Athens University of Economics and Business

jGnash

Advanced Topics in Software Engineering

Date: 5/6/2006

Names: George Paraskevopoulos (8030185)
John Georgiou (8030013)

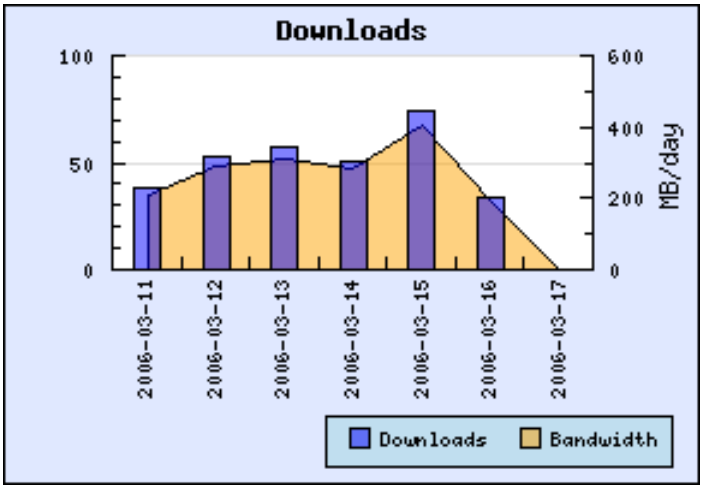
Table of Contents

1. Project Summary

jGnash is a cross platform personal finance application written in Java. It covers the basic tasks of setting up accounts and entering transactions. It contains also some advanced features as report generation, scheduled transactions, investing support, charts etc. jGnash is a double entry system with support for multiple currencies. jGnash can import Gnucash and QIF files.

2. Project Information

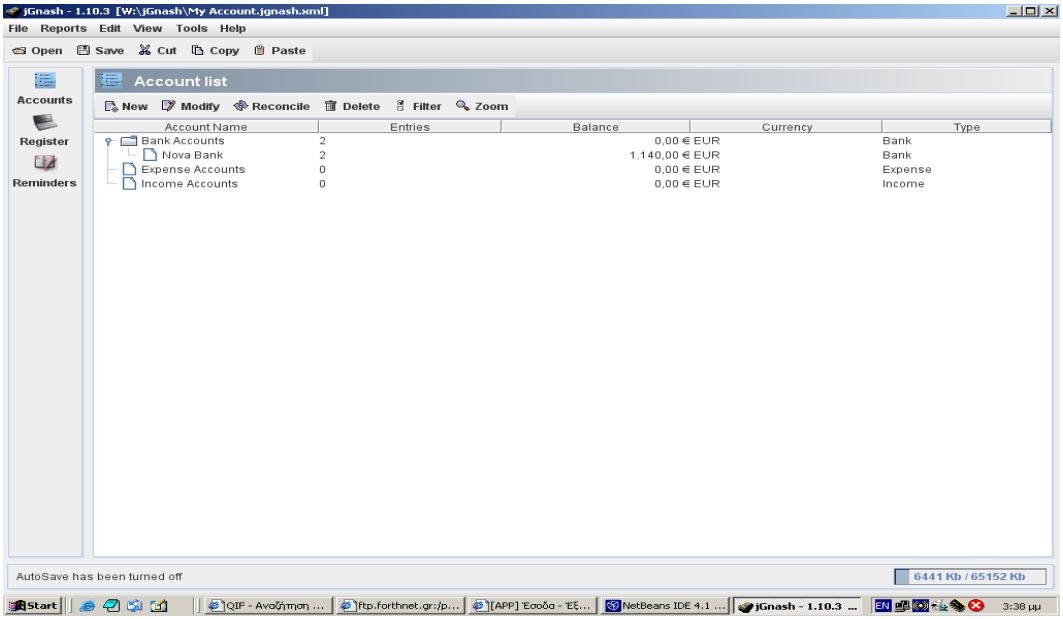
This project runs on all 32-bit MS Windows operating systems, on all POSIX and also on OS X. It is written in Java. It follows GNU General Public License (GPL). It is a project that belongs to two categories: accounting and investment. The development status of the project is 6 which means mature. It composed by 4 developers. The size of the project is 8.882KB and it is about 64.642 lines. There are translations for the languages: English, French, Italian, German, Lithuanian, Russian, Portuguese. This project is registered on sourgeforge.net on 14/5/2001. On the CVS repository there are 3.300 commits and 446 adds. It is a project which has wide – acceptance and large user – satisfaction. Below we show some project download statistics which prove the previous assertion:



