar  137241 Sep 19 16:13 libJEG.so
-rwxr-xr-x 1 sarkar sarkar  356818 Sep 19 16:02 libJG3d.so
-rwxr-xr-x 1 sarkar sarkar  320074 Sep 19 16:59 libJGpad.so
-rwxr-xr-x 1 sarkar sarkar  423966 Sep 19 16:05 libJGraf.so
-rwxr-xr-x 1 sarkar sarkar   71303 Sep 19 16:13 libJHbook.so
-rwxr-xr-x 1 sarkar sarkar  699113 Sep 19 16:08 libJHist.so
-rwxr-xr-x 1 sarkar sarkar   78090 Sep 19 16:14 libJHistpainter.so
-rwxr-xr-x 1 sarkar sarkar  217765 Sep 19 16:08 libJMatrix.so
-rwxr-xr-x 1 sarkar sarkar  221829 Sep 19 16:09 libJMeta.so
-rwxr-xr-x 1 sarkar sarkar   58125 Sep 19 16:13 libJMinuit.so
-rwxr-xr-x 1 sarkar sarkar  186624 Sep 19 16:10 libJNet.so
-rwxr-xr-x 1 sarkar sarkar  251288 Sep 19 16:14 libJPhysics.so
-rwxr-xr-x 1 sarkar sarkar   40804 Sep 19 16:10 libJPostscript.so
-rwxr-xr-x 1 sarkar sarkar    6316 Sep 19 17:00 libJRoot.so
-rwxr-xr-x 1 sarkar sarkar   46857 Sep 19 15:32 libJRuntime.so
-rwxr-xr-x 1 sarkar sarkar  472189 Sep 19 16:12 libJTree.so
-rwxr-xr-x 1 sarkar sarkar   90968 Sep 19 16:12 libJTreeplayer.so
```

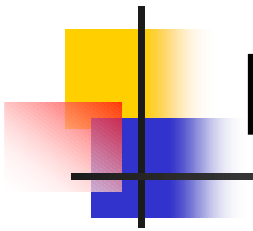
- A jar file containing all the Java classes

