ECE 462
Object-Oriented Programming
using C++ and Java

Object and Class

Yung-Hsiang Lu
yunglu@purdue.edu
What is an Object?

An object can be a “concrete and tangible” entity that can be separated with unique properties:

– you
– your book
– your car
– my computer
– Tom
– Amy’s computer
– your phone
– Sam’s digital camera
– Jennifer’s cat ...
What is an object?

• An object can be abstract and does not have to be tangible:
  – Purdue ECE's student database
  – the email sent by Mark at 9:07AM on 2008/03/22
  – the web page of Purdue ECE 462
  – the song played in WBAA at 7:02PM last night

• An object can contain other objects:
  – a car = wheels + engine + door + windshield + ...
  – a house = kitchen + bedrooms + living room + ...
  – a laptop = keyboard + display + processor + ...
Objects' Three Properties

• Each object is unique and can be **identified** using name, serial number, relationship with another object ...
• Each object has a set of **attributes**, such as location, speed, size, address, phone number, on/off ...
• Each object has unique **behaviors**, such as ring (phone), accelerate and move (car), take picture (camera), send email (computer), display caller (pager)
• Each object has **three** important properties:
  – **unique identity**
  – **attributes**, noun
  – **behavior** (action), verb
Objects’ Interactions

- You (object) press (action) the pedal (object) of your car (object). As a result, your car accelerate (action).
- When your phone (object) rings (action) and alerts (action) you (object) of an incoming call (state), you answer (action) the call (state).
- You submit (action) homework (object) and it is graded (action) with a score (state).
Object as a Special Case

• A person is an object. A student is also an object. A student is a **special case** of a person
  ⇒ A student has **all attributes** of a person:
      name, home address, parents ...
  ⇒ A student has **all behavior** of a person:
      eat, sleep, talk ...
  ⇒ A student has something that a person may not have:
      – attributes: student ID, list of courses, classmates ...
      – behavior: submit homework, take exam ...
What is a Class?

• A class **describes the commonalities** of similar objects:
  – Person: you, David, Mary, Tom, Amy ...
  – Car: your Toyota Camry, his Ford Explorer, Jennifer's Mercedes C300 ...
  – Classroom: EE170, EE117, EE129 ...
  – Building: EE, MSEE, Purdue Bell Tower, Hovde Hall...

• A class describes both the attributes and the behavior:
  – Person: name, home ... + sleep, eat, speak ...
  – Car: engine size, year ... + accelerate, brake, turn ...
Relationship among Classes

• A class can be a **special case** of another class:
  – Student is a special case of Person
  – Sedan is a special case of Car
  – Laptop is a special case of Computer
  – Computer is a special case of ElectronicMachine

⇒ This is called a **"is a"** relationship.
  – any Student object is a Person object
  – any Sedan object is a Car object
  – any Laptop object is a Computer object
  – any Computer object is an ElectronicMachine object
Class and Object

• An object is an **instantiation** (i.e. concrete example) of a class:
  – an object is unique
  – a class describes the common properties of many objects
• An object may contain an object. This must be described in the former object's class. We can say that one class "has a" class.
Encapsulation

• An object can hide information (attributes) from being manipulated by or even visible to other objects:
  A person's name is given once when the object is created. This attribute is visible but cannot be changed.

• An attribute may be modified by only restricted channels to keep consistency.
  A person's address and phone number must be change together when this person moves.
Self Test
ECE 462
Object-Oriented Programming using C++ and Java

Inheritance and Polymorphism

Yung-Hsiang Lu
yunglu@purdue.edu
Inheritance = "Is A"

- Any Student object is a Person object. Student class is a derived class of Person. Person is the base class.
  ⇒ Person is more general, with fewer attributes and behaviors.
  ⇒ Student is more specific, with more attributes (school, major) and behaviors (submit homework, take exam).
- Any TabletPC object is a Computer object. TabletPC class is a derived class of Computer.
  ⇒ Computer is more general.
  ⇒ TabletPC is more specific, with more attributes (battery lifetime) and behavior (close or turn the screen)
Derived Class

- A class may have **multiple** derived classes:
  - Car: Sedan, Truck, Sport Utility Vehicle, Sport Car ...
  - Computer: Laptop, Desktop, Server
  - Person: Student, Teacher, Father, Mother ...
- A derived class may also have derived classes:
  - Vehicle: Car, Bike ... Car: Sedan, Truck ...
  - Animal: Bird, Mammal ... Mammal: Dog, Cat ...
- Use "base" and "derived" classes. **Do not** use "super" and "sub" classes. A base class or a superclass is "smaller" (fewer attributes and behaviors)
  ⇒ too confusing
Why Object-Oriented?

- Object-oriented programming (OOP) is more natural to describe the interactions between "things" (i.e. objects).
- OOP provides better code reuse:
  - commonalities among objects described by a class
  - commonalities among classes described by a base class (inheritance)
- Objects know what to do using their attributes:
  Each object responds differently to "What is your name?"
- OOP provides encapsulation: hide data that do not have to be visible to the other objects or protect data from unintentional, inconsistent changes.
If a behavior is common among classes, the behavior should be **available** in their base class. However, this behavior may need additional information from derived classes and must be **handled** in derived classes.

- Shape: contains color, lineStyle ... attributes
- Shape **supports** getArea behavior
- getArea **cannot** be handled by Shape
- getArea must be handled by individual derived classes
- getArea **implemented** in derived classes
Override Behavior

- Polygon can support getArea.
- Derived classes (such as Triangle, Square, and Pentagon) can have better (faster) ways to getArea.
  \[ \Rightarrow \text{getArea is implemented in Polygon and the derived classes.} \]
- A Polygon object calls getArea in Polygon
- A Square object calls getArea in Square if getArea is implemented in Square.
- A Pentagon object calls getArea in Polygon if getArea is not implemented in Pentagon.
# Overriding

<table>
<thead>
<tr>
<th>Base</th>
<th>Derived</th>
<th>Object</th>
<th>Execute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>Y</td>
<td>Base</td>
<td>Base</td>
</tr>
<tr>
<td>Y</td>
<td>Y</td>
<td>Derived</td>
<td>Derived</td>
</tr>
<tr>
<td>Y</td>
<td>N</td>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td>Y</td>
<td>N</td>
<td>D</td>
<td>B</td>
</tr>
<tr>
<td>N</td>
<td>Y</td>
<td>Base</td>
<td>Error</td>
</tr>
<tr>
<td>N</td>
<td>Y</td>
<td>Derived</td>
<td>D</td>
</tr>
<tr>
<td>N</td>
<td>N</td>
<td>B</td>
<td>Error</td>
</tr>
<tr>
<td>N</td>
<td>N</td>
<td>D</td>
<td>Error</td>
</tr>
</tbody>
</table>

The behavior implemented in a sibling class (such as Square-Triangle) has no effect.
Class and Object

Polygon p1;  // this is a comment: p1 is a Polygon object
p1.getArea();  // call the implementation in Polygon
Square s2;  // s2 is a Square object
p1 = s2;  // p1 now behaves likes a square
// a Square object is always a Polygon object
p1.getArea();  // implementation in Square (if available)
// polymorphism
s2 = p1;  // error
// a Polygon object may not be a Square object
Fundamental Concepts in OOP

- object and class
- encapsulation
- inheritance
- polymorphism
Self Test
Java and Qt Documentations
JDK™ 6 Documentation

Java™ SE 6 Platform at a Glance

This document covers the Java™ Platform, Standard Edition 6 JDK. Its product version number is 6 and developer version number is 1.6.0, as described in Platform Name and Version Numbers. For programmer guides or information on a feature of the JDK, click on a component in the navigation panel.
or information on a feature of the JDK, click on a component in the diagram below.
or information on a feature of the JDK, click on a component in the diagram below.

