ECE 462 C++ and Java

Lab Exercise 04
Unified Modeling Language using Netbeans

Yung-Hsiang Lu
yunglu@purdue.edu
NetBeans UML® Project

The UML Modeling project provides UML modeling features to the NetBeans IDE. UML modeling allows analysts and designers to design applications using a standard modeling language. Developers are then able to generate source code from the UML model and update the model from changes made in their source code.

What's New

The UML team is in active development for NetBeans 6.0. See the UML developer wiki for the latest information.

For UML Users

If you are an UML user and want to learn more about the UML modules and download full releases for Windows, Linux, or Solaris, please visit the product page.

Bug reporting and Defect Tracking

Software bugs are inevitable and we'd appreciate your input if you've encountered a defect or want an enhancement included in a future release.

http://www.netbeans.org/products/uml/
NetBeans IDE 6.0 Features

UML

NetBeans IDE - UML features - Mozilla Firefox

Choose page language ▼  search:  

Download NetBeans

Overview

Base IDE
  Editor
  Version Control & Developer Collaboration

Java SE
  Swing GUI Builder
  Profiler
  Debugger

Web & Java EE
  Java EE
  Web Applications
  Web Services

Done

YHL  UML using Netbeans
UML Modelling Learning Trail

What is a UML diagram?
The purpose of the Unified Modeling Language is to provide a language-independent and platform-independent modeling notation to assist you in developing your applications.

Getting Started

Why Model with UML?

Tutorials, Guides, and Demos

Tutorials
- Developing Applications
- Designing Applications

Creating Collaboration Diagrams
Creating Sequence

Documentation
- Basic Java Programming
- Java GUIs
- Web Applications
- Java EE Applications
- Mobile Applications
- SOA Applications
- UML Modelling
- Ruby Applications
- C/C++ Applications
- NetBeans Modules and Rich-Client Applications
Why Model With UML?

Contributed by Kris Richards and Cindy Castillo, maintained by Bob May

Last Updated: October 2007

Abstract

The purpose of the Unified Modeling Language (hereinafter referred to as UML) is to provide a language-independent and platform independent modeling notation. UML tools are as versatile as the UML is foundational. This article as a primer for the basic concepts of UML while providing an understanding of the purposes of modeling. It is not to be a how-to manual, but links are provided, where appropriate, to take you to well-executed tutorials to illustrate the steps on how to use the UML features offered in the NetBeans IDE.

This article is written for those engineers who can never find time to model a project before starting to code a those engineers who have not yet considered creating models of their systems prior to coding them. This article presents some methods and strategies to help you be more efficient and possibly even save time. The information...
Create a UML Project in Netbeans
Creates a platform-dependent modeling project in the system with empty containers for your model and elements. As you create and populate your model, Java modeling (business) rules are applied to the model.
UML using Netbeans

Steps
1. Choose Project
2. Name and Location

Name and Location

- Project Name: UMLProject1
- Project Location: C:yunglu\java
- Project Folder: C:yunglu\java\UMLProject1
Use Case Diagram
1. Create New Diagram

Diagram Type:
- Activity Diagram
- Class Diagram
- Collaboration Diagram
- Component Diagram
- Deployment Diagram
- Sequence Diagram
- State Diagram
- Use Case Diagram

Diagram Name: Use Case Diagram 1
Namespace: UMLProject1
### Use Case Properties

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Deposit Money</td>
</tr>
<tr>
<td>Alias</td>
<td>Unnamed</td>
</tr>
<tr>
<td>Visibility</td>
<td>public</td>
</tr>
<tr>
<td>Stereotypes</td>
<td></td>
</tr>
<tr>
<td>Tagged Values</td>
<td></td>
</tr>
<tr>
<td>Documentation</td>
<td></td>
</tr>
<tr>
<td>Constraints</td>
<td></td>
</tr>
<tr>
<td>Template Parameters</td>
<td></td>
</tr>
<tr>
<td>Transient</td>
<td>✔️</td>
</tr>
<tr>
<td>Abstract</td>
<td></td>
</tr>
<tr>
<td>Leaf</td>
<td></td>
</tr>
<tr>
<td>Extension Points</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** The UML diagram shows a properties window for a use case, with fields for name, alias, visibility, and various other properties.
Deposit Money

Unnamed

UML using NetBeans
<table>
<thead>
<tr>
<th>Name</th>
<th>Customer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alias</td>
<td>Unnamed</td>
</tr>
<tr>
<td>Visibility</td>
<td>public</td>
</tr>
<tr>
<td>Stereotypes</td>
<td></td>
</tr>
<tr>
<td>Tagged Values</td>
<td></td>
</tr>
<tr>
<td>Documentation</td>
<td></td>
</tr>
<tr>
<td>Constraints</td>
<td></td>
</tr>
<tr>
<td>Template Parameters</td>
<td></td>
</tr>
<tr>
<td>Transient</td>
<td></td>
</tr>
<tr>
<td>Abstract</td>
<td></td>
</tr>
<tr>
<td>Leaf</td>
<td></td>
</tr>
</tbody>
</table>