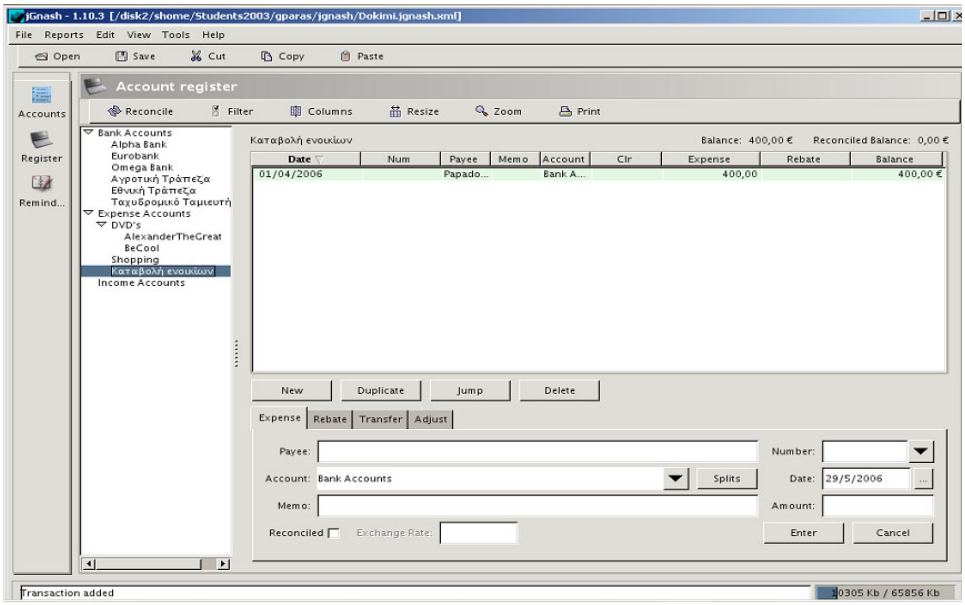
Date	Rank	Total Pages 1	Downloads	Project Web Hits
17 Mar 2006 *	454	0	0	0
16 Mar 2006	428	218	33	0
15 Mar 2006	N/D	635	74	1,606
14 Mar 2006	432	749	51	2,132
13 Mar 2006	435	646	57	2,591
12 Mar 2006	458	567	53	2,718
11 Mar 2006	250	526	38	2,719

3. Project Functionality

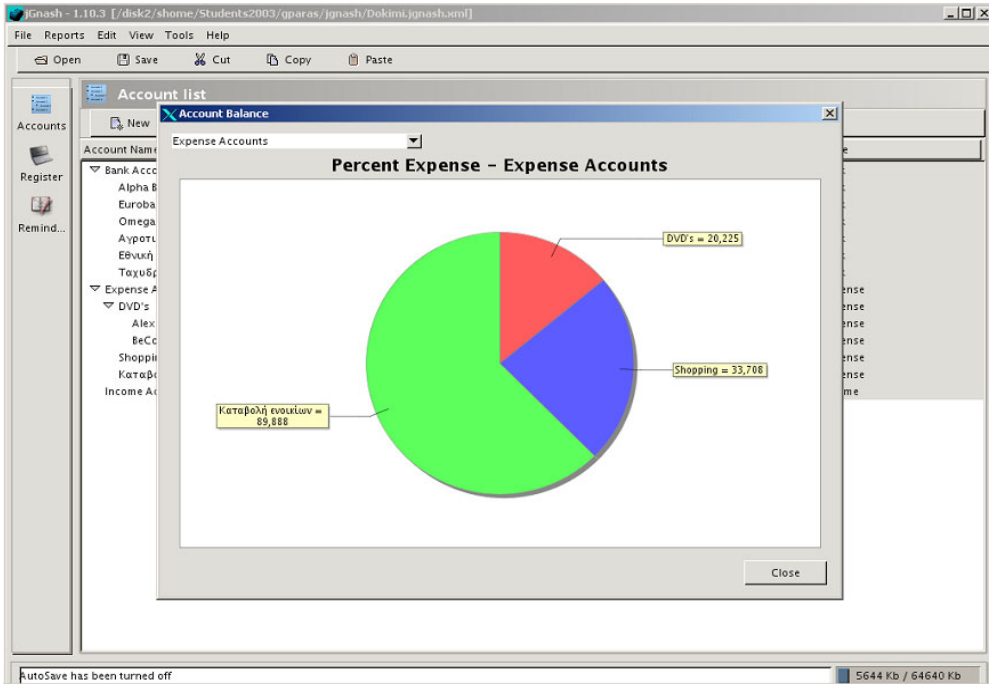
The project has a large amount of features. Three basic features compose the heart of the application. The first one is the account functionality. There you can create the accounts which you need such as bank accounts, expense and income accounts. In the example below we have created the bank account Nova Bank.