<table>
<thead>
<tr>
<th>Java Language</th>
<th>Java Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>java</td>
<td>javac</td>
</tr>
<tr>
<td>javadoc</td>
<td>apt</td>
</tr>
<tr>
<td>jar</td>
<td>javap</td>
</tr>
<tr>
<td>JPDA</td>
<td>jconsole</td>
</tr>
<tr>
<td>Security</td>
<td>Int'l</td>
</tr>
<tr>
<td>RMI</td>
<td>IDL</td>
</tr>
<tr>
<td>Deploy</td>
<td>Monitoring</td>
</tr>
<tr>
<td>Troubleshoot</td>
<td>Scripting</td>
</tr>
<tr>
<td>JVM</td>
<td>TI</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Deployment Technologies</th>
<th>Java Web Start</th>
<th>Java Plug-in</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWT</td>
<td>Swing</td>
<td>Java 2D</td>
</tr>
<tr>
<td>Accessibility</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Drag &amp; Drop</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input Methods</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Image</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Print Services</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sound</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Integration Libraries</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>IDL</td>
<td>JDBC™</td>
</tr>
<tr>
<td>JNDI™</td>
<td>JavaSE™</td>
</tr>
<tr>
<td>RMI</td>
<td>RMI-IIOP</td>
</tr>
<tr>
<td>Idl</td>
<td></td>
</tr>
<tr>
<td>JNDI</td>
<td></td>
</tr>
<tr>
<td>RMI</td>
<td></td>
</tr>
<tr>
<td>RMI-IIOP</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Networking</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Beans</td>
<td>JavaSE™</td>
</tr>
<tr>
<td>Int'l Support</td>
<td>JavaSE™</td>
</tr>
<tr>
<td>I/O</td>
<td>JavaSE™</td>
</tr>
<tr>
<td>JMX</td>
<td>JavaSE™</td>
</tr>
<tr>
<td>JNI</td>
<td>JavaSE™</td>
</tr>
<tr>
<td>Math</td>
<td>JavaSE™</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>lang and util Libraries</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>lang</td>
<td>JavaSE™</td>
</tr>
<tr>
<td>util</td>
<td>JavaSE™</td>
</tr>
<tr>
<td>Collections</td>
<td>JavaSE™</td>
</tr>
<tr>
<td>Concurrency Utilities</td>
<td>JavaSE™</td>
</tr>
<tr>
<td>JAR</td>
<td>JavaSE™</td>
</tr>
<tr>
<td>Logging</td>
<td>JavaSE™</td>
</tr>
<tr>
<td>Management</td>
<td>JavaSE™</td>
</tr>
<tr>
<td>Preferences API</td>
<td>JavaSE™</td>
</tr>
<tr>
<td>Ref Objects</td>
<td>JavaSE™</td>
</tr>
<tr>
<td>Reflection</td>
<td>JavaSE™</td>
</tr>
<tr>
<td>Regular Expressions</td>
<td>JavaSE™</td>
</tr>
<tr>
<td>Versioning</td>
<td>JavaSE™</td>
</tr>
<tr>
<td>Zip</td>
<td>JavaSE™</td>
</tr>
<tr>
<td>Instrument</td>
<td>JavaSE™</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Java Virtual Machine</th>
<th>Java Hotspot™ Client VM</th>
<th>Java Hotspot™ Server VM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solaris™</td>
<td>Linux</td>
<td>Windows</td>
</tr>
<tr>
<td>Other</td>
<td>Other</td>
<td>Other</td>
</tr>
</tbody>
</table>
Swing
(Java™ Foundation Classes)

The Swing classes (part of the Java™ Foundation Classes (JFC) software) implement a set of components for building graphical user interfaces (GUIs) and adding rich graphics functionality and interactivity to Java applications. The Swing components are implemented entirely in the Java programming language. The pluggable look and feel lets you create GUIs that can either look the same across platforms or can assume the look and feel of the current OS platform (such as Microsoft Windows, Solaris™ or Linux).

Overview

- About the JFC and Swing from The Swing Tutorial

API Specification

- API Reference
- Focus Model Specification

Tutorials and Programmer's Guides

- The Swing Tutorial
Trail: Creating a GUI with JFC/Swing

Also known as The Swing Tutorial

This trail tells you how to create graphical user interfaces (GUIs) for applications and applets, using the Swing components.

Getting Started with Swing is a start lesson. First it gives you a bit of background about the JFC and Swing. Then it tells you how to compile and run programs that use Swing components.

Learning Swing with the NetBeans IDE is the fastest and easiest way to begin working with Swing. This lesson explores the NetBeans IDE’s GUI builder, a powerful feature that lets you visually construct your Graphical User Interfaces.

Using Swing Components tells you how to use each of the Swing components — buttons, tables, text components, and all the rest. It also tells you how to use borders and icons.

Concurrency in Swing discusses concurrency as it applies to Swing programming. Information on the event dispatch thread and the SwingWorker class are included.
Getting Started with Swing

Lesson: Getting Started with Swing

About the JFC and Swing

Compiling and Running Swing Programs

This lesson gives you a brief introduction to using the Java Foundation Classes (JFC) and Swing. After telling you about JFC and Swing, it helps you get the necessary software and walks you through how to compile and run a program that uses the Swing packages.

The following lesson, Learning Swing with the NetBeans IDE, will build on these first steps to help you create several progressively more complex examples. For now, let's start with the basics.

About the JFC and Swing

This section gives you an overview of the JFC and Swing.
Trail: Creating a GUI with JFC/Swing

This trail tells you how to create graphical user interfaces (GUIs) for applications and applets, using the Swing components.

Getting Started with Swing is a quick start lesson. First it gives you a bit of background about the JFC and Swing. Then it tells you how to compile and run programs that use Swing components.

Learning Swing with the NetBeans IDE is the fastest and easiest way to begin working with Swing. This lesson explores the NetBeans IDE's GUI builder, a powerful feature that lets you visually construct your Graphical User Interfaces.

Using Swing Components explains how to use each of the Swing components — buttons, tables, text components, and all the rest. It also tells you how to use borders and icons.

Concurrency in Swing discusses concurrency as it applies to Swing programming. Information on the event dispatch thread and the SwingWorker class are included.
Using Swing Components

- Using Top-Level Containers
- The JComponent Class
- Using Text Components
  - Text Component
  - Features
  - The Text Component API
- How to Use Various Components
  - How to Make Applets
  - How to Use Buttons, Check Boxes, and Radio Buttons
- How to Use the ButtonGroup Component
- How to Use Color Choosers
- How to Use Combo Boxes

Lesson: Using Swing Components

Examples Index

This lesson gives you the background information you need to use the Swing components, and then describes every Swing component. It assumes that you have successfully compiled and run a program that uses Swing components, and that you are familiar with basic Swing concepts. These prerequisites are covered in Getting Started with Swing and Learning Swing with the NetBeans IDE.

A Visual Index to the Swing Components (Java Look and Feel)

A Visual Index to the Swing Components (Windows Look and Feel)

Before you get started, you may want to check out these pages (from the Graphical User Interfaces lesson in the Core trail) which have
Using Swing Components: Examples

The table that follows lists every example in the Using Swing Components lesson, with links to required files and to where each example is discussed. The first column of the table has links to JNLP files that let you run the examples using Java™ Web Start.

