

Lab: Version Control System and SVN

NetDB

CS, NTHU,

Fall, 2013

Outline

- Why do we need version control?
- Introduction to SVN
- Using SVN Client in Eclipse
- Exercise and Homework Submission

Outline

- Why do we need version control?
- Introduction to SVN
- Using SVN Client in Eclipse
- Exercise and Homework Submission

Why Do We Need Version Control?

- A place to store the projects
- Synchronization between modifications made by different developers
- Even when you are developing a project along, SVN still helps in showing your revision history

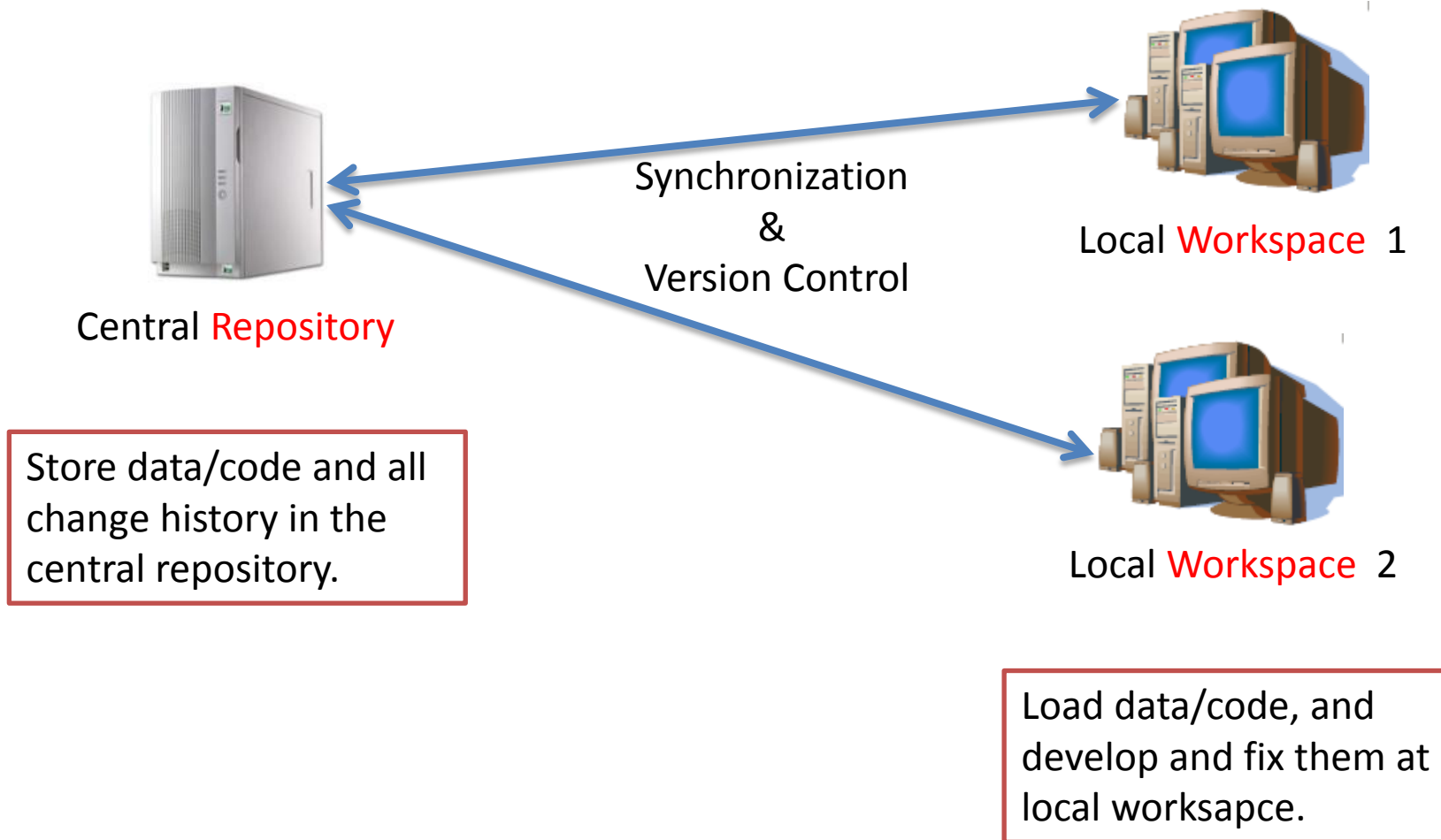
Outline

- Why do we need version control?
- **Introduction to SVN**
- Using SVN Client in Eclipse
- Exercise and Homework Submission

Introduction to SVN

- “Subversion” in short
- An open-source version control system
 - To record every changes made to files (data/code) and directories
 - Who made a change?
 - What has been changed?
 - When did they make it?
 - Why did they make it?
 - To allow the recovery of older versions
 - To trace the history of how your data/code changed
 - To collaboratively edit and share data/code

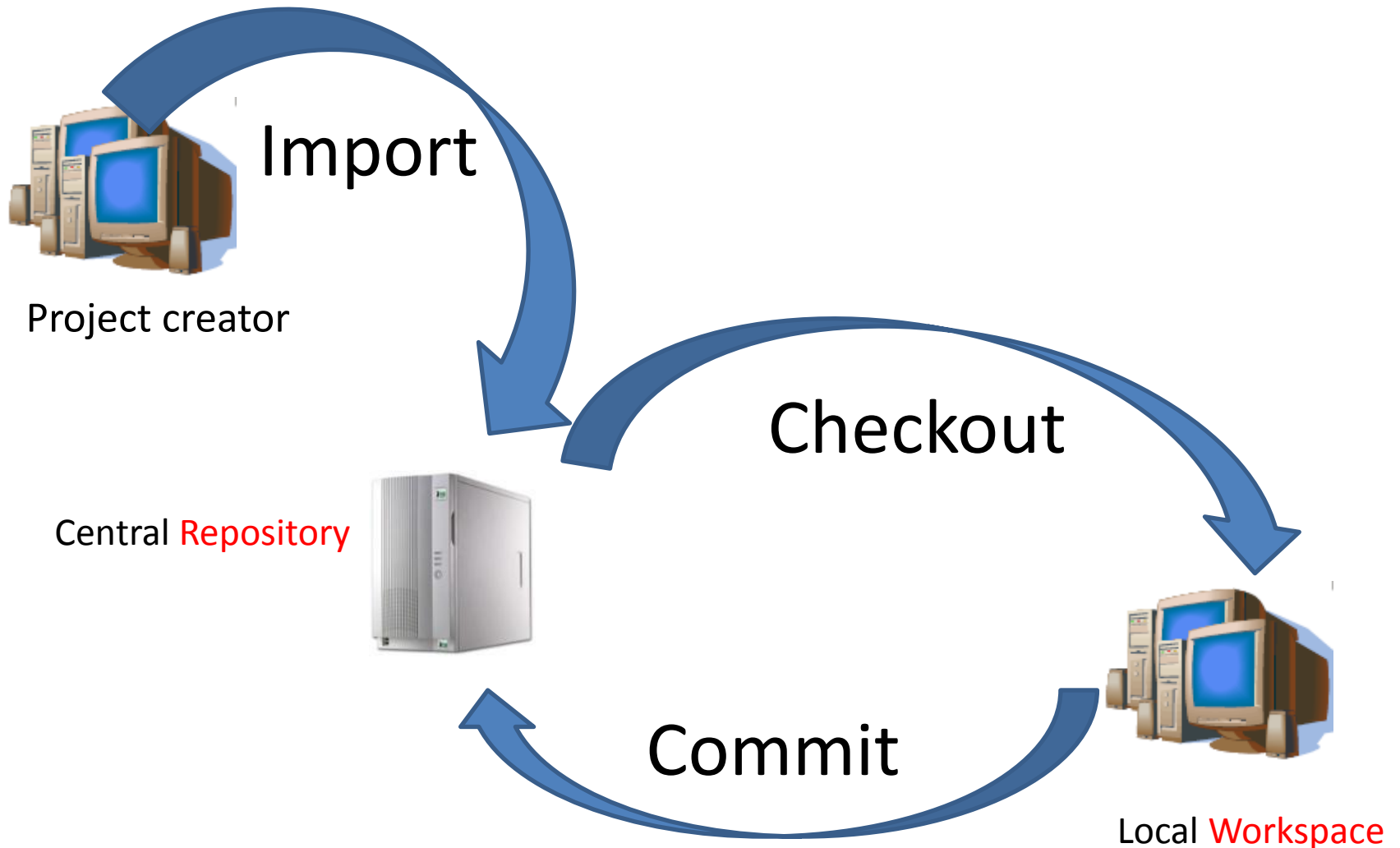
Repository and Workspace



Repository

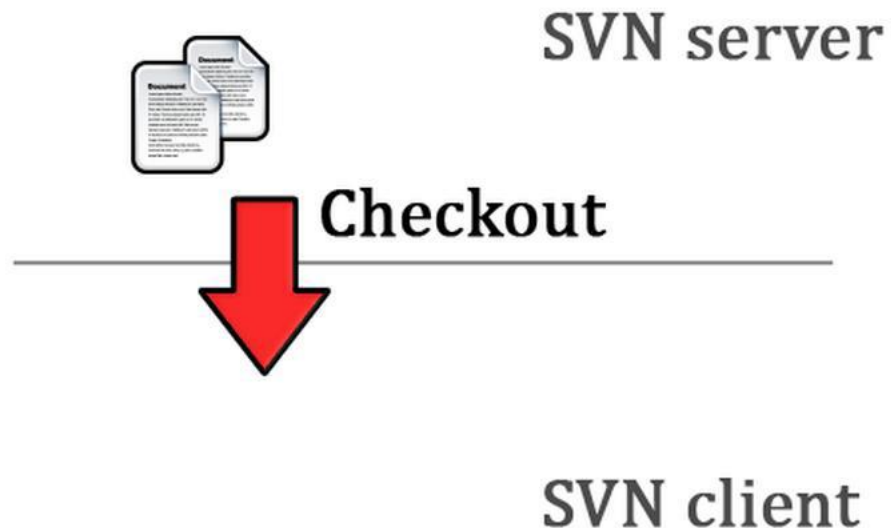
- SVN repository can be built on different protocols (e.g. svn, http, https)
- Each directory or file has a unique URL
 - **E.g.**
https://netdb.cs.nthu.edu.tw/svn/SoftwareStudio_prj/
- Some SVN clients to access the repository:
 - Web browser
 - TortoiseSVN
 - Subclipse

How to Interact with the Repository?



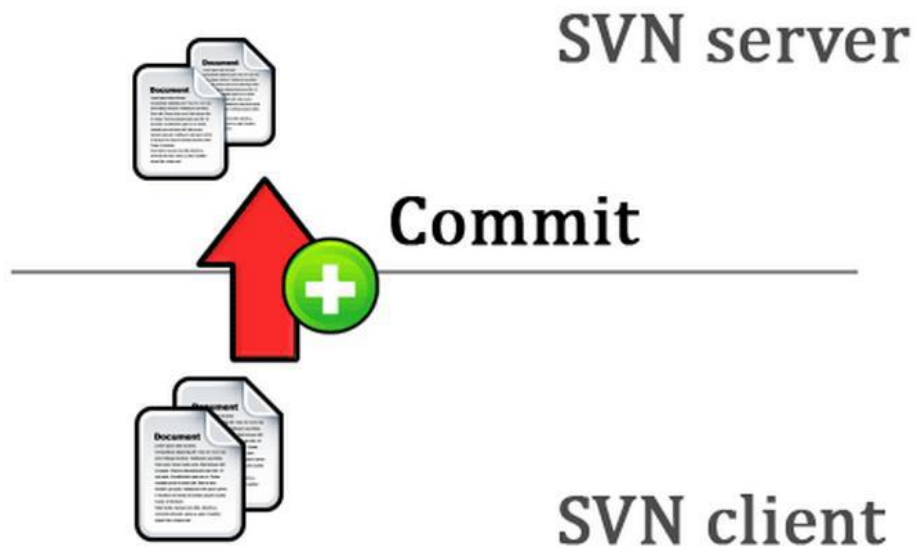
Checkout

- Checkout repository to create a working copy in local workspace
- Checked out files appear to be a directory with program's source



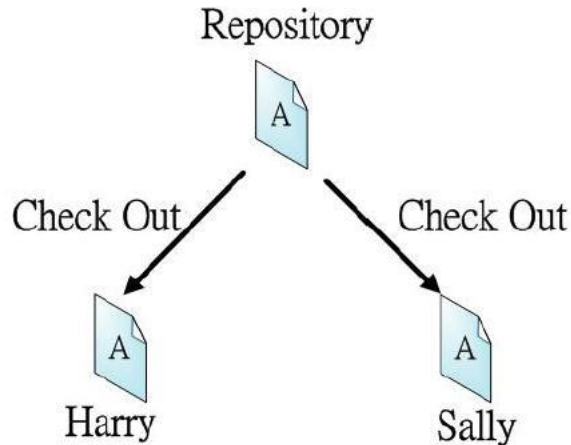
Commit

- Find changes made since checkout and commits them into the repository
- Creates a new revision number in the repository