The second feature area is called register. There you can insert the transactions which take place on the previous accounts with every detail.
For example:



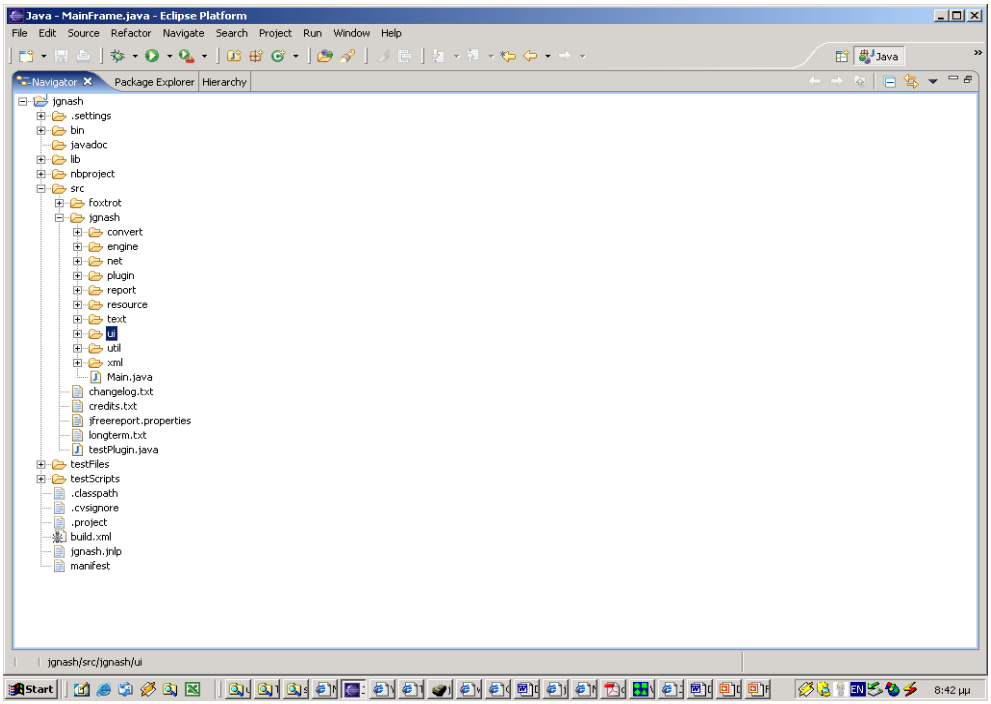
The third functionality is called report generation. The project use jfreechart (another project from sourceforge.net) in order to create reports like pie charts.



4. Design of Modification

4.1 Understanding the project

Firstly we looked thoroughly the project structure.