NOTE: Release 6.0 is required to run all applets and Java Web Start examples. Most examples will run on an earlier release but you must compile and run them locally.

To run an example using Java Web Start, click the [Launch] link in the first column of the table. The first time you run an example, there will be a delay while Java Web Start downloads the JAR file containing the class files for this lesson's examples. Afterward, the examples should execute more quickly.

Compiling and Running the Examples Locally

The second column in the table below has links to zip files for each demo that you can open and run in the NetBeans IDE. Refer to Running Tutorial Examples in NetBeans IDE for more information.
## Table of Examples

<table>
<thead>
<tr>
<th>Example</th>
<th>Zip File (contains all files necessary for the example plus NetBeans IDE project metadata)</th>
<th>Source Files (first file has the main method, except for examples that run only as applets)</th>
<th>Image and Other Files</th>
<th>Where Described</th>
</tr>
</thead>
<tbody>
<tr>
<td>BorderDemo</td>
<td><a href="BorderDemo">Launch</a> <a href="BorderDemo">Project</a></td>
<td>BorderDemo.java</td>
<td>wavy.gif</td>
<td>How to Use Borders</td>
</tr>
<tr>
<td>ButtonDemo</td>
<td><a href="ButtonDemo">Launch</a> <a href="ButtonDemo">Project</a></td>
<td>ButtonDemo.java</td>
<td>right.gif, middle.gif, left.gif</td>
<td>How to Use Buttons...</td>
</tr>
<tr>
<td>ButtonHtmlDemo</td>
<td><a href="ButtonHtmlDemo">Launch</a> <a href="ButtonHtmlDemo">Project</a></td>
<td>ButtonHtmlDemo.java</td>
<td>right.gif, middle.gif, left.gif</td>
<td>How to Use Buttons...</td>
</tr>
<tr>
<td>CheckBoxDemo</td>
<td><a href="CheckBoxDemo">Launch</a> <a href="CheckBoxDemo">Project</a></td>
<td>CheckBoxDemo.java</td>
<td>All of the images in the</td>
<td>How to Use Buttons...</td>
</tr>
</tbody>
</table>

YHL Inheritance and Polymorphism
Qt Reference Documentation (Open Source Edition)

Note: This edition is for the development of Free and Open Source software only, see Qt Commercial Editions.

Getting Started
- What's New in Qt 4.3
- How to Learn Qt
- Installation
- Tutorial and Examples
- Porting from Qt 3 to Qt 4

General
- About Qt
- About Trolltech
- Commercial Edition
- Open Source Edition
- Frequently Asked Questions

Developer Resources
- Mailing Lists
- Qt Community Web Sites
- Qt Quarterly
- How to Report a Bug
- Other Online Resources

API Reference
- All Classes
- Main Classes
- Grouped Classes
- Annotated Classes

Core Features
- Signals and Slots
- Object Model
- Layout Management
- Paint System

Key Technologies
- Multithreaded Programming
- Main Window Architecture
- Rich Text Processing
- Model/View Programming
Qt Examples

This is the list of examples in Qt's examples directory. The examples demonstrate Qt features in small, self-contained programs. They are not all designed to be impressive when you run them, but their source code is carefully written to show good Qt programming practices. You can launch any of these programs from the Examples and Demos Launcher application.

If you are new to Qt, you should probably start by going through the Tutorial before you have a look at the Application example.

In addition to the examples and the tutorial, Qt includes a selection of demos that deliberately show off Qt's features. You might want to look at these as well.

One more valuable source for examples and explanations of Qt features is the archive of the Qt Quarterly.

In the list below, examples marked with an asterisk (*) are fully documented. Eventually, all the examples will be fully documented, but sometimes we include an example before we have time to write about it, because someone might need it right now.

Categories:
now.

Categories:

- ActiveQt
- Desktop
- Dialogs
- Drag and Drop
- Graphics View
- Item Views
- Layouts
- Main Windows
- Network
- OpenGL
- Painting
- Qt Assistant
- Qt Designer
- Qt Linguist
- Qtopia Core
- Qt Script
- Rich Text
- SQL
- Threads
- Tools
Dialogs

- Class Wizard
- Config Dialog
- Extension
- Find Files
- License Wizard
- Standard Dialogs
- Tab Dialog
- Trivial Wizard

Drag and Drop

- Draggable Icons
- Draggable Text
- Drop Site
- Fridge Magnets
- Drag and Drop Puzzle

Graphics View
Find Files Example

Files:

- dialogs/findfiles/window.cpp
- dialogs/findfiles/window.h
- dialogs/findfiles/main.cpp

The Find Files example shows how to use `QProgressDialog` to provide feedback on the progress of a slow operation. The example also shows how to use `QFileDialog` to facilitate browsing, how to use `QTextStream`'s streaming operators to read a file, and how to use `QTableWidget` to provide standard table display facilities for applications.
Find Files

Named: *.cpp
Containing text:
In directory: /home/qt/examples/dialogs/findfiles

<table>
<thead>
<tr>
<th>File Name</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>main.cpp</td>
<td>1 KB</td>
</tr>
<tr>
<td>window.cpp</td>
<td>6 KB</td>
</tr>
</tbody>
</table>

2 file(s) found
ECE 462
Object-Oriented Programming using C++ and Java

Development Environment

Yung-Hsiang Lu
yunglu@purdue.edu
Demonstrations

- develop C++ project in Eclipse
- develop Java project in Eclipse
- develop Java project in Netbeans (in handout)
- compile / execute C++ program in Linux shell
- compile / execute Java program in Linux shell

- Program: Person and Student classes.
Set up Execution Environment

- You can use Netbeans or Eclipse for developing Java or C++ projects.
- Please remember to use your ee462xxx account in MSEE190. Your personal Purdue account will not work.
- Do not use the ee462xxx account for any other purpose.
- After the final exam, the account will be erased and the password will be reset.
Install Eclipse at Your Own Computer
Eclipse - an open development platform

Eclipse is an open source community whose projects are focused on building an open development platform comprised of extensible frameworks, tools and runtimes for building, deploying and managing software across the lifecycle. A large and vibrant ecosystem of major technology vendors, innovative start-ups, universities, research institutions and individuals extend, complement and support the Eclipse platform. **New to Eclipse?**

**Download Eclipse**

Eclipse is used for ...

- Enterprise Development
- Embedded + Device Development
- Rich Client Platform
- Application Frameworks
- Language IDE
The Official Eclipse FAQs

(Redirected from Eclipse FAQs)

Contents [hide]

1 Part I -- The Eclipse Ecosystem
   1.1 The Eclipse Community
   1.2 Getting Started
   1.3 Java Development in Eclipse
   1.4 Plug-In Development Environment

2 Part II -- The Rich Client Platform
   2.1 All about Plug-ins
   2.2 Runtime Facilities
   2.3 Standard Widget Toolkit (SWT)
   2.4 JFace
FAQ How is Eclipse licensed?

Getting Started

Eclipse can be seen as a very advanced Java program. Running Eclipse may sound simple—simply run the included eclipse.exe or eclipse executable—yet in practice, you may want to tweak the inner workings of the platform. First, Eclipse does not come with a Java virtual machine (JVM), so you have to get one yourself. Note that Eclipse 3.0 needs a 1.4-compatible Java runtime environment (JRE).

