INF 111 / CSE 121

Discussion Session
Week 2 - Fall 2007

Instructor: Michele Rousseau
TA: Rosalva Gallardo

Overview

- Introduction
  - Goals
  - Contact Information
  - Policies
  - Card Game
- Tools
  - Eclipse
  - JUnit

Goals

- Discuss details about the assignments
- Present tools
- Prepare for tests
- Review tests and assignments

Contact Information

Rosalva Gallardo

- Email: rgallard@uci.edu
- Office: DBH 5051
- Office hours: Monday 11am - 1pm
  - Please email me if you plan to drop by
- Comments about next discussion

Policies

- Discussion
  - Attend 1 discussion session.
  - Will not take attendance.
  - Turn off your cell phone.
  - Ask questions about assignments.
- Assignments
  - No late assignments.
  - Bring questions about the assignment to the discussion session.
  - Please do not wait until the last minute to ask questions about the assignments.

Policies

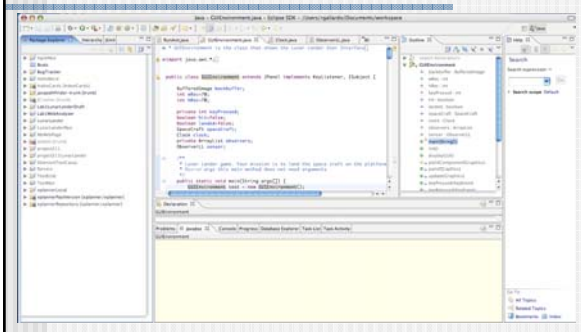
- Questions
  - Email, noteboard, office hours.
  - If the TA considers that an emails' reply is beneficial for all the class, she will reply to all or post it in the Noteboard.
  - An answer will be provided within 24 hours.
- Grading
  - Always check your partial grades.
  - If you have questions, please talk to the TA first, then with the instructor.
- Re-grade
  - Double check before you bring it.
  - Within 1 week, accompanied by a clear explanation of what needs to be reconsidered and why.

Card Game

I want to know about YOU

- Write the following information in the card:
  - Your Name
  - Underline how you want to be called
  - Your expectation for the discussion session

Eclipse
(Slides adapted from S.E.Sim)



Eclipse

- Eclipse is an IDE (Integrated Development Environment)
- It's an open source project
  - http://www.eclipse.org
  - Consortium of companies, including IBM
  - Launched in November 2001
- It's a framework for software tools ("plug-ins" in Eclipse terminology)
  - Main component is the workbench
  - Ships with two plug-ins JDT (Java Development Tools) and PDE (Plug-in Development Environment)

JDT

- JDT – Java Development Tool
- Includes a variety of programming tools
  - Editor with syntax highlighting
    - Content Assist
    - Quick Fix
  - Source code searching
  - Debugger
  - Refactoring
  - Code browser

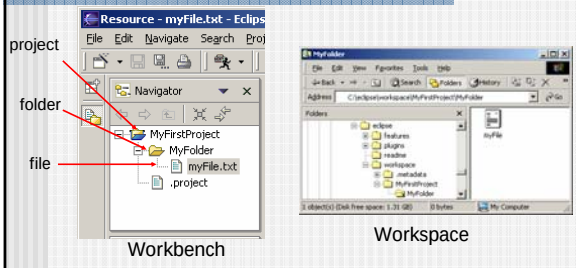
Eclipse Concepts

- Resources
- Perspectives
- Views

Resources in a Workbench

- When working with Eclipse, you work with its resources
- Resources are organized as a file/directory structure in the Workbench
  - They correspond to the actual files and directories in the Workspace
  - There are three different levels on resources:
    - Projects
    - Folders
    - Files

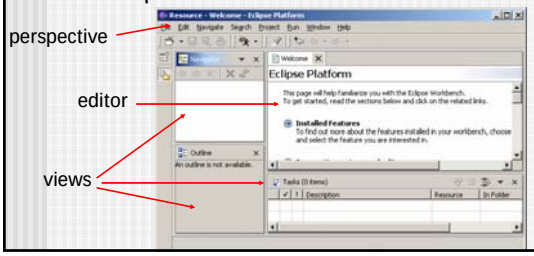
Organizing Resources



- It is possible to drag and drop resources directly between Workbench and the directory structure

Workbench Components

- Workbench contains perspectives
- Perspective contains views and editors

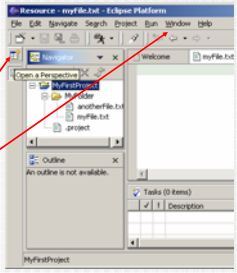


Perspectives

- Perspective defines initial layout of the views in the Workbench
- They are task oriented, i.e. they contain specific views for doing certain tasks:
  - Java Perspective for manipulating Java code
  - Resource Perspective for manipulating resources
  - Debug Perspective for debugging applications
- One Workbench window contains many perspectives

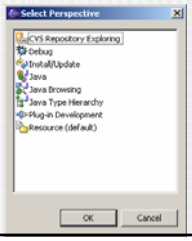
Opening Perspective

- Perspectives can be open by:
  - Clicking on a perspective shortcut button
  - Choosing Window → Open Perspective… menu option



Available Perspectives

- By default, the following perspectives are available in the Workbench:



Views…

- The main purpose of a view is:
  - To support editors
  - To provide alternative presentation and navigation in the Workbench
- Views can have their own menus and toolbars
  - Items available in menus and toolbars are available only in that view