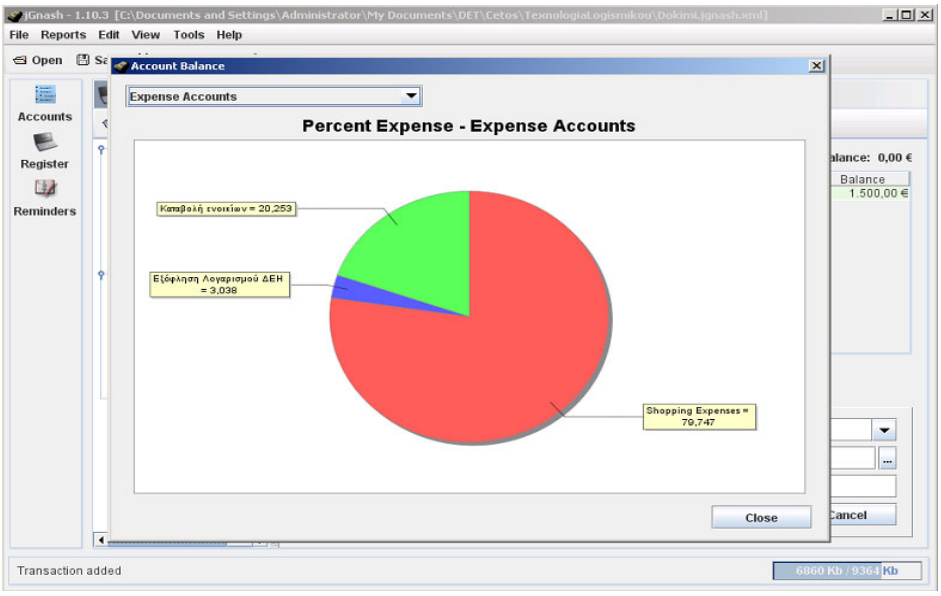
From there we understood the basic components of the application. Then we build and compile the project. We created the javadoc html files. We ran the project in order to understand how it works.

4.2 Choosing the change

We found on the home page of the jGnash a proposal for new reports. We repeat here what this proposal writes:

Proposal for New Reports
From jGnash
*[edit]**jGnash reports***
These are proposals for the modification/addition of jGnash reports. They are heavily influenced by Gnucash reports. More suggestions welcome.
*[edit]**Pie Chart Reports***
*As a general framework, pie chart reports will take these inputs: * From Date: (D1) * To Date (D2) * A set of accounts (S) * Maximum Accounts (M)*
Now, we can define a set of pie chart reports:
1. Expense Pie Chart
All the accounts in S will be of type Expense account. If D1 is the beginning of the current month and D2 is today, the pie chart will show the expenses in various categories in this month. We need to implement an "Account Chooser" widget to choose the accounts in S. In general, the user can choose any number of accounts for this report. But, |S| > M, then the top M-1 accounts will be shown and the other accounts will be grouped together in an "Others Category".
Reports can also be persisted. This means that the user can give a name to each report and save it and reload the report later. This is different from export. The contents of the report will be recomputed each time the report is loaded. Apart from a hard coded date (like 15th Jan 2006), D1 and D2 can also contain a date variable like "Today", "Beginning of Current Month", "End of Current Month", "Beginning of Current Year", "End of Current Year", "Beginning of Last Month" and "End of Last Month". If a saved report is loaded, the date variable will also be recomputed.
2. Income Pie Chart
•Same as above except that accounts in S will be of type Income. 3. Asset Pie Chart Same as above except that accounts in S will be of type Asset. 4. Liabilities Pie Chart
** Same as above except that accounts in S will be of type Liabilities.*

We decided to modify the Expense Pie Chart in order to embed the previous requirement. This means that we have to insert start and end date components and functions in the Expense Pie Charts. The current