To use Eclipse effectively, you will need to learn how to make Eclipse use a specific JRE. In addition, you may want to influence how much heap Eclipse may allocate, where it loads and saves its workspace from, and how you can add more plug-ins to your Eclipse installation.

This chapter should get you going. We also included some FAQs for the individual plug-ins that are distributed with eclipse and are currently installed.
Eclipse - an open development platform

Eclipse is an open source community whose projects are focused on building an open development platform comprised of extensible frameworks, tools and runtimes for building, deploying and managing software across the lifecycle. A large and vibrant ecosystem of major technology vendors, innovative start-ups, universities, research institutions and individuals extend, complement and support the Eclipse platform. New to Eclipse?

Download Eclipse

Eclipse is used for ...
Eclipse Downloads

To download Eclipse, select a package below or choose one of the third party Eclipse distros. **You will need a Java runtime environment (JRE) to use Eclipse (Java 5 JRE recommended).** All downloads are provided under the terms and conditions of the Eclipse Foundation Software User Agreement unless otherwise specified.

Problems extracting the ZIP file? Please read these Known Issues.

**Eclipse Europa Fall Maintenance Packages - Windows** (compare packages)

Eclipse IDE for Java Developers - Windows (78 MR)
YHL Development Environment
<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>Type</th>
<th>Data Modified</th>
</tr>
</thead>
<tbody>
<tr>
<td>eclipse-SDK-3.3.1.1-extractor.exe</td>
<td></td>
<td>EXE</td>
<td>2019-01-01 00:01:10 PM</td>
</tr>
<tr>
<td>eclipse-SDK-3.3.1.1-win32.zip</td>
<td></td>
<td>ZIP</td>
<td>2019-01-01 00:01:10 PM</td>
</tr>
</tbody>
</table>

- File and Folder Tasks:
  - Rename this file
  - Move this file
  - Copy this file
  - Publish this file to the Web
  - E-mail this file
  - Delete this file

- Other Places:
  - yunhui
  - My Documents
  - Shared Documents
  - My Computer
  - My Network Places

- Details:
  - Open
    - Browse with Paint Shop Pro 8
    - Explore
    - Extract to...
      - Extract to here
      - Extract to folder C:\yunhui\eclipse\eclipse-SDK-3.3.1.1-win32
      - Create Self-extractor (.exe)
      - E-Mail eclipse-SDK-3.3.1.1-win32.zip
  - Scan for viruses...
    - Open with
    - E-mail with Yahoo!
  - Send To
    - Cut
    - Copy
    - Create Shortcut
    - Delete
    - Rename
    - Properties
ready to run
no need to install anything
Workspace Launcher

Select a workspace

Eclipse SDK stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: C:\yunglu\eclipse\workspace

Use this as the default and do not ask again

OK  Cancel
Feature Updates

Choose the way you want to search for features to install:

- Search for updates of the currently installed features
  Select this option if you want to search for updates of the features you already have installed.

- Search for new features to install
  Select this option if you want to install new features from existing or new update sites. Some sites may already be available. You can add new update site URLs to the search.
The content of the update site "Europa Discovery Site" is also available on the mirrors listed below. You may choose to select a mirror instead of the default update site.

[United States] OSU Open Source Lab (http)
[United States] Computer Action Team - Portland State University (http)
[United States] Columbia University (http)
[United States] Georgia Tech. Software Library (http)
[United States] Indiana University (http)
[United States] Rochester Institute of Technology, Department of Computer Science
[United States] Calvin College (http)
[United States] Unixheads.org (http)
[Canada] Groupe d'utilisateurs de Linux de l'UdeS (http)
[Canada] University of Waterloo Computer Science Club (http)
[China] Actuate Shanghai (http)
[Israel] NSA Internet & Security Ltd. (http)
[Japan] Japan Advanced Institute of Science and Technology (http)
[Korea, Republic Of] Daum Communications Corp. (http)
[Korea, Republic Of] KAIST (http)

Automatically select mirrors
Search Results
Select features to install from the search result list.

Select the features to install:

- Europa Discovery Site
- C and C++ Development
- Charting and Reporting
- Communications
- Database Development
- Enabling Features
- Graphical Editors and Frameworks
- Java Development
- Models and Model Development
- Mylyn
- Other Tools

This Europa Discovery Site contains a number of Eclipse based projects released simultaneously, June 2007.

1 of 145 selected.
- Show the latest version of a feature only
- Filter features included in other features on the list

< Back  Next >  Finish  Cancel
Feature License

Some of the features have license agreements that you need to accept before proceeding with the installation.

**ECLIPSE FOUNDATION SOFTWARE USER AGREEMENT**
March 17, 2005

**Usage Of Content**

THE ECLIPSE FOUNDATION MAKES AVAILABLE SOFTWARE, DOCUMENTATION, INFORMATION AND/OR OTHER MATERIALS FOR OPEN SOURCE PROJECTS (COLLECTIVELY "CONTENT"). USE OF THE CONTENT IS GOVERNED BY THE TERMS AND CONDITIONS OF THIS AGREEMENT AND/OR THE TERMS AND CONDITIONS OF LICENSE AGREEMENTS OR NOTICES INDICATED OR REFERENCED BELOW. BY USING THE CONTENT, YOU AGREE THAT YOUR USE OF THE CONTENT IS GOVERNED BY THIS AGREEMENT AND/OR THE TERMS AND CONDITIONS OF ANY APPLICABLE LICENSE AGREEMENTS.

**I accept the terms in the license agreement**

**I do not accept the terms in the license agreement**
**Installation**

The following features will be installed. You can select a feature and change the location where the feature will be installed.

### Features to install:

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Feature Version</th>
<th>Feature Size</th>
<th>Installation Directory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eclipse C/C++ Developer</td>
<td>4.0.1.200709241202</td>
<td>Unknown</td>
<td>/C:/yunglu/eclipse/eclipse3311/</td>
</tr>
</tbody>
</table>

**Install Location:**  
C:/yunglu/eclipse/eclipse3311

**Required space:** Unknown  
**Free space:** 4.97 GB
Verification

Feature Verification

You are about to install a signed feature.
You may choose to install the feature or cancel its installation.

One of the certificates used to authenticate this feature was recognized.
The provider of this feature has been validated by a trusted third party.

Certificate:
File signed by: CN="Eclipse Foundation, Inc.", OU=Digital ID Class 3 - Java Object Signing, O="Eclipse Foundation, Inc.", L=Ottawa, ST=Ontario, C=CA
Valid between "Apr 13, 2006" and "Apr 14, 2009".
Valid certificate.

According to:
CN=VeriSign Class 3 Code Signing 2004 CA, OU=Terms of use at https://www.verisign.com/rpa (c)04, OU=VeriSign Trust Network, O="VeriSign, Inc.", C=US
Valid between "Jan 28, 1996" and "Aug 1, 2028".
Valid certificate.

Feature name: Eclipse C/C++ Development Tools
Feature Identifier: org.eclipse.cdt_4.0.1.200709241202
Provider: Eclipse.org
File Identifier: org.eclipse.cdt_4.0.1.200709241202

Install  Install All  Cancel
It is recommended you restart the Eclipse SDK for the changes to take effect, but it may be possible to apply the changes to the current configuration without restarting. Would you like to restart now?
Overview

The Eclipse software development kit is the development environment used to develop plug-ins for the Eclipse platform. It provides first-class Java programming tools, and plug-in development tools for building Eclipse-based applications and extensions.