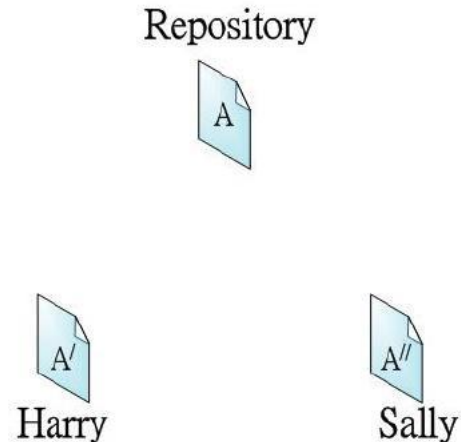


Conflict and Merge

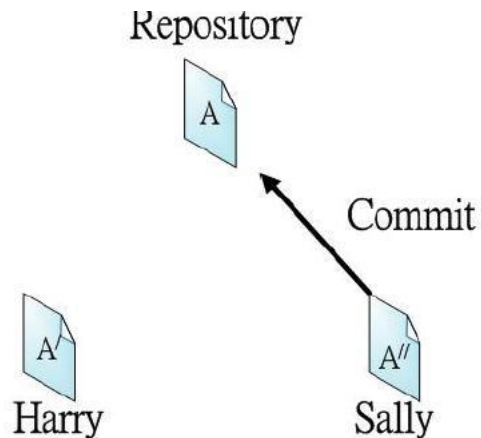
1. Checkout at the same time



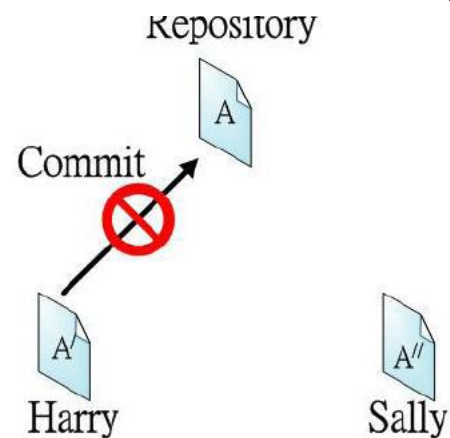
2. Each modifies his/her file



3. Sally commit first

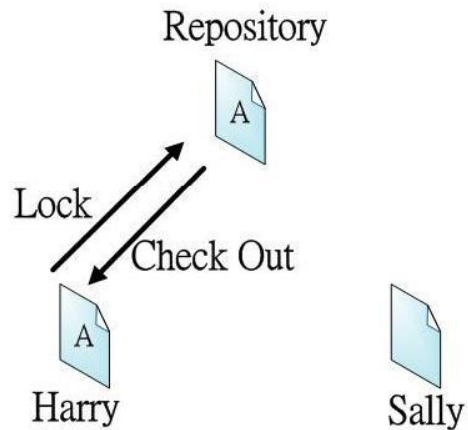


4. Out-Of-Date when Harry commit

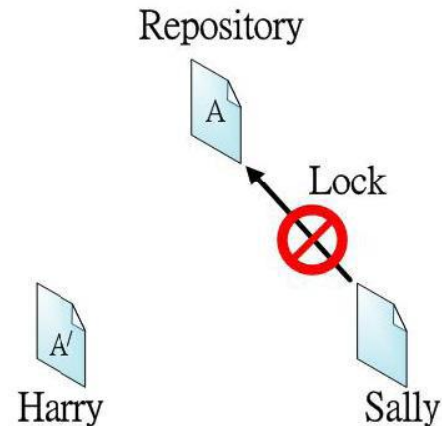


Conflict and Merge (Sol.1)

1. Harry locks the file



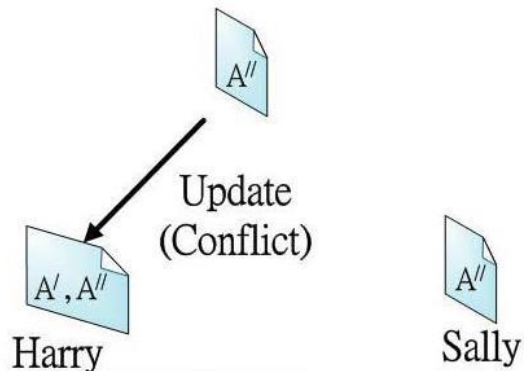
2. Sally can't edit the file until unlocking



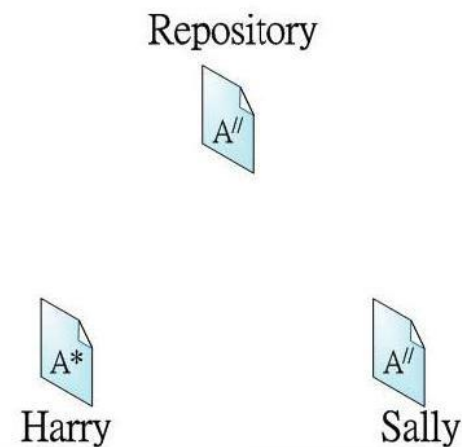
Conflict and Merge (Sol.2)

- This is the way SVN does

1. Harry commits and finds the file conflicting



2. Harry merges the differences between repository-file and his file and then re-commits



Outline

- Why do we need version control?
- Introduction to SVN
- **Using SVN Client in Eclipse**
- Exercise and Homework Submission

Using SVN in Eclipse

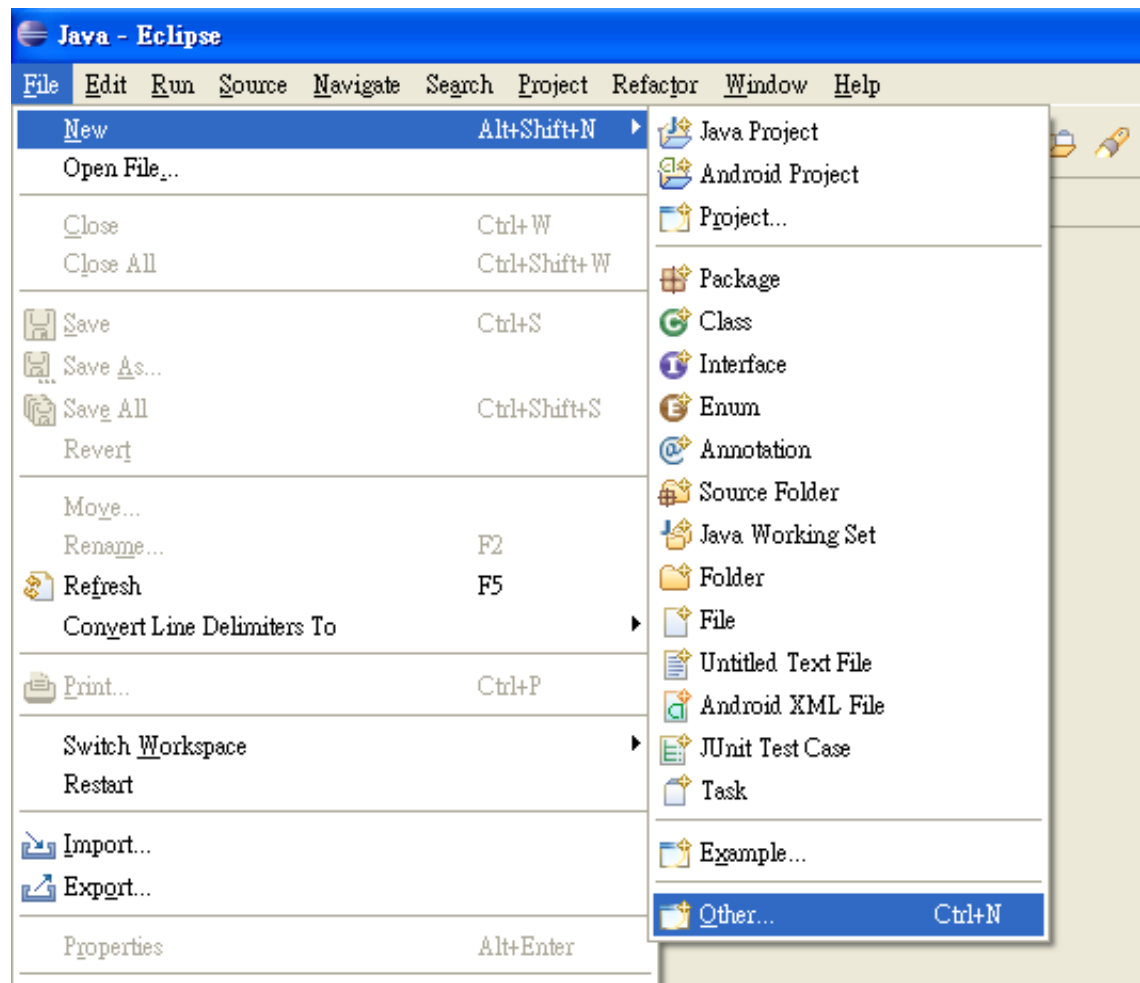
- Installing the Subclipse plugin (see appendix B on the course website)
 - A built-in SVN client in eclipse
- Subclipse operations in eclipse
 - “Checkout” a project
 - “Commit” your code from eclipse
 - “Import” project to repository

Authentication Required

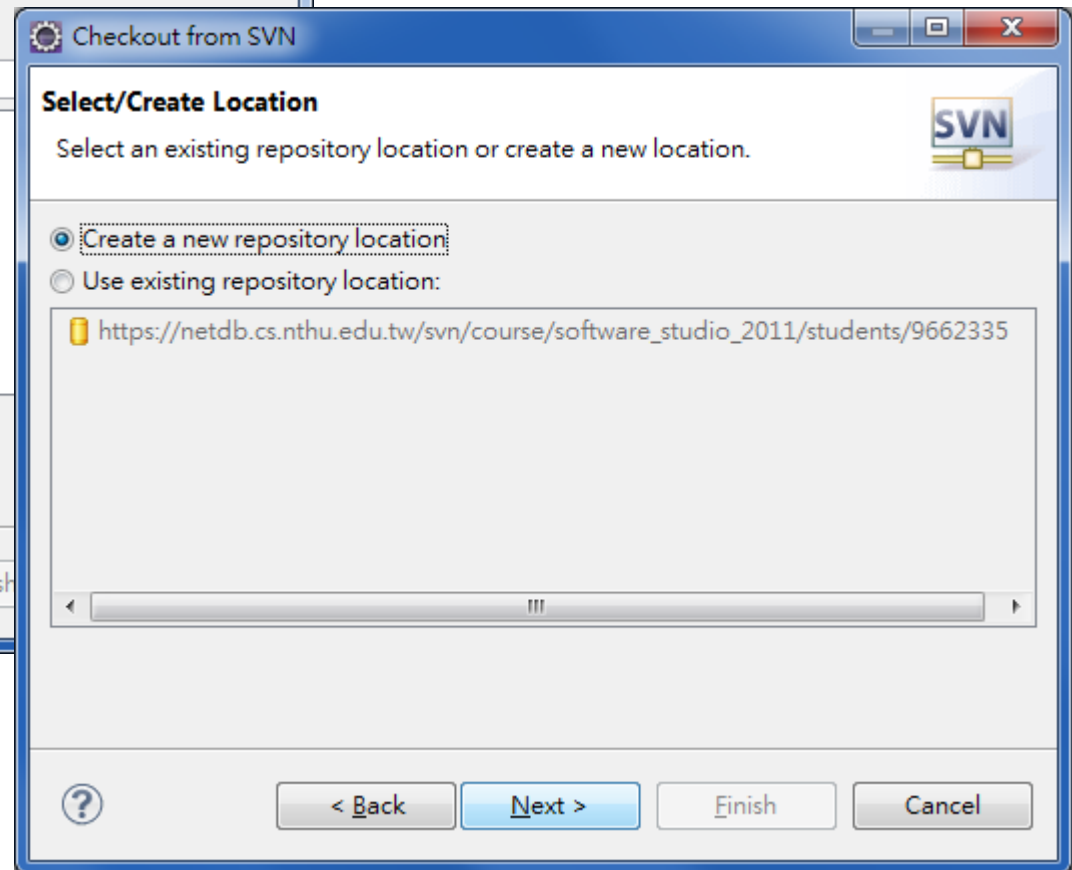
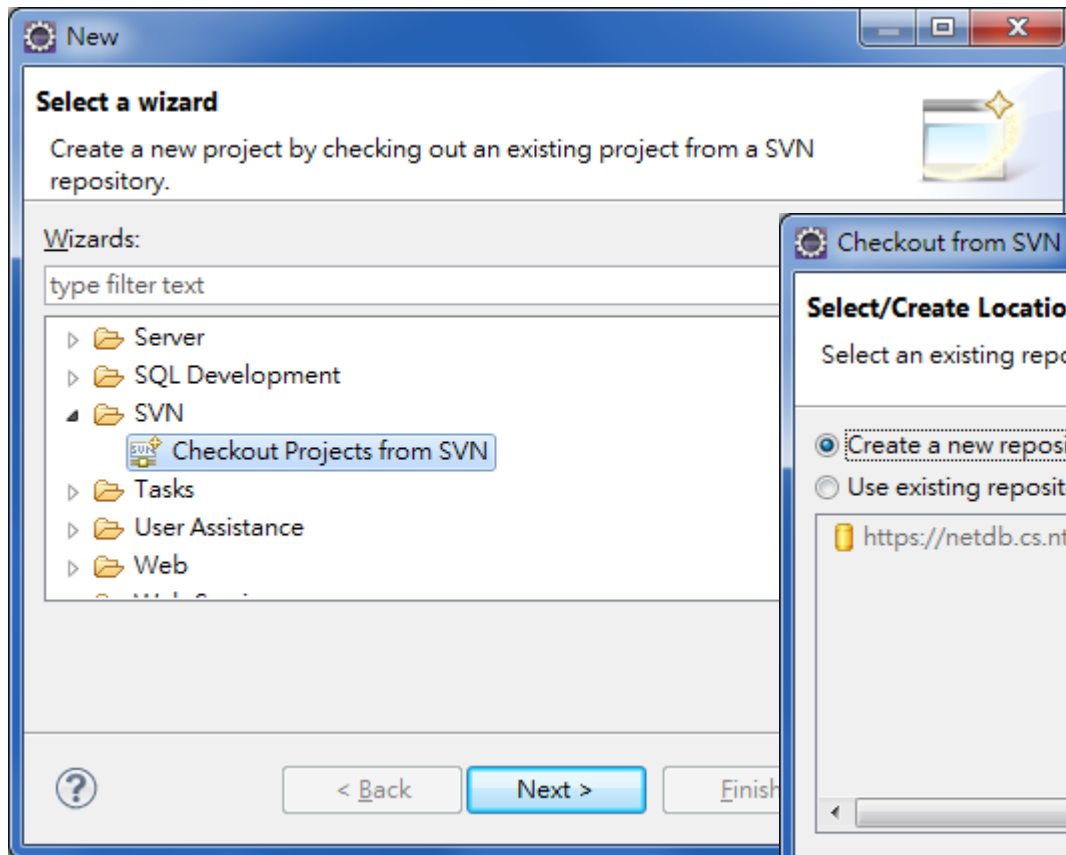
- Account:
 - Your student ID
- Password:
 - Your student ID by default
 - Change it using this page:
<https://netdb.cs.nthu.edu.tw/svntools/passwd/index.php>