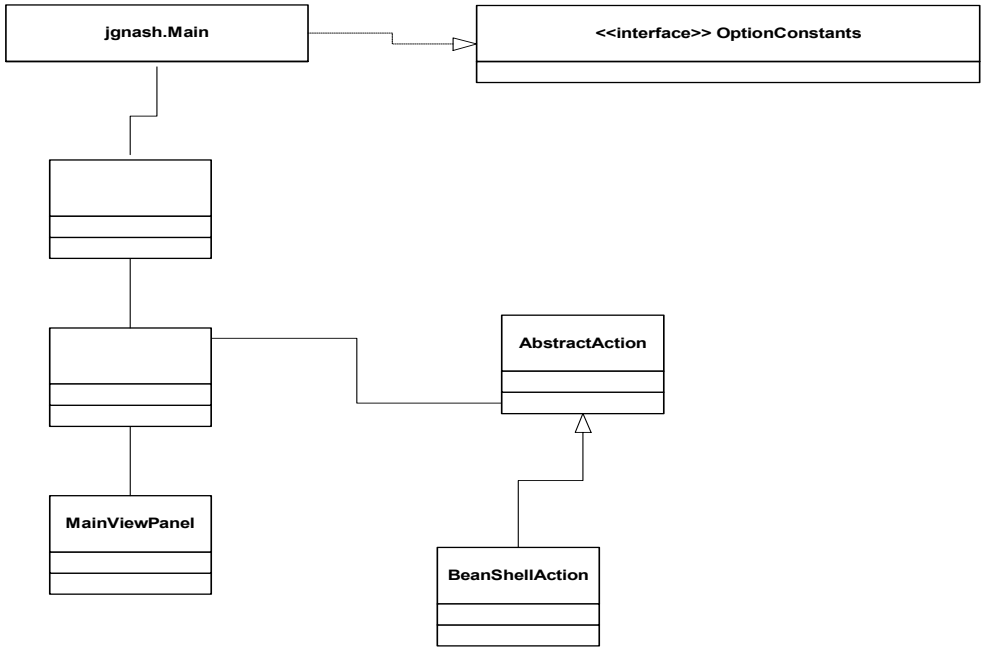


appearance of a Expense Pie Chart is depicted above.

We have also decide to add the Greek language to jGnash so it could be used by the Greeks which aren't proficient with the English language and also do not have the money to buy a similar commercial product, such as Quicken.

4.3 Class Diagram

By using the javadoc files we manage to locate the area of code which must be changed. We depict it with a class diagram:



We determined that the file that needs change is the script
IncomeExpensePieChart.bsh

5. Implementation

5.1 Modification summary

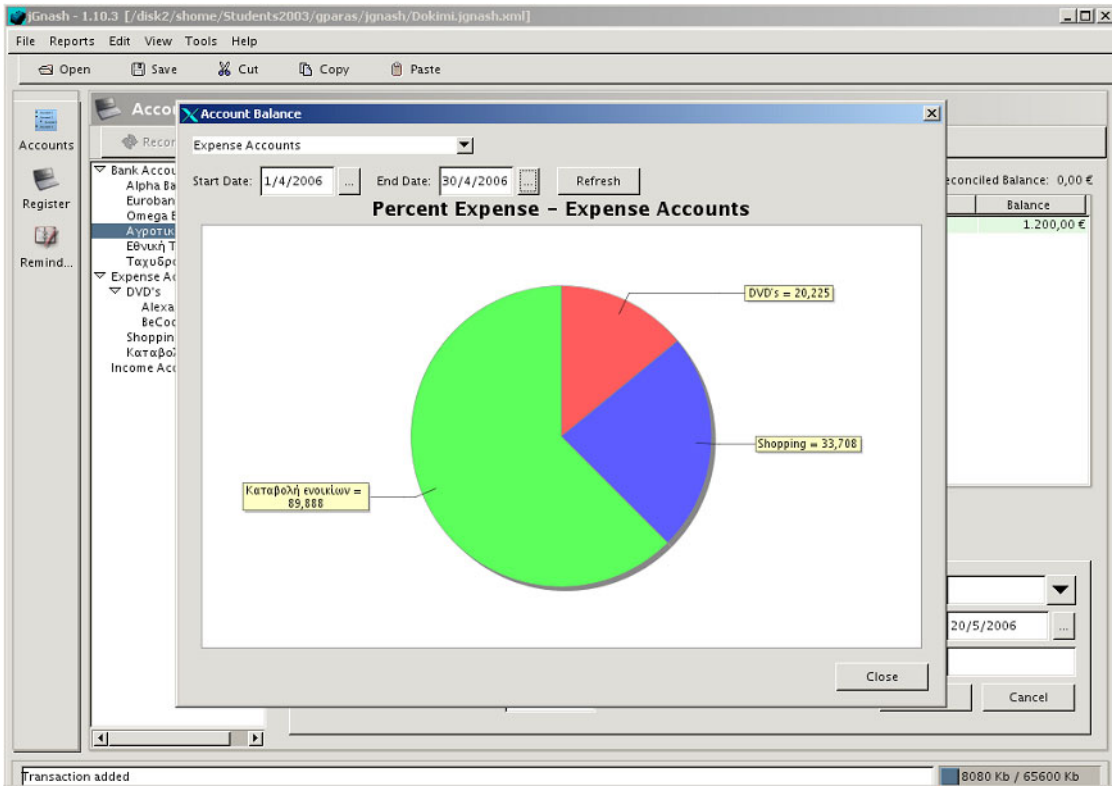
Modification Summary:
We modified the packages which are necessary for the modification.
We added the startField and endField DatePanel.
We gave initial values to the above fields.
We added a refresh button which allows the program to read and apply the new dates.
We modified the layout in order to include the previous components.
We added the necessary listener in order for the refresh button to work.