C/C++ Development
Get familiar with the C/C++ Development Tools (CDT)

Java development
Get familiar with developing Java programs using Eclipse

Workbench basics
Learn about basic Eclipse workbench concepts

Eclipse plug-in development
Learn how to extend Eclipse by building new plug-ins

Team support
Find out how to collaborate with other
C/C++ Development User Guide

The C/C++ Development Toolkit (CDT) is a collection of Eclipse-based features that provides the capability to create, edit, navigate, build, and debug projects that use C and/or C++ as a programming language.

The CDT does not include the necessary compilers and debuggers to convert C/C++ code into executable programs and to debug those programs, but it does provide the frameworks that allow such tools to be integrated in a consistent fashion. This allows you to mix and match such tools depending on your project requirements.

Often, commercial distributions of the CDT include the necessary tools and integrations. If yours does not, the base CDT does provide support for integration with the GNU tools for build and debug. Please see the Before you begin section for installation instructions.

Before you begin
Getting Started
Concepts
Tasks
Reference
What's new
Overview

The Eclipse software development kit is the development environment used to develop plug-ins for the Eclipse platform. It provides first-class Java programming tools, and plug-in development tools for building Eclipse-based applications and extensions.

- **C/C++ Development**
  - Get familiar with the C/C++ Development Tools (CDT)

- **Java development**
  - Get familiar with developing Java programs using Eclipse

- **Workbench basics**
  - Learn about basic Eclipse workbench concepts

- **Eclipse plug-in development**
  - Learn how to extend Eclipse by building new plug-ins

- **Team support**
  - Find out how to collaborate with other
Develop C++ Projects in Eclipse
C++ Project
Create C++ project of selected type

Project name: MyFirst

Use default location
Location: C:/yunglu/eclipse/workspace/MyFirst

Project types:
- Executable
- Shared Library
- Static Library
- Makefile project

Toolchain:
- Cygwin GCC

Show project types and toolchains only if they are supported on the platform
Open Associated Perspective?

This kind of project is associated with the C/C++ perspective. Do you want to open this perspective now?

- Remember my decision

Yes No
Create a new C++ class.

**Source Folder:** MyFirst

**Namespace:**

**Class Name:** Person

<table>
<thead>
<tr>
<th>Base Classes:</th>
<th>Name</th>
<th>Access</th>
<th>Virtual</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method Stubs:</th>
<th>Name</th>
<th>Access</th>
<th>Virtual</th>
<th>Inline</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Constructor</td>
<td>public</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td>Destructor</td>
<td>public</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

**Use Default:**

**Header:** Person.h

**Source:** Person.cpp

**Finish** | **Cancel**
The code here is automatically generated.
Create a new source file.

Source Folder: MyFirst
Source File: main.cpp
```cpp
#include "Person.h"
#include "Student.h"
#include <iostream>

using namespace std;

int main(int argc, char * argv[])
{
    Person p1("Johnson", "Tom");
    p1.print();
    Student s1("Smith", "Mary", "Purdue", "ECE");
    s1.print();
    return 0;
}
```
```cpp
#ifndef PERSON_H
#define PERSON_H
#include <string>
using namespace std;

class Person
{
public:
    Person(string ln, string fn);
    // use at least two letters for a variable name
    virtual ~Person();
    void print();
protected:
    const string p.lastName;
    const string p.firstName;
};
#endif /*PERSON_H */
```
```cpp
#include "Person.h"
#include <iostream>

Person::Person(string ln, string fn) :
    p_lastName(ln), p_firstName(fn) {
}

Person::~Person() {
}

void Person::print() {
    cout << "last name: " << p_lastName << endl;
    cout << "first name: " << p_firstName << endl;
}
```
```cpp
#ifndef STUDENT_H_
#define STUDENT_H_
#include "Person.h"
#include <string>
using namespace std;

class Student: public Person
{
public:
    Student(string ln, string fn, string sch, string maj);
    virtual ~Student();
    void print(); // same name in Person, override
private:
    string s_school;
    string s_major;
};

#endif /* STUDENT_H_ */
```
```cpp
#include "Student.h"
#include <iostream>

Student::Student(string ln, string fn, string sch, string maj):
    Person(ln, fn), s_school(sch), s_major(maj) {
}

Student::~Student() {
}

void Student::print() {
    Person::print();
    cout << "school: " << s_school << endl;
    cout << "major: " << s_major << endl;
}
```
```cpp
#include "Student.h"
#include <iostream>

Student::Student(string ln, string fn, string sch, string maj):
    Person(ln, fn), s_school(sch), s_major(maj) {
}

Student::~Student() {
}
```

```plaintext
last name: Johnson
first name: Tom
last name: Smith
first name: Mary
school: Purdue
major: ECE
```
Cannot change the name.

```cpp
#include "Person.h"
#include "Student.h"
#include <iostream>

using namespace std;

int main(int argc, char * argv[])
{
    Person p1("Johnson", "Tom");
    p1.print();
    p1.p_firstName = "John";
    Student s1("Smith", "Mary", "Purdue", "ECE");
    s1.print();
}
```

C-Build [MyFirst]

../main.cpp: In function ‘int main(int, char**)’:
../Person.h:14: error: ‘const std::string Person::p_firstName’ is protected
../main.cpp:9: error: within this context
../main.cpp:9: error: passing ‘const std::string’ as ‘this’ argument of ‘std::basic_string<_CharT, _Traits, _Alloc>& std::basic_string<_CharT, _Traits, _Alloc>::operator=(const _CharT*)’ [with _CharT = char, _Traits = std::char_traits<char>, _Alloc = std::allocator<char>]' discards qualifiers
make: *** [main.o] Error 1
Develop Java Projects in Eclipse
Select a wizard
Create a Java project

Wizards:
- Class
- Interface
- Java Project
- Java Project from Existing Ant Buildfile
- Plug-in Project
- General
- C
- C++
- C++ Project
- Class
- Convert to a C/C++ Make Project
- File
- Folder
- Header File
Create a Java project in the workspace or in an external location.

Project name: FirstJava

Contents
- Create new project in workspace
- Create project from existing source

Directory: C:\yunglu\eclipse\workspace\FirstJava

JRE
- Use default JRE (Currently 'jre1.6.0_03')
- Use a project specific JRE:
- Use an execution environment JRE:

Project layout
- Use project folder as root for sources and class files
- Create separate folders for sources and class files

Working sets
- Add project to working sets

< Back Next > Finish Cancel
New Java Class

Java Class
Create a new Java class.