```
-rwxr-xr-x 1 sarkar sarkar 445934 Sep 19 16:12 jroot.jar
```

Design Issues

Translation of OO concepts straightforward

Root	Java
Class	Proxy class (mirror)
Pure virtual method	Abstract method
Class with pure virtual methods	Abstract class
Multiple Inheritance Class TH1: public Tnamed, public TAttLine, public TAttFill, public TAttMarker {	Single inheritance plus interfaces public class TH1 extends TNamed implements TAttLineI, TAttFillI, TAttMarkerI {
Operator overloading	Overloaded method
Default values of method arguments	Overloaded methods



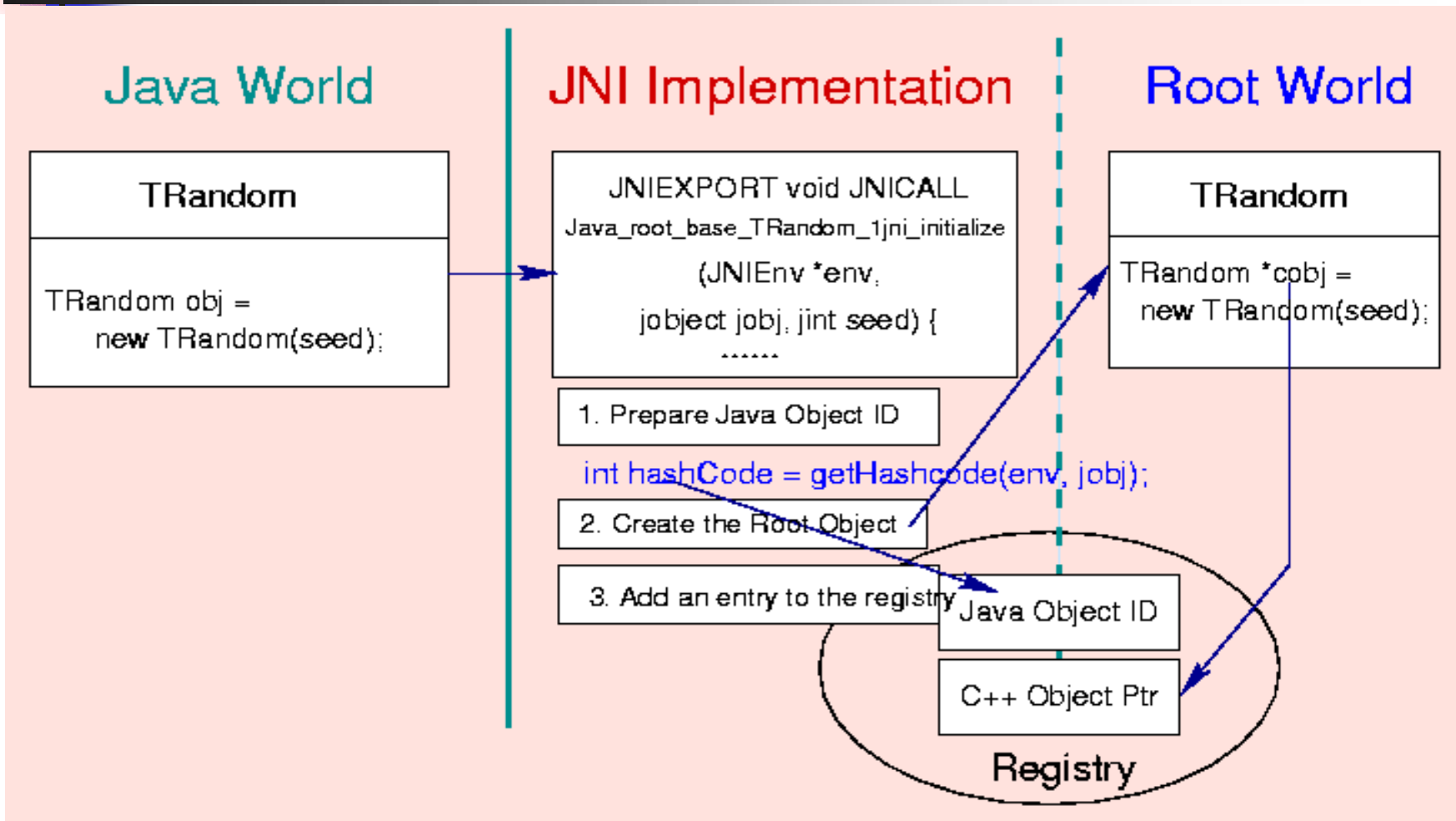
Design Issues

- A no-arg ctor is required in proxy Java classes
  - Handle cases like

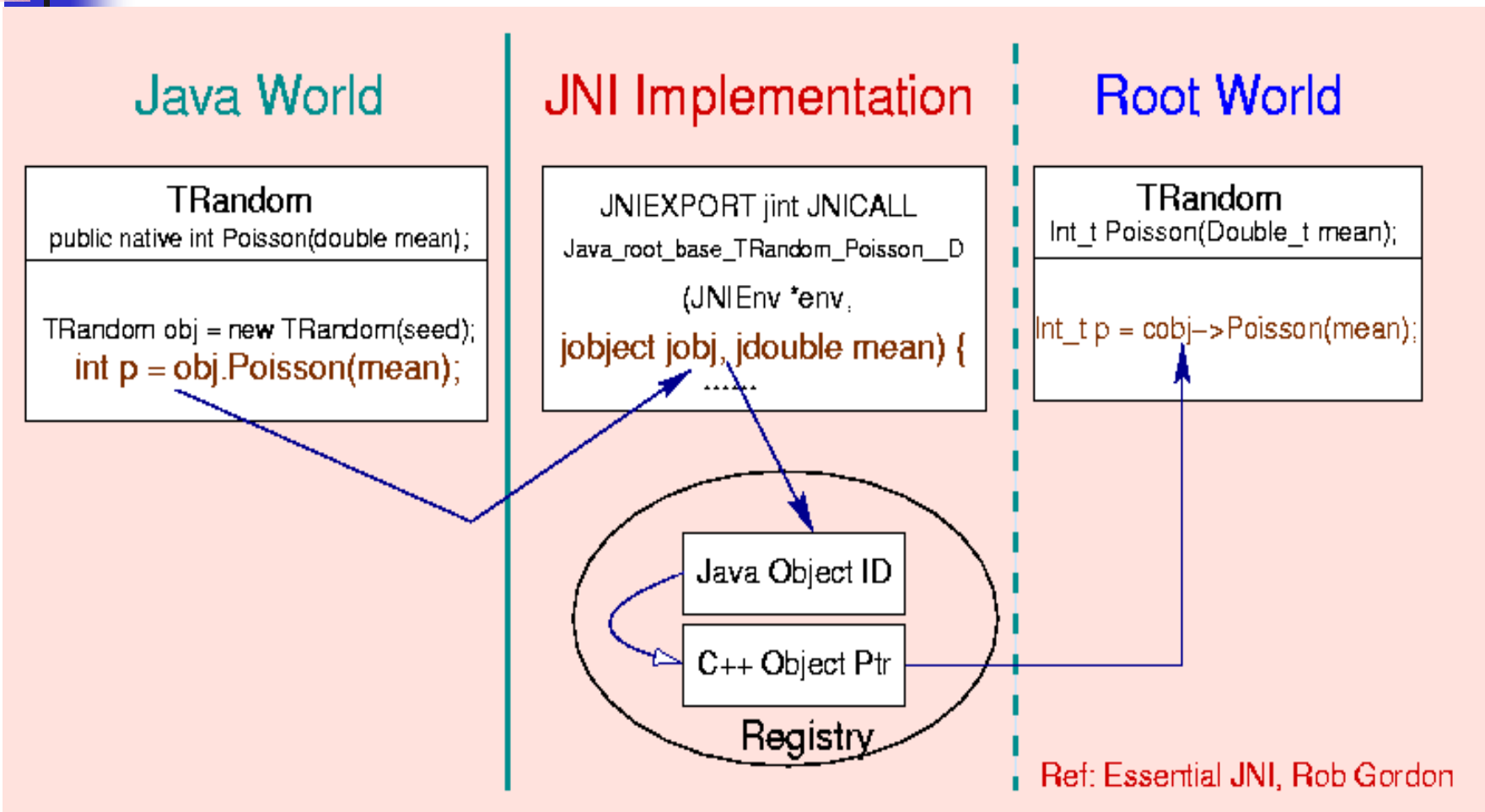
    TDirectory* TH1::GetDirectory() const;

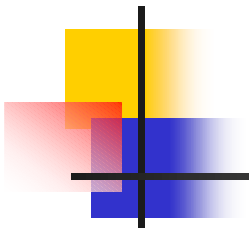
    where the Java object created in native code does not own memory, but can access the underlying Root object
  - A ctor e.g TBenchmark(boolean createNative) is added for convenience whenever relevant
    - Exception: TCanvas already has such a ctor
  - new TBenchmark(); // proxy object only
  - new TBenchmark(true); // underlying Root object as well
  - Default value ignored for ctors with only one argument

Object Creation in Java



Invoking Methods in Java





Implementation

- JavaRoot uses
  - Dictionaries for
    - Root to Java type mapping
    - Operator to method name mapping
  - Lists for
    - Methods not required in Java
    - Root data type filter
    - Header files needed in JNI source file
    - Import statements required in Java class
    - All the public methods of a Root class (+ superclasses)

Generated Code: Java