We invoked the read of the new dates.
We tested the functionality by a series of System.out.println commands.
We used the method *child.getTreeBalance(start, end);* in order to make the program calculate tha balance of the Expense account bettween a start and and date.

For the Greek localisation two new property files need to be added and a new property package needs to be created which are *engine_el_GR.properties, resource_el_GR.properties* and *jgnash.resource.text.el* respectively.

5.2 Modification's result

After modifying the script ***IncomeExpensePieChart.bsh*** we manage to implement the changes on the report Expense Pie Chart. We put the modified script in the location: /jgnash/bin/jgnash/report/scripts. The result is depicted in the following picture:



The role of the Refresh button is to make the chart to be adjusted to the new start and end date.

We can see below some of the changes done to one of the property files jGnash uses:

```
Button.AccTerms                = Χρήση Λογιστικών Όρων
Button.Accounts                = Λογαριασμοί
Button.AckSel                  = Επιβεβαίωση Επιλεγμένων
Button.Add                     = Προσθήκη
Button.AddRemoveColumns        = Προσθήκη/Αφαίρεση Στηλών
Button.Amortize                = Πληρωμή με Δόσεις
Button.Apply                   = Εφαρμογή
Button.AutoReconcile           = Automatically Reconcile Split Transactions
Button.Back                    = Πίσω
Button.BankAccounts            = Τραπεζικοί Λογαριασμοί
Button.Cancel                  = Ακύρωση
Button.CheckReminders          = Υπενθύμιση Ελέγχου
Button.Clear                   = Καθαρισμός
Button.ClearAll                = Καθαρισμός Όλων
Button.Close                   = Κλείσιμο
Button.Columns                 = Στήλες
Button.CompactDisplay          = Compact Display
Button.ConfirmReminderDelete   = Επιβεβαίωση κατα τη Διαγραφή Υπενθύμισης
Button.ConfirmTransDelete      = Επιβεβαίωση κατα τη Διαγραφή Συναλλαγής
Button.Delete                  = Διαγραφή
Button.DeleteAll               = Διαγραφή Όλων
Button.Duplicate               = Duplicate
Button.Edit                    = Επεξεργασία
```