Source folder: FirstJava/src
Package: firstjava
Enclosing type:

Name: Student
Modifiers:
- public
- default
- private
- protected
- abstract
- final
- static

Superclass: java.lang.Object
Interfaces:

Which method stubs would you like to create?
- public static void main(String[] args)
- Constructors from superclass
- Inherited abstract methods

Do you want to add comments as configured in the properties of the current project?
- Generate comments

Finish  Cancel
Creating a new Java class:

- Source folder: FirstJava/src
- Package: firstjava
- Name: FirstJavaMain
- Modifiers: public
- Superclass: java.lang.Object
- Interfaces:
- Which method stubs would you like to create?
  - public static void main(String[] args)
- Do you want to add comments as configured in the properties of the current project?
  - Generate comments
```java
package firstjava;

/**
 * @author yunqiu
 */

class Person {
    final String p_lastName;
    final String p_firstName;

    public Person(String ln, String fn) {
        p_lastName = ln;
        p_firstName = fn;
    }

    public void print() {
        System.out.println("Last name: " + p_lastName);
        System.out.println("First name: " + p_firstName);
    }
}
```
package firstjava;

@Author yunqiu

public class FirstJavaMain {

    @param args

    public static void main(String[] args) {
        Person p1 = new Person("Johnson", "Tom");
        p1.print();
        Student s1 = new Student("Smith", "Mary", "Purdue", "ECE");
        s1.print();
    }
}
package firstjava;

* @author yunjun

public class Person {

    final String p_lastName;
    final String p_firstName;

    public Person(String ln, String fn) {
        p_lastName = ln;
        p_firstName = fn;
    }

    public void print() {
        System.out.println("Last name: "+ p_lastName);
        System.out.println("First name: "+ p_firstName);
    }
}
```java
package firstjava;

/* @author yunglu */

public class Student extends Person {
    String s_school;
    String s_major;

    public Student(String ln, String fn, String sch, String maj) {
        super(ln, fn);
        s_school = sch;
        s_major = maj;
    }

    public void print() {
        super.print();
        System.out.println("School: " + s_school);
        System.out.println("Major: " + s_major);
    }
}
```
```java
package firstJavaMain;

public class FirstJavaMain {
    public static void main(String[] args) {
        Person p1 = new Person("Johnson", "Tom");
        p1.print();
        Student s1 = new Student("Smith", "Mary", "Purdue", "ECE");
        s1.print();
    }
}
```
```java
package firstjava;

/*
 * @author yungly
 *
 * @param FirstJavaMain (public class FirstJavaMain
 *
 * @param @param args
 *
 * public static void main(String[] args) {
 *     Person p1 = new Person("Johnson", "Tcm");
 *
 */

Last name: Johnson
First name: Tom
Last name: Smith
First name: Mary
School: Purdue
Major: ECE
```java
package firstjava;

public class FirstJavaMain {

    public static void main(String[] args) {
        Person p1 = new Person("John", "Tom");
    }

    public class Person {
        String name;
        String lastName;
        String school;
        String major;

        public Person(String name, String lastName) {
            this.name = name;
            this.lastName = lastName;
        }

        public String getName() {
            return name;
        }

        public String getLastName() {
            return lastName;
        }

        public void printName() {
            System.out.println(name);
        }
    }
}
```
package firstjava;

public class FirstJavaMain {
    public static void main(String[] args) {
        Person p1 = new Person("Johncn", "Tom");
    }
}

Last name: Johnson
First name: Tom
Last name: Smith
First name: Mary
School: Purdue
Major: ECE
## C++ and Java Syntax

<table>
<thead>
<tr>
<th>C++</th>
<th>Java</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>int main(int argc, char * argv[])</code></td>
<td><code>public static void main(String[] args) {</code></td>
</tr>
<tr>
<td><code>Person p1(&quot;Johnson&quot;, &quot;Tom&quot;);</code></td>
<td><code>Person p1 = new Person(&quot;Johnson&quot;, &quot;Tom&quot;);</code></td>
</tr>
<tr>
<td><code>p1.print();</code></td>
<td><code>p1.print();</code></td>
</tr>
<tr>
<td><code>class Person</code></td>
<td><code>public class Person {</code></td>
</tr>
<tr>
<td><code>{</code></td>
<td><code>    public Person(String ln, String fn) {</code></td>
</tr>
<tr>
<td><code>public:</code></td>
<td><code>    }</code></td>
</tr>
<tr>
<td><code>    Person(string ln, string fn);</code></td>
<td></td>
</tr>
<tr>
<td><code>const string p_lastName;</code></td>
<td><code>final String p_lastName;</code></td>
</tr>
<tr>
<td><code>class Student: public Person</code></td>
<td><code>class Student extends Person</code></td>
</tr>
</tbody>
</table>
Version Control

Every assignment and every lab exercise **must** be submitted using the **CVS repository**.

Submission of the source code only will **not be graded**.
CVS - Concurrent Versions System

Introduction to CVS

CVS is a version control system, an important component of Source Configuration Management (SCM). Using it, you can record the history of sources files, and documents. It fills a similar role to the free software RCS, PRCS, and Aegis packages.

CVS is a production quality system in wide use around the world, including many free software projects.

While CVS stores individual file history in the same format as RCS, it offers the following significant advantages over RCS:

- It can run scripts which you can supply to log CVS operations or enforce site-specific policies.
- Client/server CVS enables developers scattered by geography or slow modems to function as a single team. The version history is stored on a single central server and the client machines have a copy of all the files that the developers are working on. Therefore, the network between the client and the central server must be fast enough to transfer machine data but very slow compared to transferring individual files.
Update and Commit

- **update** = pull the changes made by your teammates from the repository
- **commit** = push your changes to the repository so that your teammates can see
- **update and commit** *often* ⇒ keep your files and the repository "in sync"
- focus on one task at a time, finish it and commit it
- commit after adding a feature or fixing a bug
- use meaningful comments to indicate the progression of the project
Prepare a Repository

• enter your 462 account
• make a directory called "projects"
• enter the "projects" directory
• make a directory called "CVSROOT"
Enter Repository Location Information
Define the location and protocol required to connect with an existing CVS repository.

**Location**
- **Host:** mseec190pc5.ecn.purdue.edu
- **Repository path:** /home/shay/a/ee462b30/projects

**Authentication**
- **User:** ee462b30
- **Password:** ********

**Connection**
- **Connection type:** extssh
- **Use default port**
- **Use port:**

---

Saved passwords are stored on your computer in a file that is difficult, but not impossible, for an intruder to read.

Configure connection preferences...
Enter Module Name

Select the name of the module in the CVS repository.

- Use project name as module name
- Use specified module name: [text box]
- Use an existing module (this will allow you to browse the modules in the repository)

Buttons:
- < Back
- Next
- Finish
- Cancel
The authenticity of host `insee190pc9.ecn.purdue.edu` can't be established.
Are you sure you want to continue connecting?
Share Project Resources

Review and commit resources to the repository. Use the context menu to perform additional operations on the resources.

CVS (FirstJava)

- FirstJava [msee190pc9.ecn.purdue.edu]
  - src
    - (default package)
    - firstJava
      - FirstJavaMain.java (ASCII -kky)
      - Person.java (ASCII -kky)
      - Student.java (ASCII -kky)
  - .classpath (ASCII -kky)
  - .project (ASCII -kky)

Launch the Commit wizard

Finish  Cancel  Next  < Back
Commit

Enter a comment for the commit operation.

create the repository of my first Java project

<Choose a previously entered comment>

Configure Comment Templates...

Changes

- FirstJava [msee190pc9.ecn.purdue.edu]
  - src
    - src/firstjava
      - FirstJavaMain.java (ASCII-kkv)
      - Person.java (ASCII-kkv)
      - Student.java (ASCII-kkv)
    - .classpath (ASCII-kkv)
Version 1.1
">" means the file has been changed
Update: check whether your teammate has made any changes.
Commit: allow your teammate to see the changes you have made.
Always update before commit.

Otherwise, you may overwrite your teammate's changes.
It is recoverable but you should avoid such a problem.
Only the changed file is shown.
$Log$ is replaced by the commit history.

