CIS W338

UML Example using Rational Rose
Overview

• Show how to use Rational Rose
• Give a complete example -- Lab 1
• Show the expectations for the Design Phase
Starting Rational Rose

Select Start --> Programs --> Rational Rose C++ Demo --> Rational Rose C++ Demo.
After clicking OK, you should see the following:
Expand Use Case View and Double Click on Main
Select the actor button.
Move the mouse to the Use Case Diagram and click.
Right click on the actor symbol and select Specification.

This will bring up the following.

Enter “user” into the name field and click on OK.
Select the use case button and then move the mouse to the use case diagram and left click.
Right click on the Use case and then left click on Specification

Enter the data shown.

This use case begins when the user enters the command Mean_and_Stdev <file name> on the user console. The system opens the specified file of real numbers. Reads them into a linked list, and then computes the mean and standard deviation. The resulting values are output to the user console.
Select the Unidirectional Association button.

Place the mouse over the User.
Left click and drag to the Use Case.

Release the left button.
Select the Class Diagram. (It is under the Use Case Diagram). Select the class button.

Move to the Class Diagram and left click.
Left click on the class and select Specification.

Enter the data shown below.

![Class Specification for Untitled](image)
Left Click and select insert operator.

Enter "open(name:char[])"

Enter the other two operators as shown.
Select specification.
Select the Operators tab
Select open.
Select Specification.

Enter the data shown.
Select the Pre Conditions tab.

Select the Post Conditions Tab.
Continue to add the following classes and operators:

• list
  • pushback(value : T)
  • begin():iterator
  • end():iterator
  • size():int
• list<T>::iterator
  • operator*():T
  • operator++():iterator
• list<float>
• list<float>::iterator
• sumfun
  • operator()
• Identity
  • operator():float
• meandifsq
  • meandifsq(m:float=0)
  • operator():float
• sumfun<list<float>::iterator, float, identity>
• sumfun<list<float>::iterator, float, meandifsq>
• MEANandSTDEV
  • main(argc:int, argv: char**)
  • OpenFile(name:char*, the_file:in out ifstream):bool
  • ReadData(the_input:istream, the_list:list<float>
list, list<T>::iterator, and sumfun are Parameterized Classes. In the Specification dialog, under the general tab, select this as the class type.
list<float>, list<float>::iterator, sumfun<list<float>, float, identity> and sumfun<list<float>, float, meandifsq> are Instantiated classes select this type in the specification. Create a dependency relation between the Instantiated classes and their corresponding Parameterized Class.
Create an association between identity and sumfun<>identity<>

Create an association between meandifsq and sumfun<>meandifsq<>

Create an association between list and list<T>::iterator.
Select Aggregation

Link from list<float> to MEANandSTDEV
Right Click on the link and select specification. Then enter “The_List” for Role B.

Select the Role A Detail tab and then select By Value.
The_List shown as a component of MEANandSTDEV.
Right click on the use case. Select new -- Sequence Diagram.

Give it the name as shown.
Select the user. Drag it. Into the sequence diagram.
Drag the following classes into the sequence diagram:
• MEANandSTDEV
• ifstream
• list<float>
• sumfun<list<float>,float,identity>
• identity
• sumfun<list<float>,float,meandifsq>
• meandifsq
• list<float>::iterator
Select Message

Draw a link from the User to MEAN and STDEV.
Right Click and Select the message main.
The Input:
ifstream
The List:
list<float>
sumfun(list<float>::iterator, float, identity) = sumfun(list<float>::iterator, float, meandifsq) = meandifsq
list<float>::iterator
identity

main (int, char*)
OpenFile (char*, ifstream)
open (char*)
operator bool ()
ReadData (ifstream, list<float>)
operator>> (float)
push_back (float)
begin ()
end ()
operator()
operator+ ()
operator* ()
operator++ ()
operator++ ()
size ()
operator()
sumfun

template <class iterator, class T, class FUN>
T sumfun(iterator b, iterator e, FUN fun, T initial)
{
    T result = initial;
    for (iterator i = b; i != e; i++)
    {
        result += fun(*i);
    }
    return result;
}
identity

class identity
{
    public:
        float operator()(float x)
        {
            return x;
        }
};
class meandifsq
{
public:
    meandifsq(float m) : mean(m) {}
    float operator()(float x)
    {
        float d = x - mean;
        return d*d;
    }
private:
    float mean;
};
int main(int argc, char* argv[]) 
{
    std::ifstream The_Input;
    std::list<float> The_List;

    if (OpenFile(argv[1], The_Input))
    {
        ReadData(The_Input, The_List);
        double sum = sumfun(The_List.begin(), The_List.end(), identity(), 0.0);
        float mean = sum/The_List.size();
        double sumdif = sumfun(The_List.begin(), The_List.end(),
                                meandifsq(mean), 0.0);
        float stdev = sqrt(sumdif/(The_List.size()-1));
        std::cout << "Mean : " << mean << " StdDev : " << stdev << std::endl;
        return 0;
    }
    else
    {
        return 1;
    }
}
bool OpenFile(char* name, std::ifstream& the_file)
{
    the_file.open(name);
    if (the_file)
        return true;
    std::cerr << "Unable to open file: " << name << 
        " for input";
    exit(1);
    return false;
}
ReadData

void ReadData(std::istream& the_input,
              std::list<float>& the_list)
{
    float d;
    while (the_input >> d)
        the_list.push_back(d);
}
Visitor

abstract class Visitor {
    public abstract void Apply(Object o);
};
import java.util.*;

class ListWithVisitor extends LinkedList {
    public void VisitEach(Visitor v) {
        Iterator itr = this.iterator();
        while (itr.hasNext()) {
            v.Apply(itr.next());
        }
    }
}

import java.util.*;
import java.io.*;

class ListOfDoubles extends ListWithVisitor {
    public void ReadData(String name) {
        try {
            FileReader fileIn = new FileReader(name);
            StreamTokenizer in = new StreamTokenizer(fileIn);
            while (in.nextToken() != StreamTokenizer.TT_EOF)
                if (in.ttype == StreamTokenizer.TT_NUMBER)
                    this.add(new Double(in.nval));
        } catch (IOException e) { System.err.println(e); }
    }
};
Sum

class Sum extends Visitor {
    public Sum() {sum = 0.0;}
    public double Result() {return sum;}
    public void Apply (Object o) {
        sum += ((Double)o).doubleValue();
    }
    private double sum;
}

2/27/2001  ©2000 P. Wolfgang
class SumMeanDiffSq extends Visitor {

    public SumMeanDiffSq(double mean) {sum = 0.0; m = mean;}

    public double Result() {return sum;}

    public void Apply(Object o) {
        double t = ((Double)o).doubleValue() - m;
        sum += t*t;
    }

    private double m;
    private double sum;
}

SumMeanDiffSq