Checkout Project from Repository

- File -> New -> Other

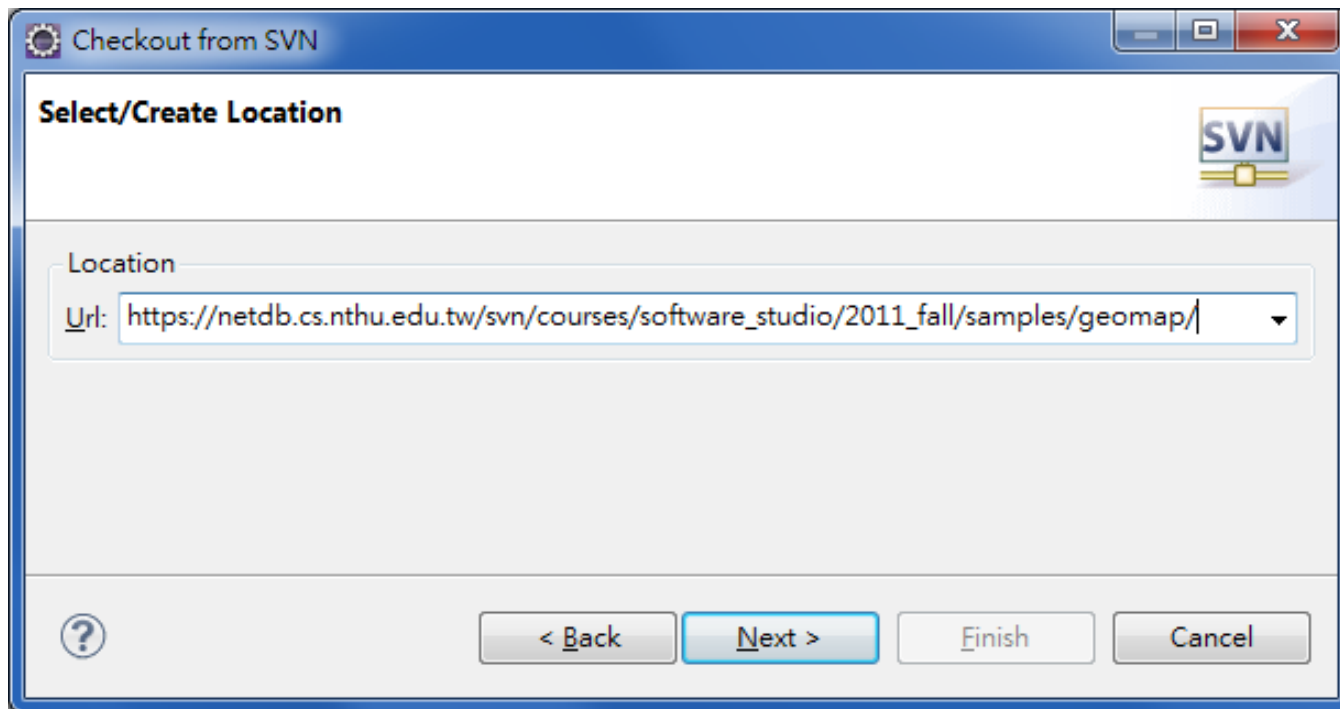


Checkout Project from Repository

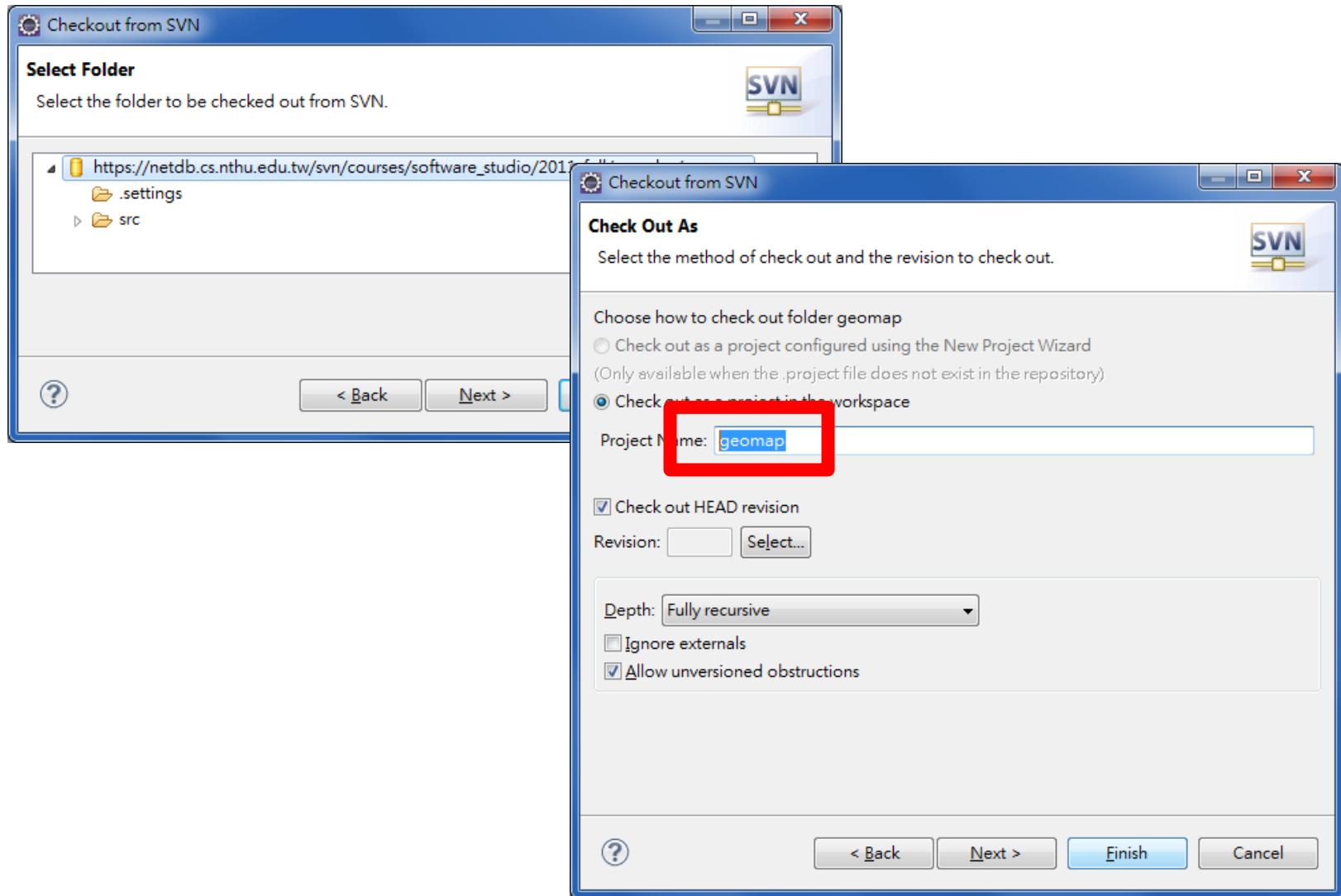


Checkout Project from Repository

- SVN Repository URL for sample projects
https://netdb.cs.nthu.edu.tw/svn/courses/software_studio/2013_fall/samples/