```java
/**
 * @author
 */

// This is a comment. The Log tag will be automatically
// history of changes
// $Log: Person.java,v $
// Revision 1.2 2008/02/16 20:25:02 ee462b30
// Add a comment in Person.java

public class Person {
    final String p_LastName;
    final String p_FirstName;

    public Person(String in, String fn) {
        p_LastName = in;
        p_FirstName = fn;
    }
}
```
Develop Java Projects in Netbeans
Install Netbeans at Your Own Computer
http://www.netbeans.org/
Welcome to the NetBeans IDE 6.0 Installer

The installer will install the NetBeans IDE with the following packs and runtimes.
Click Customize to select the packs and runtimes to install.

Base IDE
Java SE
Web & Java EE
Mobility
UML
Ruby
C/C++

Installation Size: 359.5 MB
License Agreement

Please read the following license agreement carefully.

NETBEANS IDE 6.0

Please review the complete list of open-source licenses governing software included in the Product. They can be found in the THIRDPARTYLICENSE.txt file. Please review the list of libraries and licenses provided for use. This license file contains five distinct licenses.

Unless specified below, the use of NetBeans IDE 6.0 and components from the GlassFish runtime are governed by the terms of either the GNU General Public License Version 2 with Classpath Exception or the Common Development and Distribution License. The Product also contains components from OpenJDK, which are governed exclusively by the terms of the GPLv2 with Classpath Exception.

LICENSE #1: GPLv2 with Classpath Exception.

I accept the terms in the license agreement.
NetBeans IDE 6.0 Installation
Choose the installation folder and JDK™ for the NetBeans IDE.

Install the NetBeans IDE to:
C:\yunglu\Netbeans60

JDK™ for the NetBeans IDE:
C:\Program Files\java\jdk1.5.0_06
NetBeans IDE Installer

Summary
Click Install to start the installation.

NetBeans IDE Installation Folder:
C:\yungui\Netbeans60

Total Installation Size:
359.5 MB
Installation

Please wait while the installer installs NetBeans IDE and runtimes.

Installing Web & Java EE...

Extracting C:\yunglu\Netbeans60\visualweb1\modules\ext\webui-sf-suntheme.jar.pack.gz
Installation completed successfully.

To launch the IDE, use either the Start menu or the NetBeans desktop icon.

To change installed components and add NetBeans plugins, use Plugin Manager that is an integral part of NetBeans IDE.
NetBeans IDE 6.0

NetBeans IDE and NetBeans Platform are based on software from netbeans.org, which has been dual licensed under the Common Development and Distribution License (CDDL) and the GNU General Public License version 2 with Classpath exception. For more information, please visit www.netbeans.org.

Loading modules...
YHL Development Environment 101
Creates a new Java SE application in a standard IDE project. You can also generate a main class in the project. Standard projects use an IDE-generated Ant build script to build, run, and debug your project.
```java
package javaapplication14;

/**
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

public class Main {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
    }
}
```
```java
public static void main(String[] args) {
    // TODO code application logic here
}
```
package javaapplication14;

/**
 * @author yunglu
 */

public class Person {

    public Person(String ln, String fn) {
        p_lastName = ln;
        p_firstName = fn;
    }

    public void print() {
        System.out.println("Last name: " + p_lastName);
        System.out.println("First name: " + p_firstName);
        final String p_lastName;
        final String p_firstName;
    }
}
```java
import java.util.Scanner;

public class Student {
    private String firstName;
    private String lastName;

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter your first name: ");
        firstName = scanner.nextLine();
        System.out.println("Enter your last name: ");
        lastName = scanner.nextLine();

        System.out.println("First name: " + firstName);
        System.out.println("Last name: " + lastName);
    }
}
```
Attributes can be declared at the end of the class.
```java
public class Student extends Person {

    String s_school;
    String s_major;

    public Student(String ln, String fn, String sch, String maj) {
        super(ln, fn);
        s_school = sch;
        s_major = maj;
    }

    @Override
    public void print() {
        super.print();
        System.out.println("School: " + s_school);
        System.out.println("Major: " + s_major);
    }
}
```
```java
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package javaapplication14;

/**
 * @author yungiu
 */

class Main {

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

public static void main(String[] args) {
    // TODO code application logic here
    Person p1 = new Person("Johnson", "Tom");
    p1.print();
    Student s1 = new Student("Smith", "Mary", "Purdue", "ECE");
    s1.print();
}
```
package javaapplication14;

public class Main {
    public static void main(String[] args) {
        Student s1 = new Student("Smith", "Mary", "Purdue", "SCE");
        s1.print();
    }
}

public class Student {
    String firstName, lastName, school, department;

    public Student(String first, String last, String sch, String dept) {
        firstName = first;
        lastName = last;
        school = sch;
        department = dept;
    }

    public void print() {
        System.out.println("First Name: "+firstName +
                          " Last Name: "+lastName +
                          " School: "+school +
                          " Department: "+department);
    }
}

/ * To change code, right-click the Java source file in JavaApplication14/
/ * and open its properties. In the properties window, change the code folder
/ * to the new code folder.
/ */

package javaapplication14;

public class Main {
    public static void main(String[] args) {
        Student s1 = new Student("Smith", "Mary", "Purdue", "SCE");
        s1.print();
    }
}

public class Student {
    String firstName, lastName, school, department;

    public Student(String first, String last, String sch, String dept) {
        firstName = first;
        lastName = last;
        school = sch;
        department = dept;
    }

    public void print() {
        System.out.println("First Name: "+firstName +
                          " Last Name: "+lastName +
                          " School: "+school +
                          " Department: "+department);
    }
}
YHL Development Environment

JavaApplication14: Main.java

/* To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package javaapplication14;

/*
*/

init:
deps-jar:
Created dir: C:\yunglu\java\JavaApplication14\build\classes
Compiling 1 source file to C:\yunglu\java\JavaApplication14\build\classes
compile-single:
run-single:
last name: Johnson
first name: Tom
last name: Smith
first name: Mary
School: Purdue
Major: ECE
BUILD SUCCESSFUL (total time: 8 seconds)
```java
import this template, choose Tools | Templates
the template in the editor.

Main.java;

gungiu

public class Main {
    // am args the command line arguments
    static void main(String[] args) {
        // TODO code application logic here
        Person pl = new Person("Johnson", "Tom");
        print();
        Person p2 = new Person("Matthew", "Sarah", "Purdue", "SCE");
    }
}
```
Import Project Options

CVS Root
Specify location of CVS repository defined by CVS root.

CVS Root: p30@msee190pc5.ecn.purdue.edu:/home/shay/a/ee462b30/projects
(:ext:username@hostname:/repository_path)

☐ Use Internal SSH
Password: *********
☐ Remember Password
Proxy Configuration...

☐ Use External Shell
Shell Command: Browse...

< Back Next > Finish Cancel Help
package javaapplication14;

/*
 * This is a comment. The Log tag will be automatically updated
 * history of changes
 * $Log$
 *|
 */

/**...*/

public class Main {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        Person p1 = new Person("Johnson", "Tom");
        p1.print();
        Student s1 = new Student("Smith", "Mary", "Purdue", "EC");
        s1.print();
    }
}

Main.java saved.
JavaApplication14 - NetBeans IDE 6.0

Projects
- JavaApplication14
- Source Packages
- Test Packages
- Libraries
- Test Libraries
- New
- Build
- Clean and Build
- Clean
- Generate Javadoc
- Run
- Debug
- Profile
- Test
- Set Configuration
- Set as Main Project
- Open Required Projects
- Close
- Rename...
- Move...
- Copy...
- Delete
- Find...
- Reverse Engineer...
- CVS
- Local History
- Properties

Show Changes
- Diff
- Update
- Update with Dependencies
- Commit...
- Tag...
- Branch...
- Switch to Branch...
- Merge Changes from Branch...
- View Revision...
- Search History...
- Revert Modifications
- Resolve Conflicts...
- Exclude from Commit

javaapplication14:

```java
public class Main {
    public static void main(String[] args) {
        Student s1 = new Student("Smith", "Mary", "Purdue", "ECB");
        Student s2 = new Student("Johnson", "Tom");
        // logic here
    }
}
```

Change this template, choose Tools | Templates
Open the template in the editor.

The tag will be automatically updated.

File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help

YHL Development Environment 122
/* To change this template, choose Tools | Templates
* and open the template in the editor.
*/

package javaapplication14;

/**
* This is a comment. The Log tag will be automatically updated
* history of changes
* $Log$
*/

public class Main {
```java
package javaapplication14;

public class Main {

    // This is a comment. The history of changes can be seen with $Log$
    // @author yunglu

    public static void main(String[] args) {
        // To change this template, choose Tools | Templates | Java EE | Java Application
        // and open the template in the editor.
    }
}
```
YHL Development Environment 125
Commit Message:

add a comment, CVS replaces Log by this commit comment

Files to Commit:

<table>
<thead>
<tr>
<th>File</th>
<th>Status</th>
<th>Content</th>
<th>Repository path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Man.java</td>
<td>Locally</td>
<td>Commit</td>
<td>...n14/src/javaapplication14</td>
</tr>
</tbody>
</table>

Commit | Cancel | Help
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package javaapplication14;

/*
 * This is a comment. The Log tag will be automatically updated
 * history of changes
 * @Log: Main.java, v 
 *
 * Revision 1.2  2008/02/16 20:52:04  ee462b30
 * add a comment, CVS replaces Log by this commit comment
 * 
 * 
 * */

/**
 *
 * @author yungkin
 */

public class Main {

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

9:12   INS
Compile / Execute C++ Programs in Linux Shell
mkdir cpp
setenv CVSROOT ~/.projects/
cd cpp/
cvs checkout MyFirst
(cvs checkout: Updating MyFirst
U MyFirst/.cproject
U MyFirst/.project
U MyFirst/Person.cpp
U MyFirst/Person.h
U MyFirst/Student.cpp
U MyFirst/Student.h
U MyFirst/main.cpp

[$(msee190pc9) ~/cpp/MyFirst/ ] ls
CVS/ main.cpp Person.cpp Person.h Student.cpp Student.h
[($msee190pc9) ~/cpp/MyFirst/ ]
ls
CVS/ main.cpp Person.cpp Person.h Student.cpp Student.h
which qmake
/home/shay/a/swtools/public/qt4.3.0/bin/qmake
qmake -version
QMake version 2.01a
Using Qt version 4.3.0 in /home/shay/a/swtools/public/qt4.3.0/lib
YHL Development Environment

 CVS/ main.cpp Person.cpp Person.h Student.cpp Student.h
 [(msee190pc9) ~/cpp/MyFirst/ ] which qmake
 /home/shay/a/sfwtools/public/qt4.3.0/bin/qmake
 [(msee190pc9) ~/cpp/MyFirst/ ] qmake -version
 QMake version 2.0La
 Using Qt version 4.3.0 in /home/shay/a/sfwtools/public/qt4.3.0/lib
 [(msee190pc9) ~/cpp/MyFirst/ ] qmake -project
 [(msee190pc9) ~/cpp/MyFirst/ ] qmake
 [(msee190pc9) ~/cpp/MyFirst/ ] make
 g++ -c -m64 -pipe -O2 -Wall -W -D_REENTRANT -DQT_NO_DEBUG -DQT_GUI_LIB
 -DQT_CORE_LIB -DQT_SHARED -I..//..//sfwtools/public/qt4.3.0/mkspecs/
 linux-g++-64 -I. -I..//..//sfwtools/public/qt4.3.0/include/qtCore -I.
 ..//..//sfwtools/public/qt4.3.0/include/qtCore -I..//..//sfwtools/public/
 qt4.3.0/include/qtGui -I..//..//sfwtools/public/qt4.3.0/include/qtGui
 -I..//..//sfwtools/public/qt4.3.0/include -I. -I. -o main.o
 main.cpp:6: warning: unused parameter 'argc'
 main.cpp:6: warning: unused parameter 'argv'
 g++ -c -m64 -pipe -O2 -Wall -W -D_REENTRANT -DQT_NO_DEBUG -DQT_GUI_LIB
 -DQT_CORE_LIB -DQT_SHARED -I..//..//sfwtools/public/qt4.3.0/mkspecs/
 linux-g++-64 -I. -I..//..//sfwtools/public/qt4.3.0/include/qtCore -I.
 ..//..//sfwtools/public/qt4.3.0/include/qtCore -I..//..//sfwtools/public/
 qt4.3.0/include/qtGui -I..//..//sfwtools/public/qt4.3.0/include/qtGui
 -I..//..//sfwtools/public/qt4.3.0/include -I. -I. -o Person.o
 g++ -c -m64 -pipe -O2 -Wall -W -D_REENTRANT -DQT_NO_DEBUG -DQT_GUI_LIB
lic/qt4.3.0/include/QtGui -I..../..../sfwtools/public/qt4.3.0/include/QtCore -I..../..../sfwtools/public/qt4.3.0/include -I. -I. -I. -o Person.o Person.cpp
/usr/bin/ld: warning: libstdc++.so.5, needed by /home/shay/a/sfwtools/public/qt4.3.0/lib/libQtGui.so, may conflict with libstdc++.so.6
[(msee190pc9) ~/cpp/MyQtGui.so, may conflict with libstdc++.so.6
[(msee190pc9) ~/cpp/MyFirst/]
MyFirst

last name: johnson
first name: Tom
last name: smith
first name: mary
school: purdue
major: ECE
[(msee190pc9) ~/cpp/MyFirst/]
Compile / Execute Java Programs in Linux Shell
mkdir java
setenv CVSROOT ~/projects/
cd java/
cvs checkout FirstJava
  cvs checkout: Updating FirstJava
  U FirstJava/.classpath
  U FirstJava/.project
  cvs checkout: Updating FirstJava/src
  cvs checkout: Updating FirstJava/src/firstjava
  U FirstJava/src/firstjava/FirstJavaMain.java
  U FirstJava/src/firstjava/Person.java
  U FirstJava/src/firstjava/Student.java
[(msee190pc9) ~/java/ ]
```
ls
CVS/ FirstJavaMain.java Person.java Student.java
```
Remove package firstjava; in the files
YHL Development Environment

```
((msee190pc9) ~/java/FirstJava/src/firstjava/ ] ls
CVS/ FirstJavaMain.java Person.java Student.java
((msee190pc9) ~/java/FirstJava/src/firstjava/ ] which javac
/home/shay/a/swtools/public/jdk1.6.0_02/bin//javac
((msee190pc9) ~/java/FirstJava/src/firstjava/ ] javac -version
javac 1.6.0_02
((msee190pc9) ~/java/FirstJava/src/firstjava/ ] javac Person.java
((msee190pc9) ~/java/FirstJava/src/firstjava/ ] javac Student.java
((msee190pc9) ~/java/FirstJava/src/firstjava/ ] javac FirstJavaMain.java

((msee190pc9) ~/java/FirstJava/src/firstjava/ ] ls
CVS/
FirstJavaMain.class Person.class Student.class
((msee190pc9) ~/java/FirstJava/src/firstjava/ ] java FirstJavaMain
Last name: Johnson
First name: Tom
Last name: Smith
First name: Mary
School: Purdue
Major: ECE
```

```
Self Test