Appendix

The source code of the modified IncomeExpensePieChart is the following:

```
/*
 * jGnash, a personal finance application
 * Copyright (C) 2001-2005 Craig Cavanaugh
 *
 * This program is free software; you can redistribute it and/or
 * modify it under the terms of the GNU General Public License
 * as published by the Free Software Foundation; either version 2
 * of the License, or (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See
the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA  02111-1307,
USA.
 */

/** This displays a dialog that shows a pie chart with account distribution
 *
 * @author Jeff Prickett prickett@users.sourceforge.net
   @author Craig Cavanaugh ccavanaugh@users.sourceforge.net
 */

import java.awt.event.ActionListener;
import java.math.BigDecimal;

import javax.swing.JPanel;

import com.jgoodies.forms.builder.DefaultFormBuilder;
import com.jgoodies.forms.layout.FormLayout;
import com.jgoodies.forms.layout.RowSpec;

import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PiePlot;
import org.jfree.data.general.DefaultPieDataset;
import org.jfree.data.general.PieDataset;

import jgnash.engine.*;


import jgnash.ui.util.GenericCloseDialog;
```

```
/**packages necessary for the modification*/
import java.util.Date;
import jgnash.util.DateUtils;
import org.jfree.data.time.*;
import jgnash.ui.components.*;
import jgnash.ui.util.*;

AccountListComboBox combo = AccountListComboBox.getFullInstance();

UIResource rb = (UIResource) UIResource.get();
DatePanel startField = new DatePanel();
DatePanel endField = new DatePanel();

static Date end = DateUtils.lastDayOfMonth(endField.getDate());
static Date start = DateUtils.previousYear(end);

JPanel p;
GenericCloseDialog d;

JPanel createPanel() {

   startField.setDate(start);
   JButton refreshButton = new JButton(rb.getString("Button.Refresh"));




   Account a = combo.getSelectedAccount();
   JFreeChart chart = createPieChart(a);
   ChartPanel chartPanel = new ChartPanel(chart);

   FormLayout layout = new FormLayout("p, 4dlu:g", "");
   DefaultFormBuilder builder = new DefaultFormBuilder(layout);

   FormLayout dLayout = new FormLayout("p, 4dlu, p, 8dlu, p, 4dlu, p, 8dlu,
p", "");
   DefaultFormBuilder dBuilder = new DefaultFormBuilder(dLayout);
   dBuilder.append(rb.getString("Label.StartDate"), startField);
   dBuilder.append(rb.getString("Label.EndDate"), endField);
   dBuilder.append(refreshButton);
   builder.append(combo);
   builder.nextLine();
   builder.appendRelatedComponentsGapRow();
   builder.nextLine();
   builder.append(dBuilder.getPanel(), 2);
   builder.nextLine();
   builder.appendRow(new RowSpec("fill:p:g"));
   builder.append(chartPanel, 2);
```

```
    JPanel panel = builder.getPanel();

    listener = new ActionListener() {
        actionPerformed(ActionEvent e) {
            try {
                Account a = combo.getSelectedAccount();
                if (a == null) { return; }
                chartPanel.setChart(createPieChart(a));
                panel.validate();
            } catch (Exception ex) {
                print(ex);
            }
        }
    };

    combo.addActionListener(listener);
    refreshButton.addActionListener(listener);

    return panel;
}

JFreeChart createPieChart(Account a) {


    String title = null;

    // pick an appropriate title
    if (a instanceof ExpenseAccount) {
        title = rb.getString("Title.PercentExpense");
    } else if (a instanceof IncomeAccount) {
        title = rb.getString("Title.PercentIncome");
    } else {
        title = rb.getString("Title.PercentDist");
    }

    title = title + " - " + a.getPathName();

    start = startField.getDate();
    end = endField.getDate();

    PieDataset data = createPieDataset(a);
    PiePlot plot = new PiePlot(data);


    JFreeChart chart = new JFreeChart(title,
JFreeChart.DEFAULT_TITLE_FONT, plot, false);

    chart.setBackgroundPaint(null);

    return chart;
```

```
}

PieDataset createPieDataset(Account a) {

   DefaultPieDataset returnValue = new DefaultPieDataset();
   Date [] list = new Date [0];

   if(a != null)
   {

      Date dateTrans = a.getTransactionAt(0).getDate();
      BigDecimal total = a.getTreeBalance();
      if(a.getChildCount() > 0 && total != null)
      {

         for(int i=0; i<a.getChildCount(); i++)
         {
System.out.println(a.toString());
System.out.println(a.getChildCount().toString());
Account child = a.getChildAt(i);
System.out.println(child.toString() + i);

System.out.println(start.toString());
System.out.println(end.toString());

           System.out.println(dateTrans.toString());

            Account child = a.getChildAt(i);
            if(child != null)
            {



               String key = child.getName();
System.out.println(a.getCommodityNode().toString());

           BigDecimal value = child.getTreeBalance(start, end);
               if(key != null && value != null)
               {
                  double percent = (value.doubleValue() /
                        total.doubleValue()) * 100;
                  returnValue.setValue(key, percent);
               }
            }

         }
      }
   }
   return returnValue;
}
```

```
p = createPanel();
d = new GenericCloseDialog(p);
d.setTitle(rb.getString("Title.AccountBalance"));
d.pack();
d.setModal(false);
d.show();
```