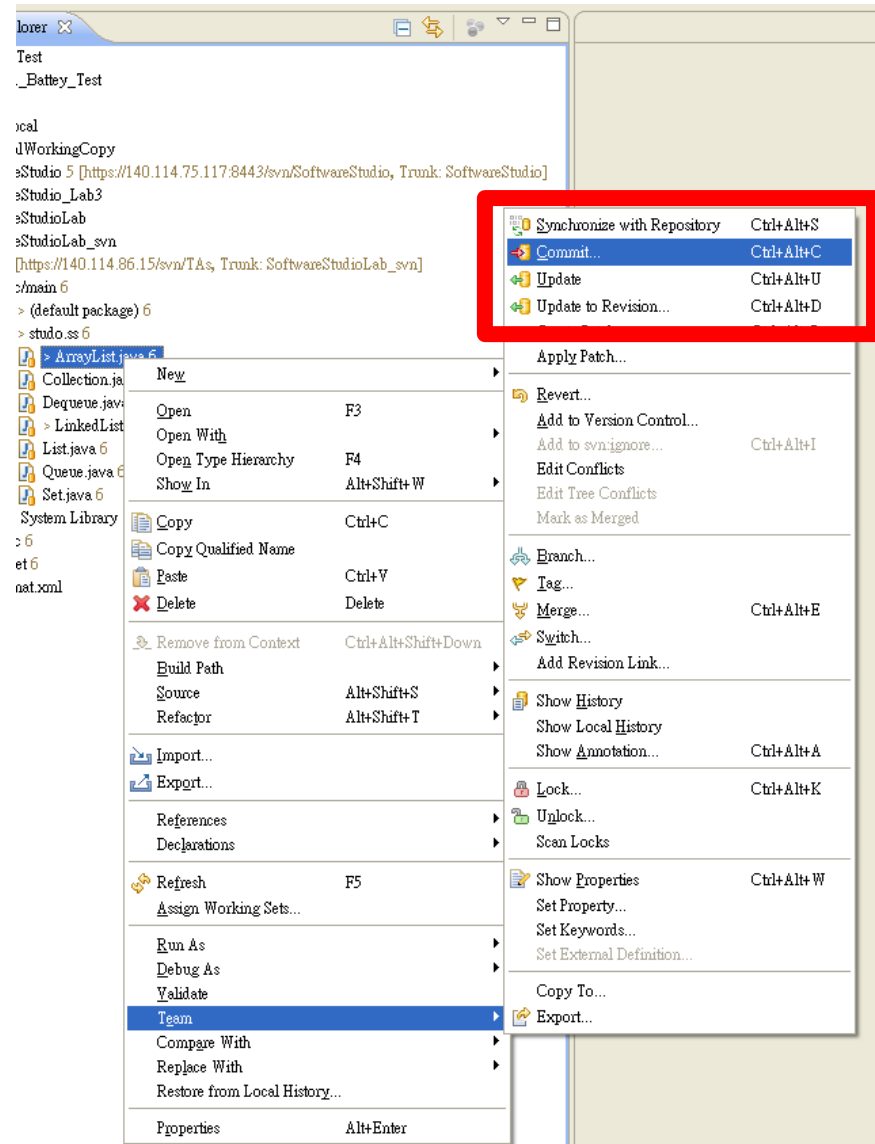


Checkout Project from Repository



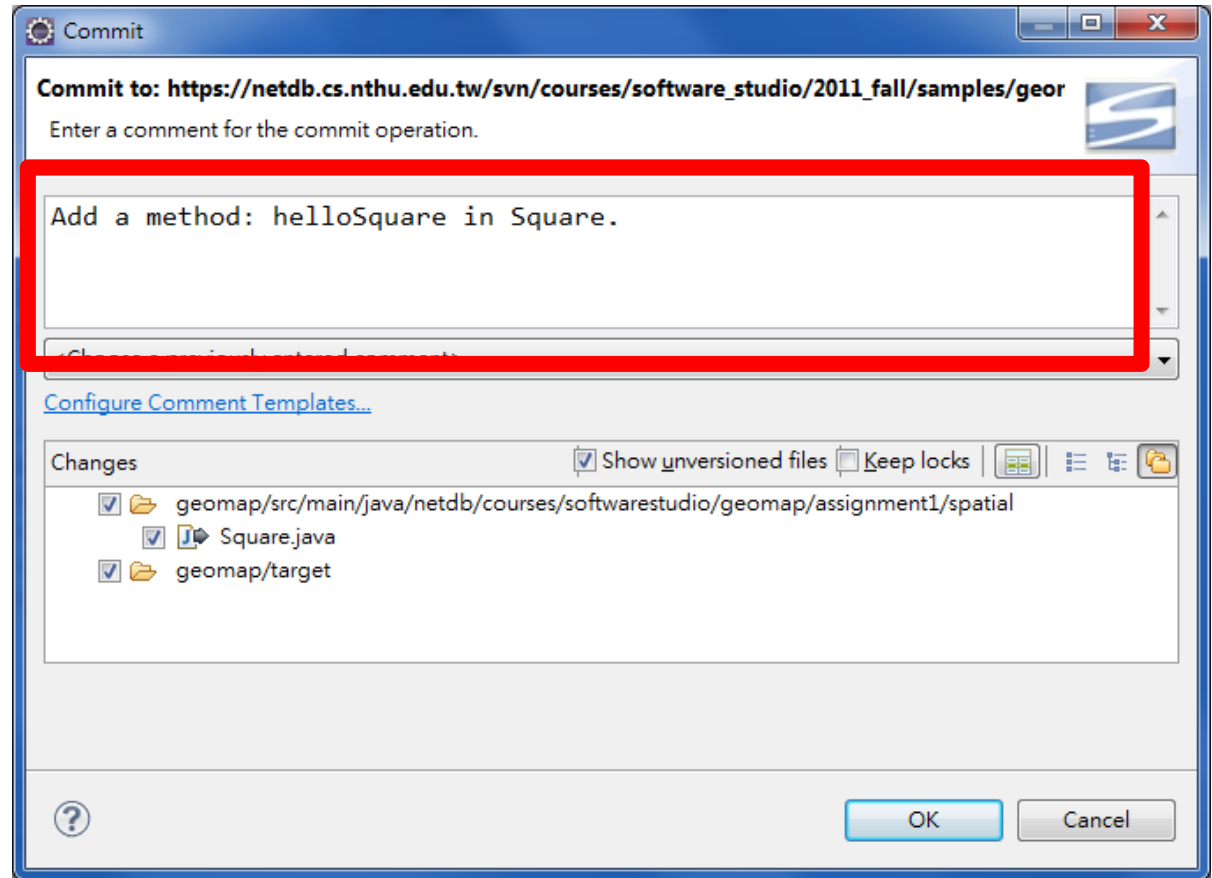
Commit Changes to Repository

- Checked-out projects are connected to SVN
- Changes can be committed back
 - Right click -> Team -> Commit
- You can commit either files or directories



Commit Changes to Repository

- It is a good practice to always comment on the changes you made

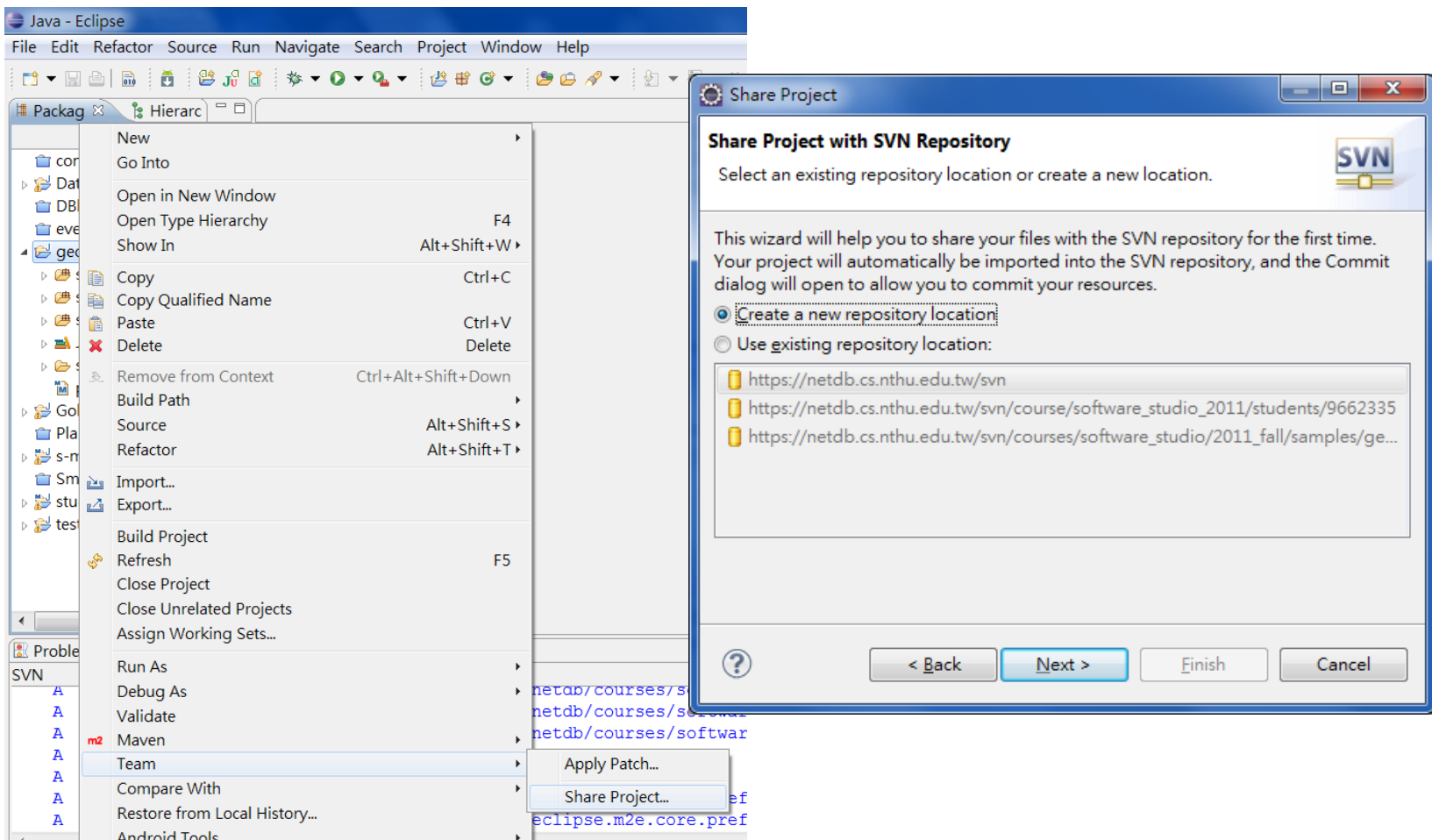


Import Project to Repository

- How to create projects on the SVN server?
 - Create a local project
 - "Import" it into SVN

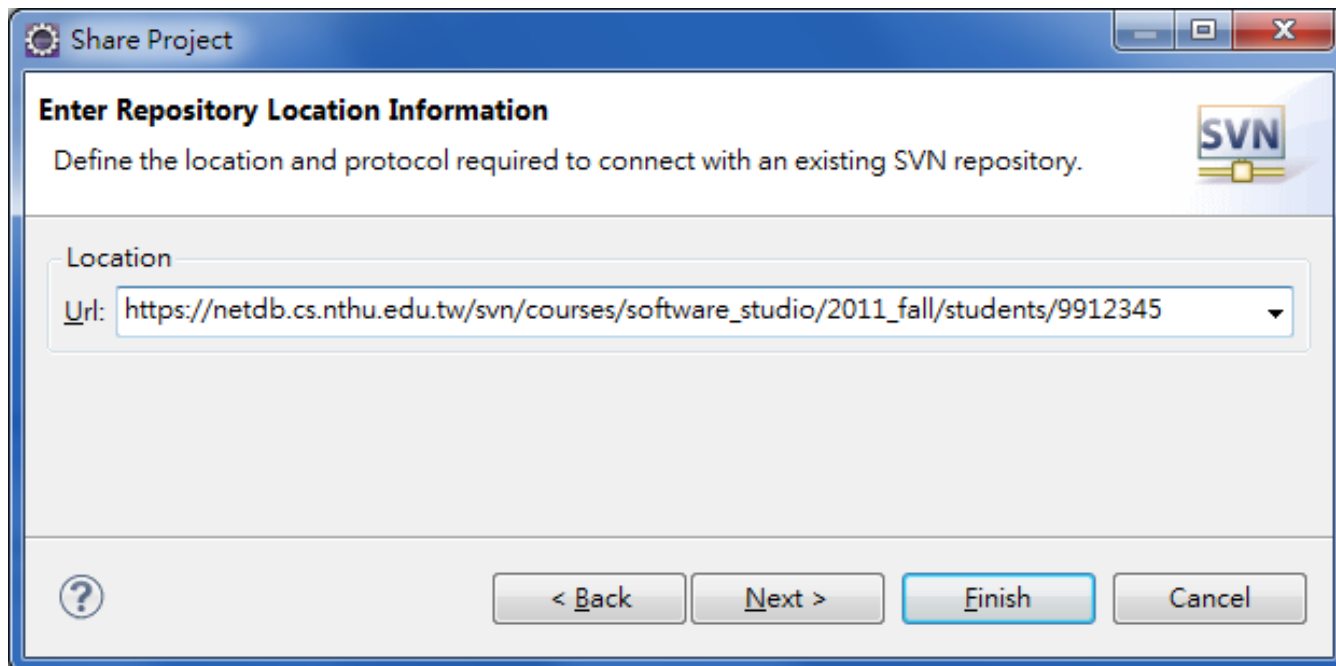
Import Project to Repository

- Right click the project -> Team -> Share Project



Import Project to Repository

- [https://netdb.cs.nthu.edu.tw/svn/courses/software_studio/2013_fall/students/\\${your-id}](https://netdb.cs.nthu.edu.tw/svn/courses/software_studio/2013_fall/students/${your-id})

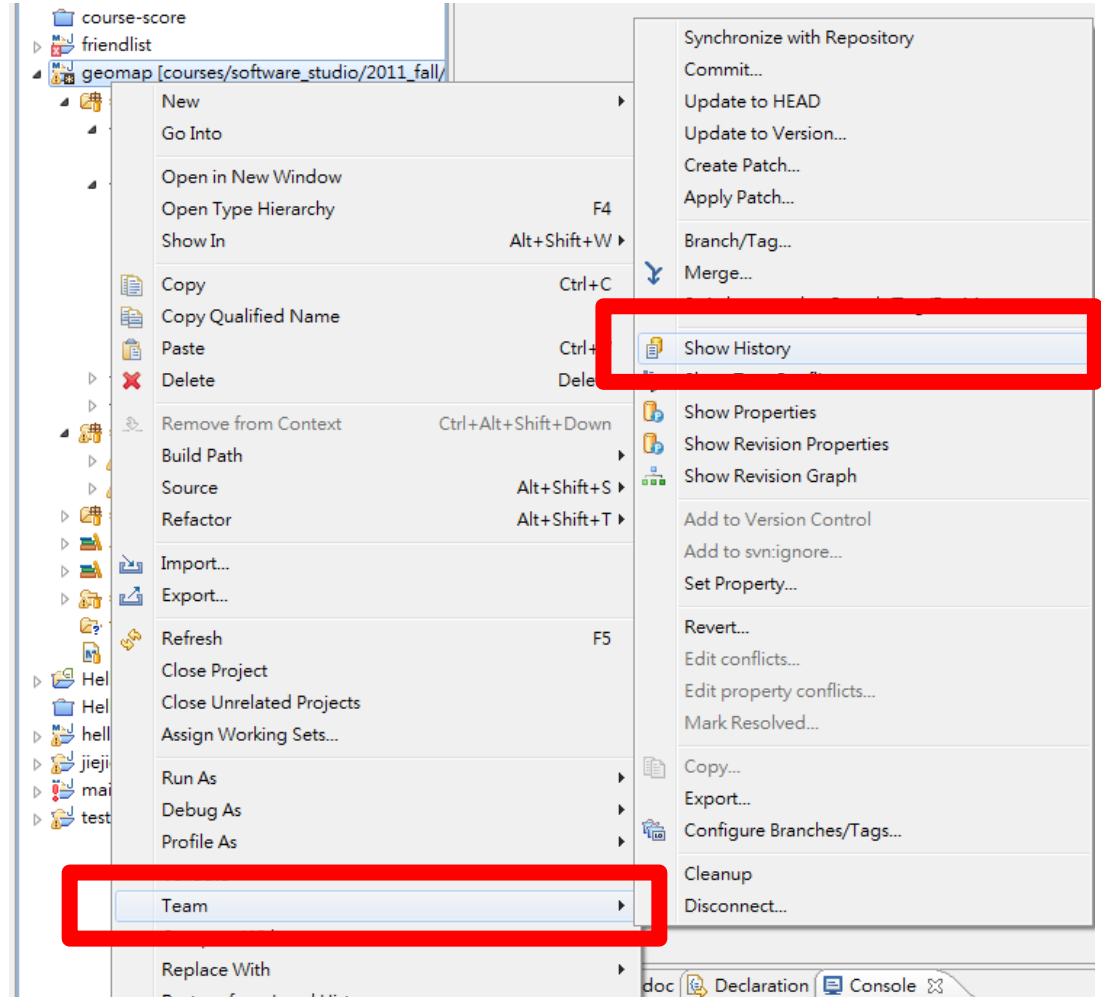


What's Next?