```
/* DO NOT EDIT THIS FILE - it is machine generated */
package root.base;
import root.cont.TArrayC;
import root.cont.TObjArray;
import root.cont.TList;
public  class TFile extends TDirectory implements Cloneable  {
    public TFile(String fname, String option, String ftitle, int compress)  {
      _jni_initialize(fname, option, ftitle, compress);
    }
    private native  void _jni_initialize(String fname, String option,
                                         String ftitle, int compress);
    ...
    public native void Draw(String option);
    public void Draw()  { Draw(" "); }
    ...
}
```

Generated Code: JNI

```
/* DO NOT EDIT THIS FILE - it is machine generated */
JNIEXPORT void JNICALL Java_root_base_TFile_Draw
          (JNIEnv* env, jobject jobj, jstring option)
{
    // Get back the Root arguments
    Option_t* _option =
        const_cast<Option_t*>(env->GetStringUTFChars(option, NULL));
    assert(_option != NULL);

    // Get the Root object reference
    TFile* _cobj = getCObject(env, jobj);
    _cobj->Draw(_option);   // Execution

    // Release resources
    env->ReleaseStringUTFChars(option, _option);
}
```

Default Value of Method Arguments

- Root

```
TPad TPad(const char* name, const char* title, Double_t xlow, Double_t ylow,
    Double_t xup, Double_t yup, Color_t color = -1, Short_t bordersize = -1,
    Short_t bordermode = -2);
```

- Java

```
public TPad(String name, String title, double xlow, double ylow, double xup,
            double yup, short color, short bordersize, short bordermode) {
  _jni_initialize(name,title,xlow,ylow,xup,yup,color,bordersize,bordermode);
}
public TPad(String name, String title, double xlow, double ylow, double xup,
            double yup, short color, short bordersize) {
   this(name, title, xlow, ylow, xup, yup, color, bordersize, (short) -2);
}
public TPad(String name, String title, double xlow, double ylow, double xup,
            double yup, short color){
   this(name, title, xlow, ylow, xup, yup, color, (short) -1, (short) -2);
}
public TPad(String name, String title, double xlow, double ylow,
            double xup, double yup) {
   this(name, title, xlow, ylow, xup, yup, (short) -1, (short) -1, (short)2);
}
```

JavaBean Property

```
>> import java
>> from root.hist import TH1F
>> java.lang.System.loadLibrary("JRoot")
>> h = TH1F("h", "hist", 100, -3., 3.)
>> h.FillRandom(“gaus”, 10000)
>> h.title
'hist'
>> h.title = "A gaussian distribution“
>> h.GetTitle()
'A gaussian distribution‘
>> h.entries
10000.0
```

Runtime Support

Minimal runtime support required

- A registry to hold Java to Root object mapping
- Code to handle Polymorphic return types
  - TObject* FindObject(const char*)
- Accessing globals like gROOT, gEnv through
  - root.runtime.RootGlobal.gROOT(), etc.
- Related JNI statements put together into C functions for convenience, e.g
  - A pointer/reference <type> as an output arg (type = int,float)

Root	Java	JNI
<type*>/<type&>	org.omg.CORBA.<Type>Holder	set<Type>ObjectValue(...)

Examples: Java

```
import root.base.*;
import root.gpad.TCanvas;
import root.g3d.*;
public class Shapes {
    static { System.loadLibrary("JRoot");}    // try/catch block omitted
    public Shapes()  {
        TCanvas c1  = new TCanvas("c1", "Geometry Shapes", 200, 10, 700, 500);
        TBRIK   brik    =   new TBRIK("BRIK", "BRIK", "void", 200, 150, 150);
        TTRD1  trd1    =   new TTRD1("TRD1", "TRD1", "void", 200, 50, 100, 100);
        TTRD2  trd2    =   new TTRD2("TRD2", "TRD2", "void", 200, 50, 200, 50, 100);
        TTRAP  trap    =   new TTRAP("TRAP", "TRAP", "void", 190, 0, 0, 60, 40, 90, 15, 120, 80, 180, 15);
        ...
        node1.cd(); node1.Draw(); c1.Update(); c1.x3d();
    }
    public static void main(String [] argv) {
      TApplication app = new TApplication("ROOT Application");  new Shapes(); app.Run(true);
   }
}
```

Example: hsimple

CINT

```
{
   gROOT->Reset();
   c1 = new TCanvas("c1","Dynamic Filling
        Example",200,10,700,500);
   c1->SetFillColor(42);
   c1->GetFrame()->SetFillColor(21);
   c1->GetFrame()->SetBorderSize(6);
   c1->GetFrame()->SetBorderMode(-1);

   TFile *hfile = gROOT->FindObject("hsimple.root");
   if (hfile) hfile->Close();
   hfile = new TFile("hsimple.root","RECREATE",
           "Demo ROOT file with histograms");
   hpx     = new TH1F("hpx","This is the px distribution",
                      100,-4,4);
   hpxpy = new TH2F("hpxpy","py vs px",40,-4,4,40,-4,4)
   hprof  = new TProfile("hprof","Profile of pz versus px",
                       100,-4,4,0,20);
   ntuple = new TNtuple("ntuple","Demo  ntuple",
              "px:py:pz:random:i");
   gBenchmark->Start("hsimple");
   gRandom->SetSeed();
```

- Java