**Unannotated Diagram**

**Close**
Deposit Money

Customer
Class Diagram
UML using Netbeans
UML using Netbeans
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>BankAccount</td>
</tr>
<tr>
<td>Alias</td>
<td>Unnamed</td>
</tr>
<tr>
<td>Visibility</td>
<td>public</td>
</tr>
<tr>
<td>Stereotypes</td>
<td></td>
</tr>
<tr>
<td>Tagged Values</td>
<td></td>
</tr>
<tr>
<td>Documentation</td>
<td></td>
</tr>
<tr>
<td>Constraints</td>
<td></td>
</tr>
<tr>
<td>Template Parameters</td>
<td></td>
</tr>
<tr>
<td>Transient</td>
<td>✓</td>
</tr>
<tr>
<td>Abstract</td>
<td></td>
</tr>
<tr>
<td>Leaf</td>
<td></td>
</tr>
<tr>
<td>Active</td>
<td></td>
</tr>
</tbody>
</table>

![UML using Netbeans](image)
BankAccount

Attributes
private int balance

Operations
public BankAccount()
public int getBalance()
public void setBalance(int val)
BankAccount

private int balance

public BankAccount()

public int getBalance()

public void setBalance(int balance)
BankAccount

Attributes
private int balance

Operations
public BankAccount()
public int getBalance()
public void setBalance(int)
BankAccount

Attributes
private int balance

Operations
public BankAccount()
public int getBalance()
public void setBalance(int val)
public void addInterest()
Generate Java Code from UML
BankAccount

Attributes:
private int balance

Operations:
public BankAccount()
public int getBalance()
public void setBalance(int balance)
public void addInterest()
At least one valid target project must be open.
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package javaapplication4;

/**
 *
 * @author yunglu
 */

public class Main {

/**
 * @param args the command line arguments
 */

public static void main(String[] args) {
    // TODO code application logic here
    NewJFrame njf = new NewJFrame();
    njf.setVisible(true);
}
}
/ * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package javaapplication4;

/**
 * @author yunglu
 */

public class Main {

/**
 * @param args the command line arguments
 */

==task successful (total time: 4 seconds)==
public class BankAccount {

    private int balance;

    public BankAccount () {
    }

    public int getBalance () {
        return balance;
    }

    public void setBalance (int val) {
        this.balance = val;
    }

    public void addInterest () {
    }
}
public class BankAccount {

    private int balance;

    public BankAccount () {
    }

    public int getBalance () {
        return balance;
    }

    public void setBalance (int val) {
        this.balance = val;
    }

}
Steps
1. Choose File Type
2. Create New Diagram

Create New Diagram
Diagram Type:
- Activity Diagram
- Class Diagram
- Collaboration Diagram
- Component Diagram
- Deployment Diagram
- Sequence Diagram
- State Diagram
- Use Case Diagram

Diagram Name: Sequence Diagram 1
Namespace: UMLProject1
UML using Netbeans YHL 47
<table>
<thead>
<tr>
<th>Message - Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Message</strong></td>
</tr>
<tr>
<td><strong>Name</strong></td>
</tr>
<tr>
<td><strong>Alias</strong></td>
</tr>
<tr>
<td><strong>Visibility</strong></td>
</tr>
<tr>
<td><strong>Stereotypes</strong></td>
</tr>
<tr>
<td><strong>Tagged Values</strong></td>
</tr>
<tr>
<td><strong>Documentation</strong></td>
</tr>
<tr>
<td><strong>Constraints</strong></td>
</tr>
<tr>
<td><strong>Message Kind</strong></td>
</tr>
<tr>
<td><strong>Receiving Classifier</strong></td>
</tr>
<tr>
<td><strong>Sending Classifier</strong></td>
</tr>
</tbody>
</table>

**Name**

[Image of UML diagram showing properties of a message in Netbeans]
<table>
<thead>
<tr>
<th>Name</th>
<th>Request PIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alias</td>
<td></td>
</tr>
<tr>
<td>Visibility</td>
<td>public</td>
</tr>
<tr>
<td>Stereotypes</td>
<td></td>
</tr>
<tr>
<td>Tagged Values</td>
<td></td>
</tr>
<tr>
<td>Documentation</td>
<td></td>
</tr>
<tr>
<td>Constraints</td>
<td></td>
</tr>
<tr>
<td>Message Kind</td>
<td>result</td>
</tr>
<tr>
<td>Receiving Classifier</td>
<td>User</td>
</tr>
<tr>
<td>Sending Classifier</td>
<td>ATM</td>
</tr>
</tbody>
</table>
UML using Netbeans
A UML sequence diagram showing interactions between a User, ATM, and Account. The sequence includes:

- User inserts a card into the ATM.
- ATM queries the account.
- ATM returns the account record.
- User requests a PIN.
- ATM returns the account record.
State Diagram
UML using Netbeans

YHL

58
Wait for ATM Card
Wait for ATM Card
Wait for ATM Card

Input PIN

Verify
Wait for ATM Card

Input PIN

Verify
- Incorrect
  - Correct

Select Transaction
Submission

- Use Case Diagram
- Class Diagram
- Sequence Diagram
- State Diagram

as shown in the slides

Submit a zip file of the project.