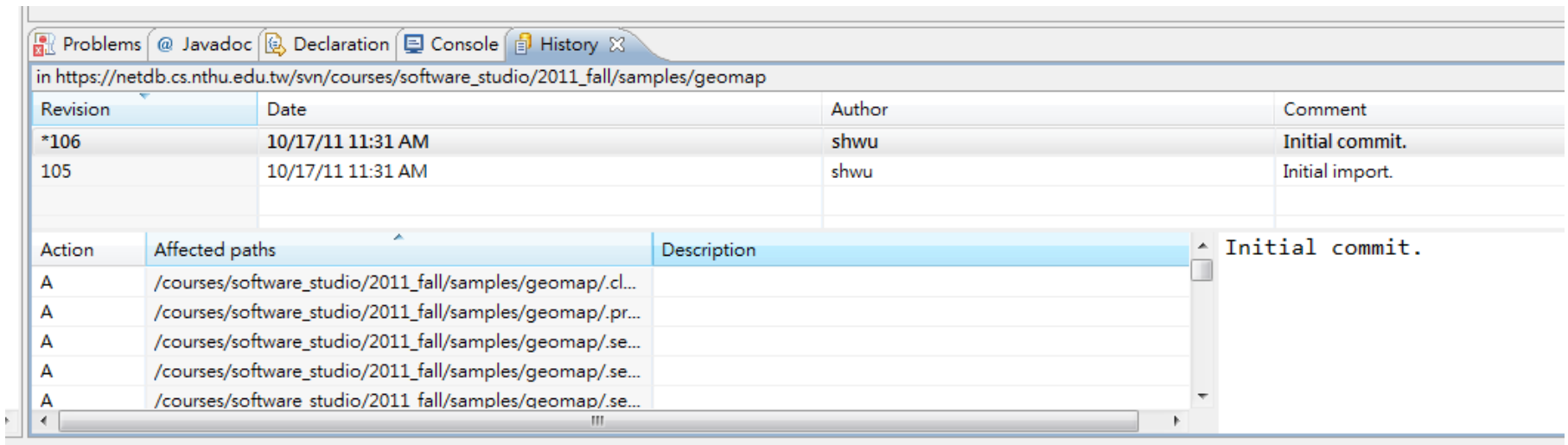
- View the revision history
- Lookup differences while conflicting
- Switch to older versions

Revision History

- Right click -> Team -> Show history



Revision History



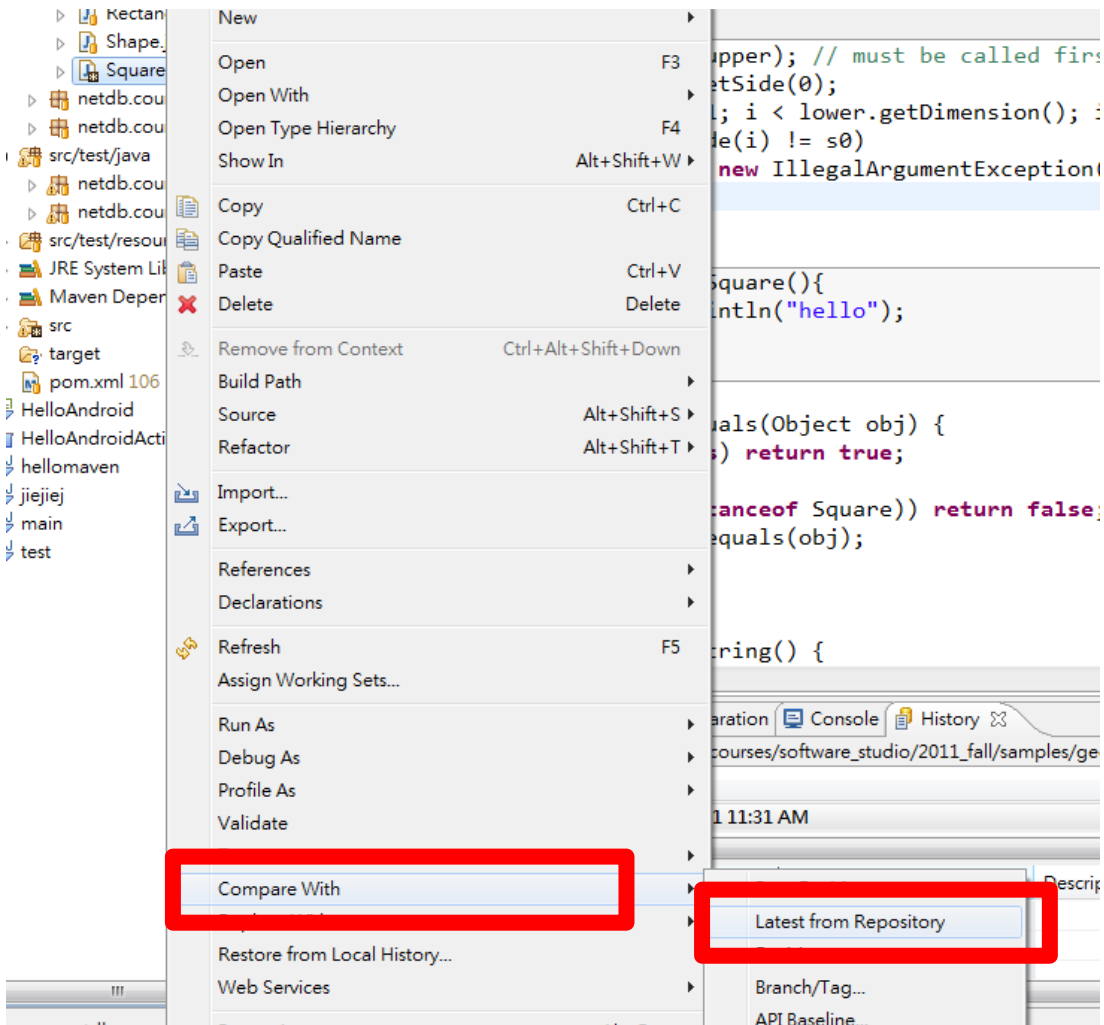
The screenshot shows an IDE window with the 'History' tab selected. The address bar indicates the repository path: `https://netdb.cs.nthu.edu.tw/svn/courses/software_studio/2011_fall/samples/geomap`. The main area displays a table of revisions, with revision 106 highlighted. Below this, a detailed view of the selected revision shows the actions performed and the files affected.

Revision	Date	Author	Comment
*106	10/17/11 11:31 AM	shwu	Initial commit.
105	10/17/11 11:31 AM	shwu	Initial import.

Action	Affected paths	Description
A	/courses/software_studio/2011_fall/samples/geomap/.cl...	
A	/courses/software_studio/2011_fall/samples/geomap/.pr...	
A	/courses/software_studio/2011_fall/samples/geomap/.se...	
A	/courses/software_studio/2011_fall/samples/geomap/.se...	
A	/courses/software_studio/2011 fall/samples/qeomap/.se...	

Initial commit.

Lookup Difference



- Right click -> Compare with -
> ...
- We can lookup the differences between
 1. Local files
 2. Local file and file in repository

Lookup Difference

Local file

File in the repository

Java Source Compare

Workspace file: Square.java

```
super(lower, upper); // must be called first
double s0 = getSide(0);
for (int i = 1; i < lower.getDimension(); i++) {
    if (getSide(i) != s0)
        throw new IllegalArgumentException();
}

public void helloSquare(){
    System.out.println("hello");
}

public boolean equals(Object obj) {
    if(obj == this) return true;

    if (!(obj instanceof Square)) return false;
    return super.equals(obj);
}

@Override
public String toString() {
```

Repository file: Square.java

```
super(lower, upper); // must be called first
double s0 = getSide(0);
for (int i = 1; i < lower.getDimension(); i++) {
    if (getSide(i) != s0)
        throw new IllegalArgumentException();
}

@Override
public boolean equals(Object obj) {
    if(obj == this) return true;

    if (!(obj instanceof Square)) return false;
    return super.equals(obj);
}

@Override
public String toString() {
    StringBuffer ret = new StringBuffer("Square@{");
    ret.append(lower.toString());
    ret.append(", " + upper.toString() + "}");
    return ret.toString();
}
```

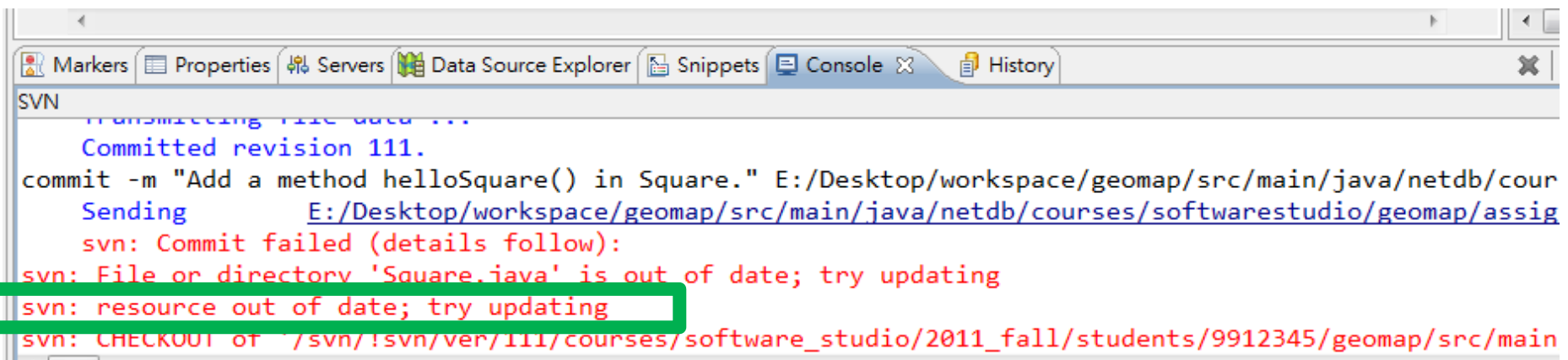
Different part will be covered by a box

Merging Conflicts

- What to do when conflicts occur at commit time?
 - You must resolve conflicts before proceeding
- First, lookup the differences
- Then you have options:
 - Modify your code locally, merge the changes, and commit merged version to SVN
 - Discard your changes by reverting your code to an older version
 - Contact other authors to modify/revert their code

Merging Conflicts

- Conflict when commit



The screenshot shows an IDE's console window with several tabs: Markers, Properties, Servers, Data Source Explorer, Snippets, Console, and History. The Console tab is active, displaying the output of an SVN commit command. The text in the console is as follows:

```
SVN
Transmitting file data ...
Committed revision 111.
commit -m "Add a method helloSquare() in Square." E:/Desktop/workspace/geomap/src/main/java/netdb/cour
Sending      E:/Desktop/workspace/geomap/src/main/java/netdb/courses/softwarestudio/geomap/assig
svn: Commit failed (details follow):
svn: File or directory 'Square.java' is out of date; try updating
svn: resource out of date; try updating
svn: CHECKOUT of '/svn!svn/ver/111/courses/software_studio/2011_fall/students/9912345/geomap/src/main
```

The line "svn: resource out of date; try updating" is highlighted with a green rectangular box.