```
import org.omg.CORBA.FloatHolder;
public HsimpleTest()  {
   TCanvas c1 = new TCanvas("c1","Dynamic Filling
       ",200,10,700,500);
   c1.SetFillColor((short)42);
   c1.GetFrame().SetFillColor((short)21);
   c1.GetFrame().SetBorderSize((short)6);
   c1.GetFrame().SetBorderMode((short)-1);

   TFile hFile = new TFile("hsimple.root", "RECREATE",
          "Demo ROOT file with histograms");

   TH1F hpx  = new TH1F("h1d","This is the px  distribution",
                       100,-4.0,4.0);
   TH2F hpxy =  new TH2F("h2d","px vs py distribution",
                       40,-4.0,4.0,40,-4.0, 4.0);
   TProfile  hprof = new TProfile("hpro","Profile of pz vs px",
                      100,-4.0,4.0,0.0,20.0);
   TNtuple ntuple = new TNtuple("ntuple","Demo ntuple",
                       "px:py:pz:x:i",0);
   hpx.SetFillColor((short)48);

   root.runtime.RootGlobal.gBenchmark().Start("hsimple");
   TRandom gRandom = root.runtime.RootGlobal.gRandom();
```

Example: hsimple continued ..

- CINT

```
Float_t px, py, pz;
 const Int_t kUPDATE = 1000;
 for ( Int_t i=0; i<25000; i++) {
    gRandom->Rannor(px,py);
    pz = px*px + py*py;
    Float_t random = gRandom->Rndm(1);
    hpx->Fill(px);
    hpxpy->Fill(px,py);
    hprof->Fill(px,pz);
    ntuple->Fill(px,py,pz,random,i);
    if (i && (i%kUPDATE) == 0) {
       if (i == kUPDATE) hpx->Draw();
       c1->Modified();
       c1->Update();
    }
 }
 gBenchmark->Show("hsimple");

 hpx->SetFillColor(0);
 hfile->Write();
 hpx->SetFillColor(48);
}
```

- Java

```
int kUPDATE = 1000;
FloatHolder pxH = new FloatHolder();
FloatHolder pyH = new FloatHolder();
for (int i = 0; i < 25000; i++) {
  random.Rannor(pxH, pyH);
   float px = pxH.value;
   float py = pyH.value;
   float pz = px*px+py*py;
   float x  = (float)random.Rndm(1);
   float xi = (float) i;
   hpx.Fill(px);
   hpxy.Fill(px, py);
   hprof.Fill(px, py);
   ntuple.Fill(px, py, pz, x, xi);
   If (i > 0 && (i%kUPDATE) == 0) {
       if  (i == kUPDATE)  hpx.Draw();
       c1.Modified();
       c1.Update();
   }
 }
root.runtime.RootGlobal.gBenchmark().Show("hsimple")
hpx.SetFillColor((short)0);
 hFile.Write();
hpx.SetFillColor((short)48);
}
```

Example: Jython