…Views

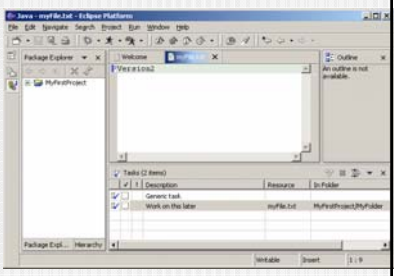
- Views can:
  - Appear on their own
  - Appear stacked with other views
- Layout of the views can be changed by clicking on the title bar and moving views
  - Single views can be moved together with other views
  - Stacked views can be moved to be single views

Perspectives Available for Java

- When developing Java code commonly used perspectives are:
  - Java Perspective
    - Designed for working with Java projects
  - Java Browsing Perspective
    - Designed for browsing structure of Java projects
  - Java Type Hierarchy Perspective
    - Designed for exploring type hierarchy
  - Debug Perspective
    - Designed for debugging Java programs

Java Perspective

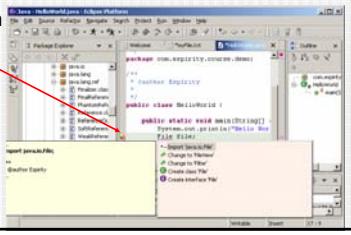
- Contains:
  - Editor area
  - Package Explorer View
  - Hierarchy View
  - Outline View
  - Tasks View



Using Quick Fix

- Useful if Java compiler shows errors
  - Gives options for fixing the errors
  - Activated through Edit → Quick Fix menu option

Error indication



Eclipse

- Useful links
  - Eclipse Glossary: http://www.eclipse.org/glossary.html
  - List of Plug-ins: http://www.eclipseplugincentral.com/

Eclipse

- Demo

JUnit

- JUnit is an open source testing framework for Java.
  - http://www.junit.org/
- It provides a common and reusable structure that is required for developing automate and repeatable unit tests for Java classes.
- It provides:
  - A base class called TestCase that can be extended to create series of tests for your classes.
  - An assertion library that can be used to evaluate the results of the tests.

Junit key classes

- TestCase - subclass contains your tests
- TestSuite - a composite of TestCases and/or TestSuites
- TestRunners - to run TestCases or TestSuites
- TestResult - collects results of multiple tests

How to use JUnit

```
Class C {
  method m1();
  method m2();
}

Class D {
  method m3();
  method m4();
}
```

```
Class CTest extends TestCase {
  method testM1();
  method testM2();
}

Class DTest extends TestCase {
  method testM3();
  method testM4();
}
```

- Each test class exercises one class in the system. Each test method exercises one method in the system. You also write additional test methods to exercise combinations of system methods.

How to use JUnit

- Each test method consists of a sequence of steps and some checks of the results.
- Once you have the unit tests written, you run them. You could run them directly from main(), but it is easier to use a test running utility.
  - Options: Junit TestRunners or the Ant Junit task.

Junit Test Runner Sequence

- Test runner is given a list of test classes
- For each test class
  - Create an instance of the test class
    - For each test*() method
      - Run setUp() method
        - Run test method steps and checks
        - If a check fails, an exception is thrown and the test method fails
- Test runner produces a report
- Some test runners work interactively

Junit Methods

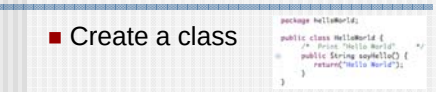
- **assertEquals(x,y)** - Test passes if x and y are equal
  - X and y can be primitives or any type with an appropiate equals method
  - Three argument versions exist for floating point numbers
- **assertFalse(b)** - Test passes if boolean value b is false
- **assertTrue(b)** - Test passes if boolean value b is tru
- **assertNull(o)** - Test passes if object o is null
- **assertNotNull(o)** - Test passes if object o is not null
- **assertSame(ox,oy)** - Test passes if ox and oy refer to the same object
- **assertNotSame(ox,oy)** - Test passes if ox and oy do not refer to the same object

JUnit and Eclipse

- The recent version of the Eclipse JDT already has JUnit Plug-in built in.
- The plug-in includes:
  - A wizard for assisting in creating a test case and test suite.
  - An environment for running test cases.

Testing Hello World with JUnit

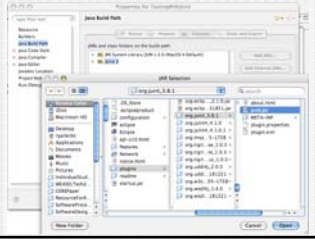
- Create a class



- Create a Test Case

Testing Hello World with JUnit

- Make sure Junit.jar is in the Build Path. Right Click on Project - > Properties



Testing HelloWorld with JUnit

- Run Test

Test Passed

Test did not pass

Junit

- Demo
  - HelloWorld
  - ShoppingCart (Adapted from Junit Primer)

Things to Notice

- setUp() method makes some variables that are used in the tests
  - Oficially called “fixtures”
- tearDown() frees memory, prevents results of one test from affecting the next
- Only the first failure in a test method is reported
  - Do not do too much in a single test
- Missing test cases: a new cart should be empty, add the same product twice, remove a product that was already removed, etc.

More Information

- Eclipse Help
  - Help -> Help Contents -> Java Development User Guide -> Getting Started -> Basic Tutorial -> Writing and running Junit tests
- JUnit Home Page
  - http://www.junit.org
- Junit Primer
  - http://www.clarkware.com/articles/JUnitPrimer.html