Merging Conflicts

- Lookup Difference

The screenshot displays an IDE window titled 'Structure Compare' with a sub-tab 'Java Source Compare'. It compares two versions of the `Square` class: 'Workspace file: geomap' on the left and 'Repository file: geomap' on the right. The 'Workspace' version has a `helloSquare()` method that prints 'Hello my square', while the 'Repository' version prints 'Hello 9912345'. The `helloSquare()` method in both versions is highlighted with a blue background, indicating a conflict. The `main` method in both versions is identical, printing 'Hello my square'.

```
Workspace file: geomap
public Square(Point lower, Point upper) {
    super(lower, upper); // must be called first
    double s0 = getSide(0);
    for (int i = 1; i < lower.getDimension(); i++) {
        if (getSide(i) != s0)
            throw new IllegalArgumentException();
    }
}

public void helloSquare(){
    System.out.println("Hello my square");
}

@Override
public boolean equals(Object obj) {
    if(obj == this) return true;

    if (!(obj instanceof Square)) return false;
    return super.equals(obj);
}

@Override
public String toString() {
    StringBuffer ret = new StringBuffer("Square@{");
    ret.append(lower.toString());
    ret.append(", " + upper.toString() + "}");
    return ret.toString();
}

Repository file: geomap
public Square(Point lower, Point upper) {
    super(lower, upper); // must be called first
    double s0 = getSide(0);
    for (int i = 1; i < lower.getDimension(); i++) {
        if (getSide(i) != s0)
            throw new IllegalArgumentException();
    }
}

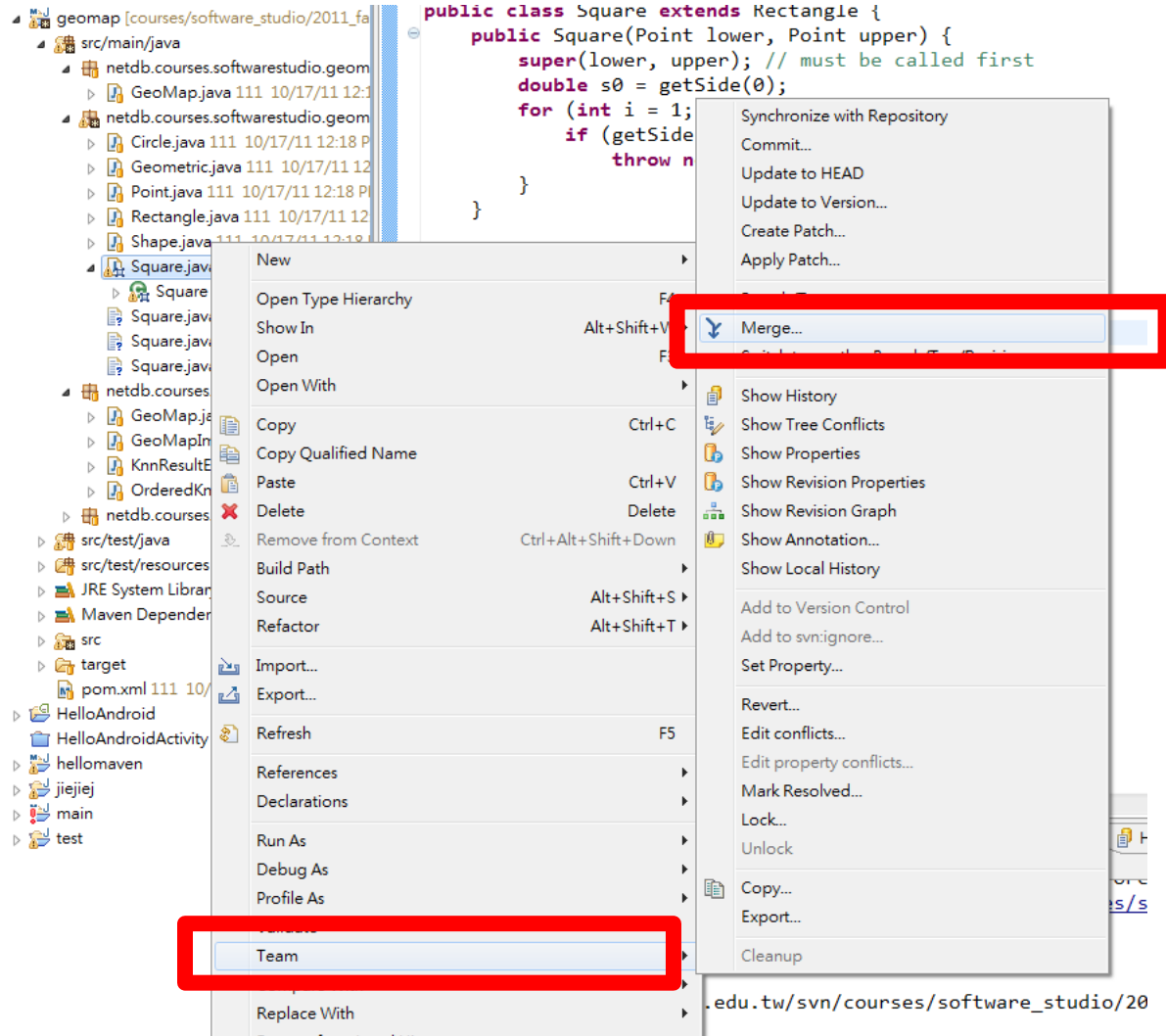
public void helloSquare(){
    System.out.println("Hello 9912345");
}

@Override
public boolean equals(Object obj) {
    if(obj == this) return true;

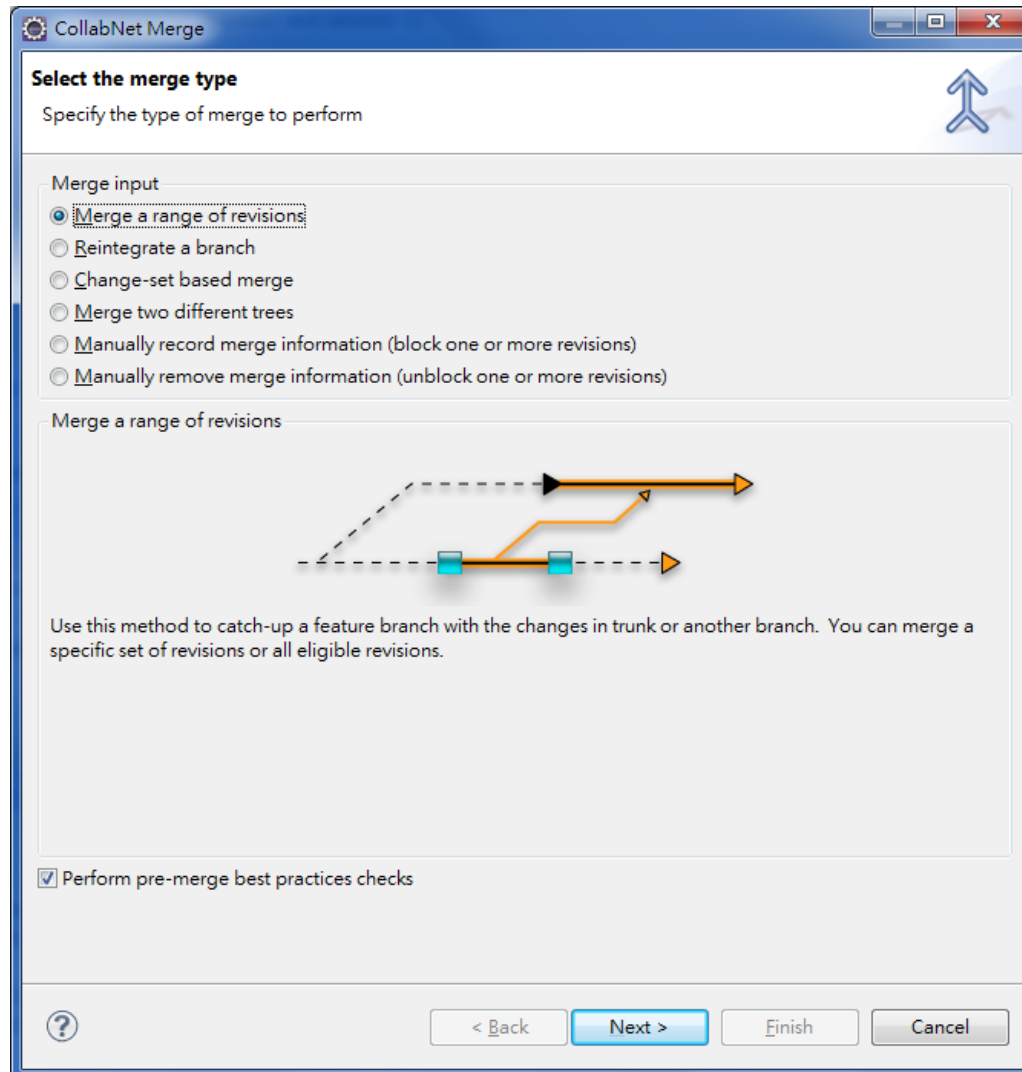
    if (!(obj instanceof Square)) return false;
    return super.equals(obj);
}

@Override
public String toString() {
    StringBuffer ret = new StringBuffer("Square@{");
    ret.append(lower.toString());
    ret.append(", " + upper.toString() + "}");
    return ret.toString();
}
```