```
# FeynmanTest.py
import java.lang  as lang
Import root.base  as base
Import root.gpad as gpad
Import root.graf  as graf
java.lang.System.loadLibrary("JRoot")
class FeynmanTest:
  def __init__(self):
    c1 = gpad.TCanvas("c1", "A canvas",
            10, 10, 600, 300)
    c1.Range(0, 0, 140, 60)
    tex = graf.TLatex(1)
    tex.SetTextAlign(22) ; tex.SetTextSize(0.1)
    l =  graf.TLine(10, 10, 30, 30);l.Draw()
    l =  graf.TLine(10, 50, 30, 30);l.Draw()
    ginit =  graf.TCurlyArc(30, 30, 12.5*lang.Math.sqrt(2), 135,
        225, 0.05, 0.02)
    ginit.SetWavy() ; ginit.Draw()
    tex.DrawLatex(7,6,"e^{-}")
    tex.DrawLatex(7,55,"e^{+}")
    tex.DrawLatex(7,30,"#gamma")
    gamma =  graf.TCurlyLine(30, 30, 55, 30, 0.05, 0.02)
    gamma.SetWavy();  gamma.Draw()
    tex.DrawLatex(42.5,37.7,"#gamma")
    a =  graf.TArc(70, 30, 15, 0, 360)
    a.Draw()
    tex.DrawLatex(55, 45,"#bar{q}"); tex.DrawLatex(85, 15,"q")
    gluon =  graf.TCurlyLine(70, 45, 70, 15, 0.05, 0.02)
    gluon.Draw()
    tex.DrawLatex(77.5,30,"g")
    z0 =  graf.TCurlyLine(85, 30, 110, 30, 0.05, 0.02)
    z0.SetWavy();
    z0.Draw()
    tex.DrawLatex(100, 37.5,"Z^{0}")
    l =  graf.TLine(110, 30, 130, 10) ; l.Draw("")
    l =  graf.TLine(110, 30, 130, 50) ; l.Draw("")
    gluon1 =  graf.TCurlyArc(110,
       30, 12.5*lang.Math.sqrt(2), 315, 45, 0.05, 0.02)
    gluon1.Draw()
    tex.DrawLatex(135,6,"#bar{q}")
    tex.DrawLatex(135,55,"q"); tex.DrawLatex(135,30,"g")
    c1.Update()

if __name__ == '__main__':
   FeynmanTest()
```

> Jython FeynmanTest.py

Example: fit1

- CINT

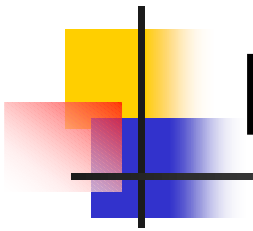
```
{
  gROOT->Reset();
  c1 = new TCanvas("c1","The Fit
       Canvas",200,10,700,500);
  c1->SetGridx();
  c1->SetGridy();
  c1->GetFrame()->SetFillColor(21);
  c1->GetFrame()->SetBorderMode(-1);
  c1->GetFrame()->SetBorderSize(5);

  gBenchmark->Start("fit1");
  TFile fill("fillrandom.root");
  fill.ls();
  sqroot->Print();
  h1f->SetFillColor(45);
  h1f->Fit("sqroot");

  fitlabel = new TPaveText(0.6,0.3,0.9,0.80,"NDC");
  fitlabel->SetTextAlign(12);
  fitlabel->SetFillColor(42);
  fitlabel->ReadFile("fit1_C.C");
  fitlabel->Draw();
  c1->Update();
  gBenchmark->Show("fit1");
}
```

- Jython

```
c1 = gpad.Tcanvas("c1","The FitCanvas", 200,10,700,500);
c1.SetGridx();
c1.SetGridy();
c1.GetFrame().SetFillColor(21);
c1.GetFrame().SetBorderMode(-1);
c1.GetFrame().SetBorderSize(5);

gBenchmark = root.runtime.RootGlobal.gBenchmark();
gBenchmark.Start("fit1");
file = new base.TFile("fillrandom.root");
file.ls();
sqroot = file.Get("sqroot");
sqroot.Print();
 h1f = file.Get("h1f");
 h1f.SetFillColor(45);
 h1f.Fit("sqroot");

 fitlabel =  graf.TPaveText(0.6,0.3,0.9,0.80,"NDC");
 fitlabel.SetTextAlign(12);
 fitlabel.SetFillColor(42);
 fitlabel.ReadFile("./fit1_C.C");
 fitlabel.Draw();
 c1.Update();
 gBenchmark.Show(“fit1”).
```

Examples: BeanShell

```
# FeldmanCousins.bsh
import root.physics.TFeldmanCousins;
import root.base.TString;
java.lang.System.loadLibrary("JRoot");

f = new TFeldmanCousins(90.0, new TString(""));
Nobserved   = 10.0;
Nbackground = 3.0;
ul = f.CalculateUpperLimit(Nobserved, Nbackground);
ll = f.GetLowerLimit();
System.out.println("For " + Nobserved +
       " data observed with an estimated background  of " + Nbackground);
System.out.println(" candidates, the Feldman-Cousins method of calculating CL gives:");
System.out.println("\tUpper Limit = " + ul);
System.out.println("\tLower Limit = " + ll);
System.out.println("at the 90% CL");