Merging Conflicts

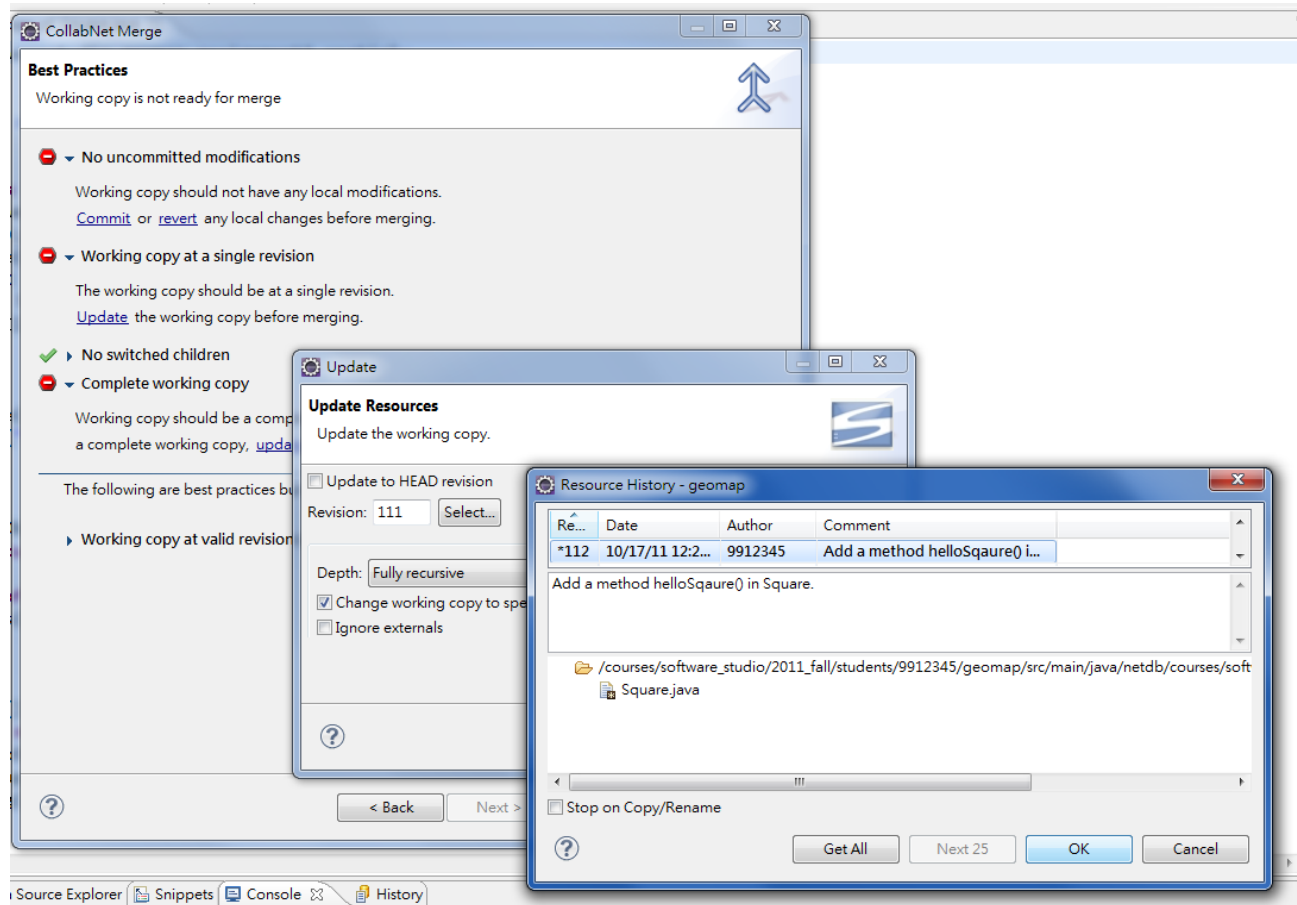


Merging Conflicts



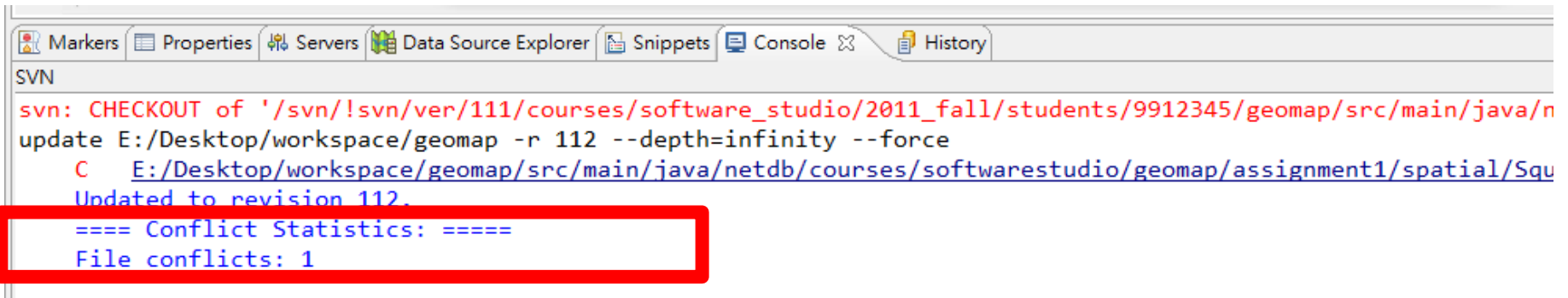
Merging Conflicts

- There are options to process the merge.



Merging Conflicts

- The console will show the conflicts

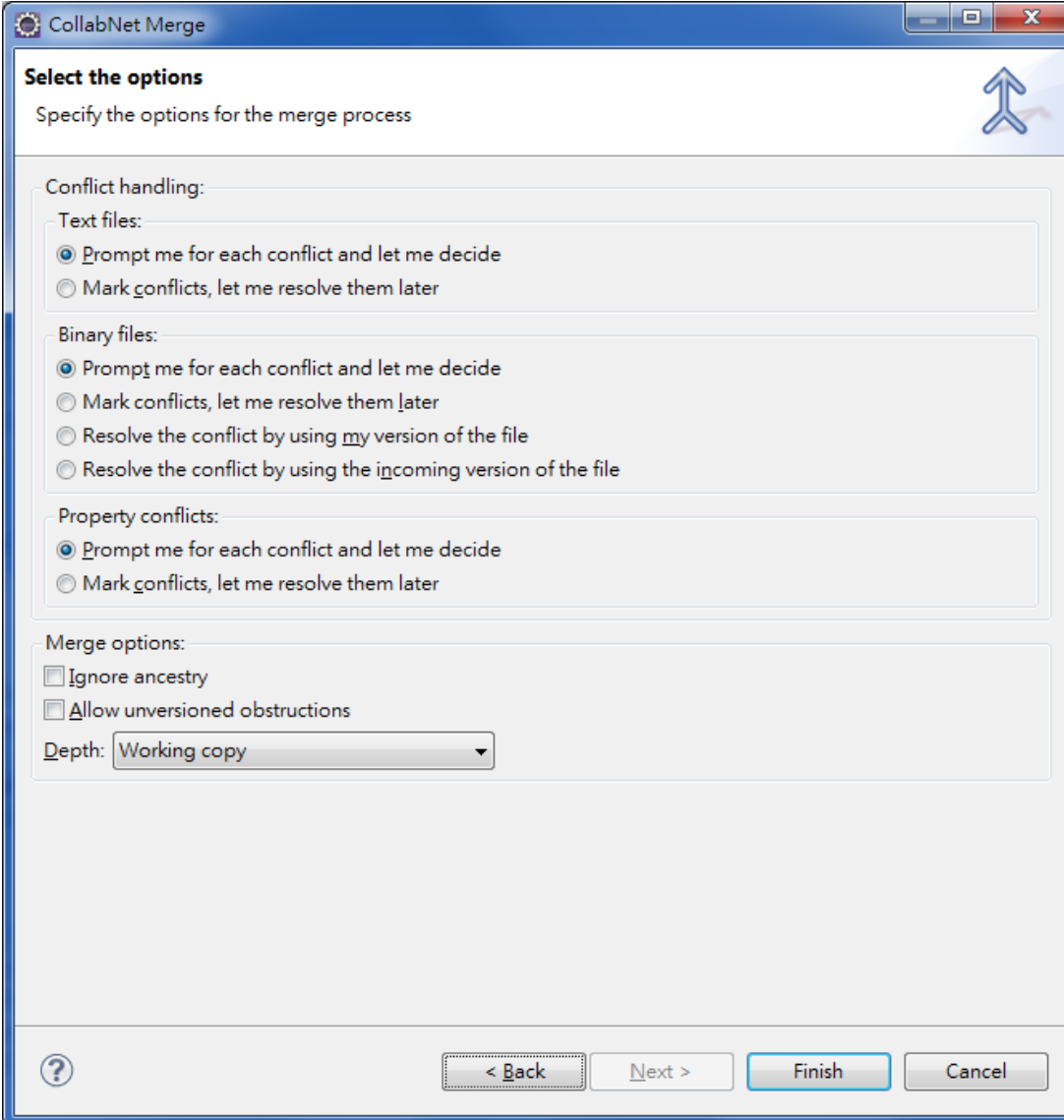


The screenshot shows an IDE's console window with the following text:

```
SVN
svn: CHECKOUT of '/svn/!svn/ver/111/courses/software_studio/2011_fall/students/9912345/geomap/src/main/java/n
update E:/Desktop/workspace/geomap -r 112 --depth=infinity --force
C E:/Desktop/workspace/geomap/src/main/java/netdb/courses/softwarestudio/geomap/assignment1/spatial/Squ
Updated to revision 112.
==== Conflict Statistics: ====
File conflicts: 1
```

The last two lines, "==== Conflict Statistics: =====" and "File conflicts: 1", are enclosed in a red rectangular box.

Merging Conflicts



The image shows a screenshot of the 'CollabNet Merge' dialog box. The window has a blue title bar with the text 'CollabNet Merge' and standard window controls. The main area is titled 'Select the options' with a subtitle 'Specify the options for the merge process'. There is a blue icon of two arrows pointing up in the top right corner. The dialog is divided into several sections: 'Conflict handling:' with sub-sections for 'Text files:', 'Binary files:', and 'Property conflicts:'. Each sub-section contains two radio button options. The 'Merge options:' section at the bottom contains two checkbox options and a 'Depth:' dropdown menu. At the bottom of the dialog are four buttons: a help button (question mark), '< Back', 'Next >', and 'Finish'. The 'Finish' button is highlighted in blue.

CollabNet Merge

Select the options
Specify the options for the merge process

Conflict handling:

Text files:

- ☒ Prompt me for each conflict and let me decide
- ☐ Mark conflicts, let me resolve them later

Binary files:

- ☒ Prompt me for each conflict and let me decide
- ☐ Mark conflicts, let me resolve them later
- ☐ Resolve the conflict by using my version of the file
- ☐ Resolve the conflict by using the incoming version of the file

Property conflicts:

- ☒ Prompt me for each conflict and let me decide
- ☐ Mark conflicts, let me resolve them later

Merge options:

- ☐ Ignore ancestry
- ☐ Allow unversioned obstructions

Depth: Working copy

? < Back Next > Finish Cancel

Merging Conflicts

```
GeoMap.java *Square.java copy_of_Square.java

/* A generalized square in a geometric space.
 *
 */
public class Square extends Rectangle {
    public Square(Point lower, Point upper) {
        super(lower, upper); // must be called first
        double s0 = getSide(0);
        for (int i = 1; i < lower.getDimension(); i++) {
            if (getSide(i) != s0)
                throw new IllegalArgumentException();
        }
    }
}

<<<<<<< .mine
    public void helloSquare(){
        System.out.println("Hello square");
    }

=====
    public void helloSquare(){
        System.out.println("Hello 9912345");
    }

>>>>>>> .r112
```


Merging Conflicts

- Commit it after the conflicts are resolved

```
/* A generalized square in a geometric space.
 *
 */
public class Square extends Rectangle {
    public Square(Point lower, Point upper) {
        super(lower, upper); // must be called first
        double s0 = getSide(0);
        for (int i = 1; i < lower.getDimension(); i++) {
            if (getSide(i) != s0)
                throw new IllegalArgumentException();
        }
    }

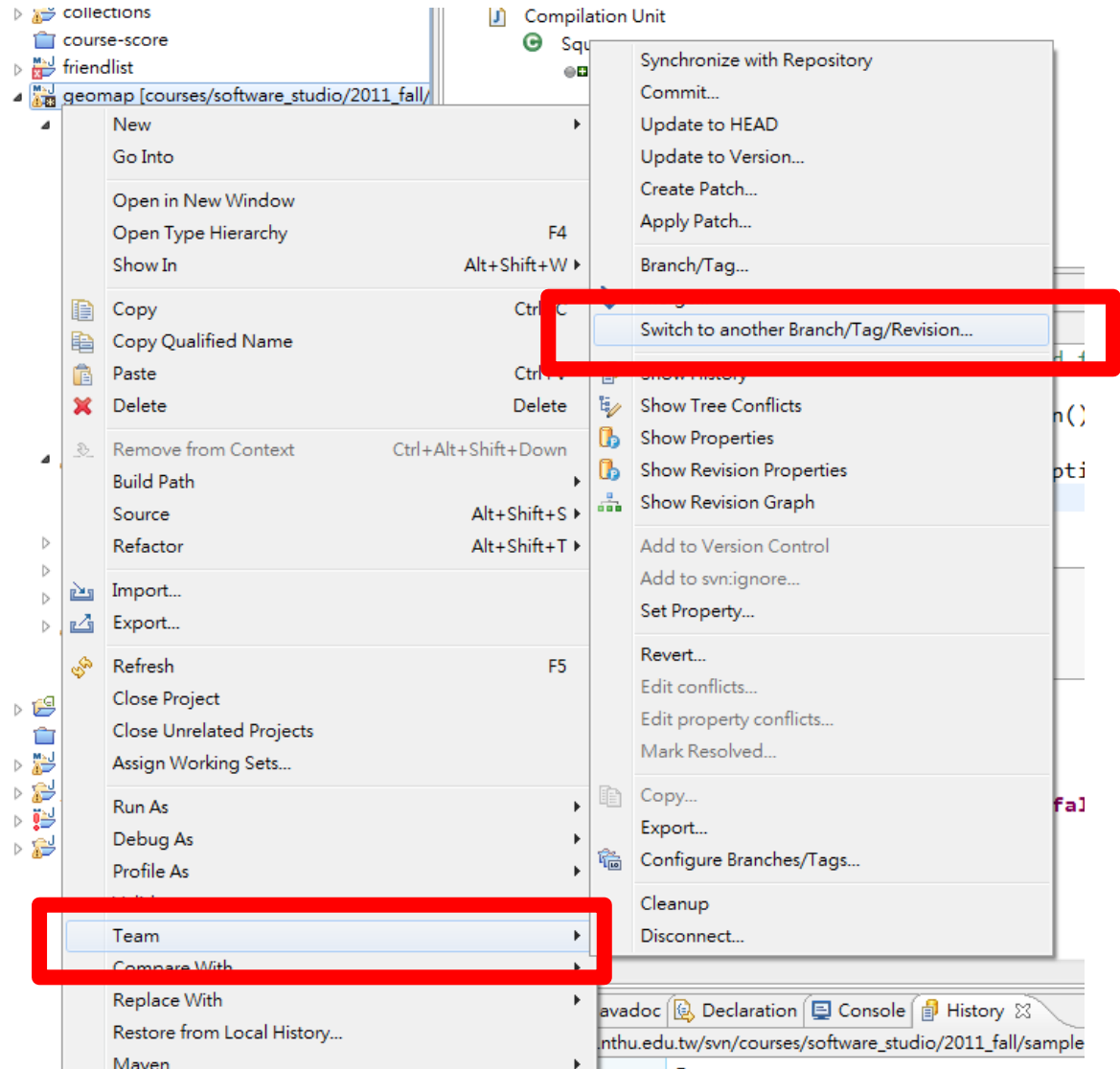
    public void helloSquare(){
        System.out.println("Hello 9912345's square");
    }

    @Override
    public boolean equals(Object obj) {
        if(obj == this) return true;

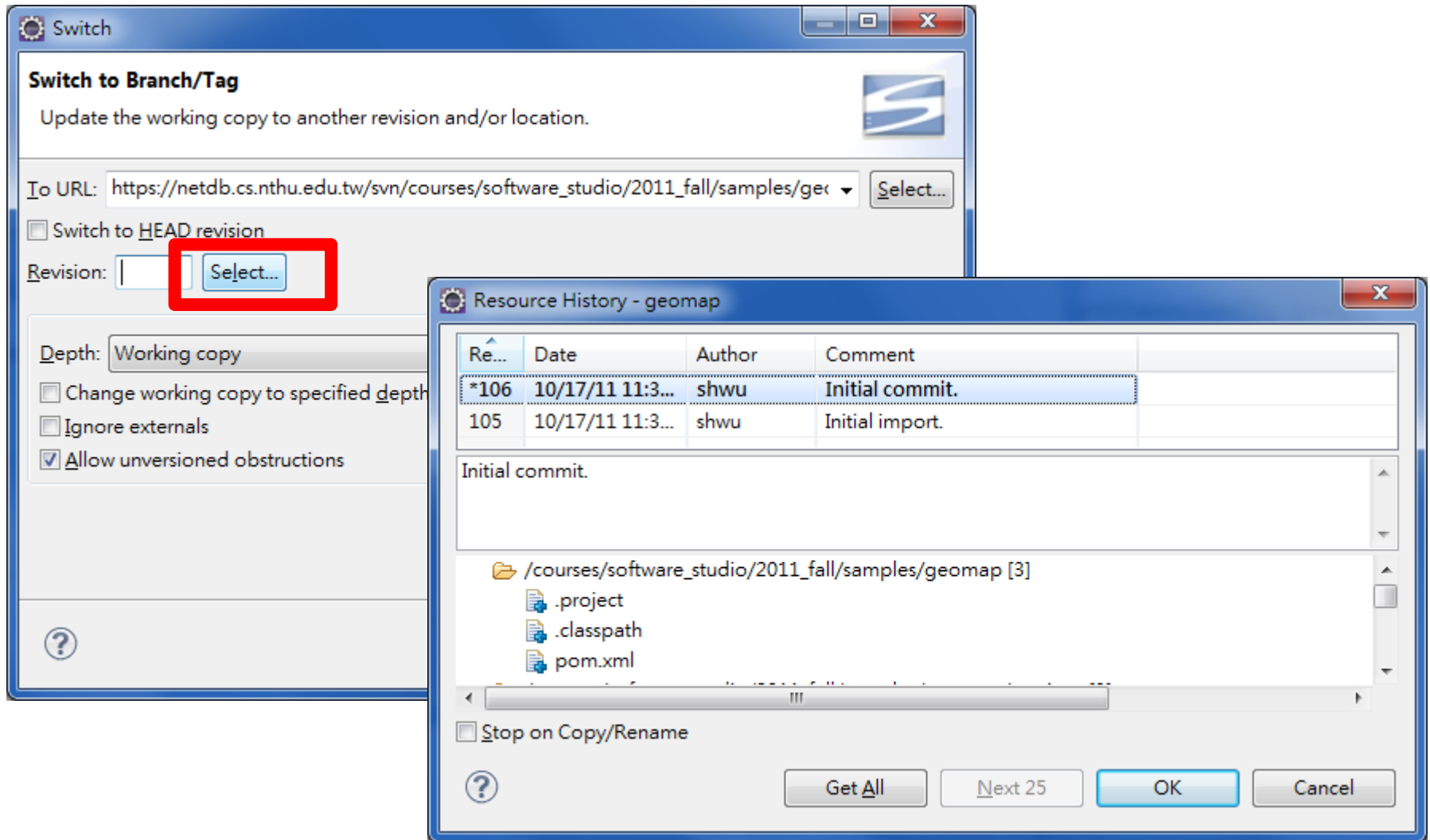
        if (!(obj instanceof Square)) return false;
        return super.equals(obj);
    }
}
```