> java bsh.Interpreter FeldmanCousins.bsh       #  Execution
```



Root Benchmark

- Original Root benchmark suite partially implemented
- Development environment
  - Celeron 400MHz, 192 Mb Ram, 2 Mb Video Ram Laptop
  - Linux RH62, egcs-2.91.66
  - Root 3.03/06
  - JDK1.4
  - Jython 2.1
  - No JIT compiler

Benchmark Results

```
------ ROOT 3.03/06 benchmarks summary (in ROOTMARKS) -----
For comparison,a Celeron 400Mhz is benchmarked at 280 ROOTMARKS(CINT)
```

Java		
hsimple =	42.67 RealMARKS,	29.30 CpuMARKS
hsum =	59.76 RealMARKS,	47.24 CpuMARKS
fillrandom =	131.25 RealMARKS,	80.00 CpuMARKS
fit1 =	88.89 RealMARKS,	162.50 CpuMARKS
tornado =	125.00 RealMARKS,	75.00 CpuMARKS
na49view =	43.98 RealMARKS,	11.84 CpuMARKS
ntuple1 =	56.89 RealMARKS,	52.69 CpuMARKS

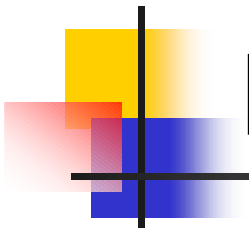
```
*************************************************
* Your machine is estimated at  105.42 ROOTMARKS *
*************************************************
```

Benchmark Results

```
----- ROOT 3.03/06 benchmarks summary (in ROOTMARKS) -----
For comparison,a Celeron 400Mhz is benchmarked at 280 ROOTMARKS(CINT)
```

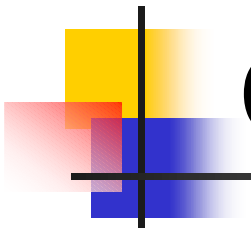
Jython

```
   hsimple =  16.78  RealMARKS,  11.07  CpuMARKS
      hsum =  30.03  RealMARKS,  22.46  CpuMARKS
fillrandom = 140.00  RealMARKS,  75.00  CpuMARKS
      fit1 =  80.00  RealMARKS, 118.18  CpuMARKS
   tornado =  16.00  RealMARKS,  10.53  CpuMARKS
  na49view =  43.32  RealMARKS,  11.61  CpuMARKS
   ntuple1 =  50.69  RealMARKS,  44.22  CpuMARKS
**************************************************
*  Your machine is estimated at    51.80  ROOTMARKS *
**************************************************
```



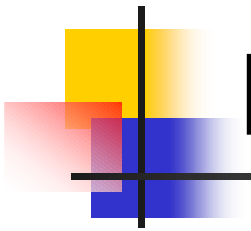
Hurdles

- Yet to obtain a solution
  - pointers
    - void * (TTree::Branch(...), TBranch::SetAddress(...))
    - pointer-to-pointer
    - pointer return type
    - function pointer (TF1, TMinuit)
- Not addressed at all
  - Coexistence of both Java and Root mainloops
  - Implementation of
    - GUI and Thread packages, not really needed
    - ofstream and other STL types



One Possible Future Direction

- JavaRoot does not support STL, might prove too difficult
- Combine Root RTTI with SWIG
  - SWIG supports
    - STL
    - Many languages (Perl, Python, Scheme etc.)
  - Use Root introspection API
    - filter unwanted methods (and data members)
    - create header files easy for SWIG to parse
  - Maintenance of the interface will be much simpler
  - Root will enjoy many language support automatically
- Boost might be a better alternative to SWIG



Plans and Conclusion

- JavaRoot works for a large number of Root classes
- Benchmark results encouraging, a JIT enabled JVM would perform better
- Address the pending problems
  - Java vs Root mainloop
  - Function pointers, void pointers
- More tests required before a public release
  - e.g Root stress test
- Final goal is to make it an integrated part of Root
  - gmake rootjava should wrap a standard set of classes
- For more information see http://sarkar.home.cern.ch/sarkar/jroot/main.html