Switch to Older Versions

- Right click ->
Team -> Switch



Switch to Older Versions



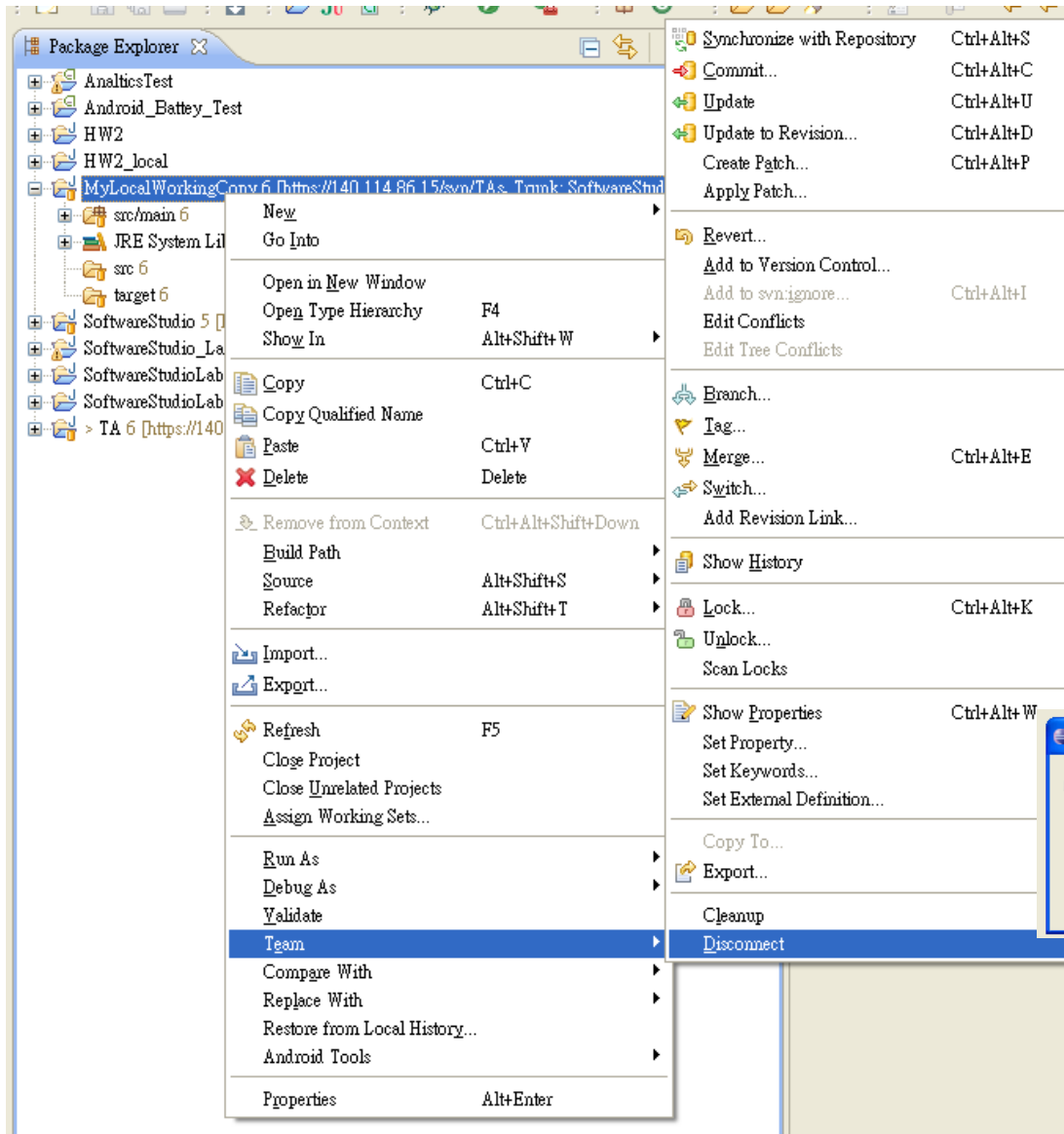
Outline

- Why do we need version control?
- Introduction to SVN
- Using SVN Client in Eclipse
- **Exercise and Homework Submission**

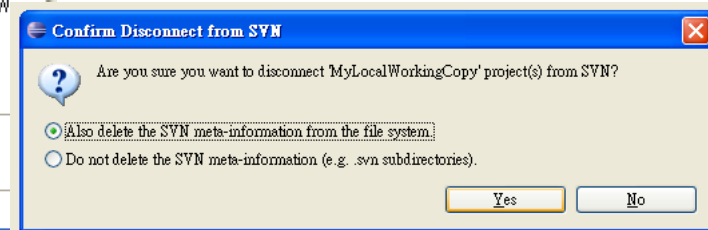
How to Submit Your Solutions to Assignments?

- Before the deadline, import your project into [https://netdb.cs.nthu.edu.tw/svn/courses/software_studio/2013_fall/students/\\${your-student-id}/\\${proj-name}](https://netdb.cs.nthu.edu.tw/svn/courses/software_studio/2013_fall/students/${your-student-id}/${proj-name})
 - E.g.,
https://netdb.cs.nthu.edu.tw/svn/courses/software_studio/2013_fall/students/9912345/geomap
- **Don't** modify the sample projects checked out from https://netdb.cs.nthu.edu.tw/svn/courses/software_studio/2013_fall/samples directly!
 - They are connected to sample projects on SVN
 - You need to break the connection first

Breaking Connections



- After breaking the connection, you have a "local" project
- Then modify this project, and import it to SVN as your solution



Authentication Required

- Account:
 - Your student ID
- Password:
 - Your student ID by default
 - Change it using this page:
<https://netdb.cs.nthu.edu.tw/svntools/passwd/index.php>

A Note on Common Directory Structure

- In practice, many (open source) projects consist of the following 3 subdirectories:
 - trunk
 - Main line of development
 - Changing frequently
 - branches
 - Bug fixing
 - New features
 - Preparation of release
 - Should be merged back into trunk later
 - tags
 - Releases
 - Milestones of development
 - Making searches of older versions easier
- For simplicity, the projects we develop in this course contains only what resides in trunk

Lab: Java Debugging in Eclipse

NetDB

CS, NTHU,

Fall, 2013

Outline

- Overview
- Start a Debug Session
 - How To Start
 - Setting Breakpoints
 - Starting the Debugging
- Stack Frame
- Variables(inspect、 watch)
- Exercise

Outline

- Overview
- Start a Debug Session
 - How To Start
 - Setting Breakpoints
 - Starting the Debugging
- Stack Frame
- Variables(inspect、 watch)
- Exercise

Overview

- How to use Eclipse Java debugger to look inside the Java running program?
- What is a debugger?
 - A tool which lets us pause a running program at any point to see the **contents of variables**.
- Debugger helps you...
 - Fix program mistakes(**semantic errors**).
 - Learn more about the inner working of Java.

Outline

- Overview
- Start a Debug Session
 - How To Start
 - Setting Breakpoints
 - Starting the Debugging
- Stack Frame
- Variables(inspect、 watch)
- Exercise

How To Start

- You need to follow two steps to start:
 1. Setting the **breakpoints**, which tell the debugger where to pause.
 2. Run the program in **debug mode**.

Outline

- Overview
- Start a Debug Session
 - How To Start
 - Setting Breakpoints
 - Starting the Debugging
- Stack Frame
- Variables(inspect、 watch)
- Exercise

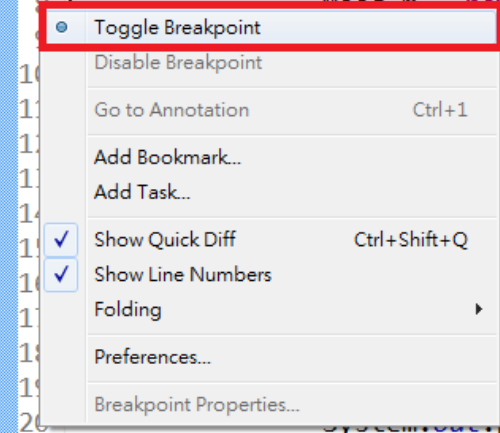
Setting Breakpoints

- Right click in the small left column in your source code editor and select toggle breakpoint. Or you can double click on this place.

```
1 package netdb;
2
3 public class App {
4
5     public static void main(String... args) {
6         try {
7             int g = 3;
8             Mood m = new Mood(Mood.GOOD);
9             Student s1 = new Student(g, m);
10            System.out.println(s1.sayHello());
11        }
12    }
13 }
```

Double click

```
1 package netdb;
2
3 public class App {
4
5     public static void main(String... args) {
6         try {
7             int g = 3;
8             Mood m = new Mood(Mood.GOOD);
9             Student s1 = new Student(g, m);
10            System.out.println(s1.sayHello());
11        }
12    }
13 }
```



Toggle Breakpoint

Disable Breakpoint

Go to Annotation Ctrl+1

Add Bookmark...

Add Task...

Show Quick Diff Ctrl+Shift+Q

Show Line Numbers

Folding

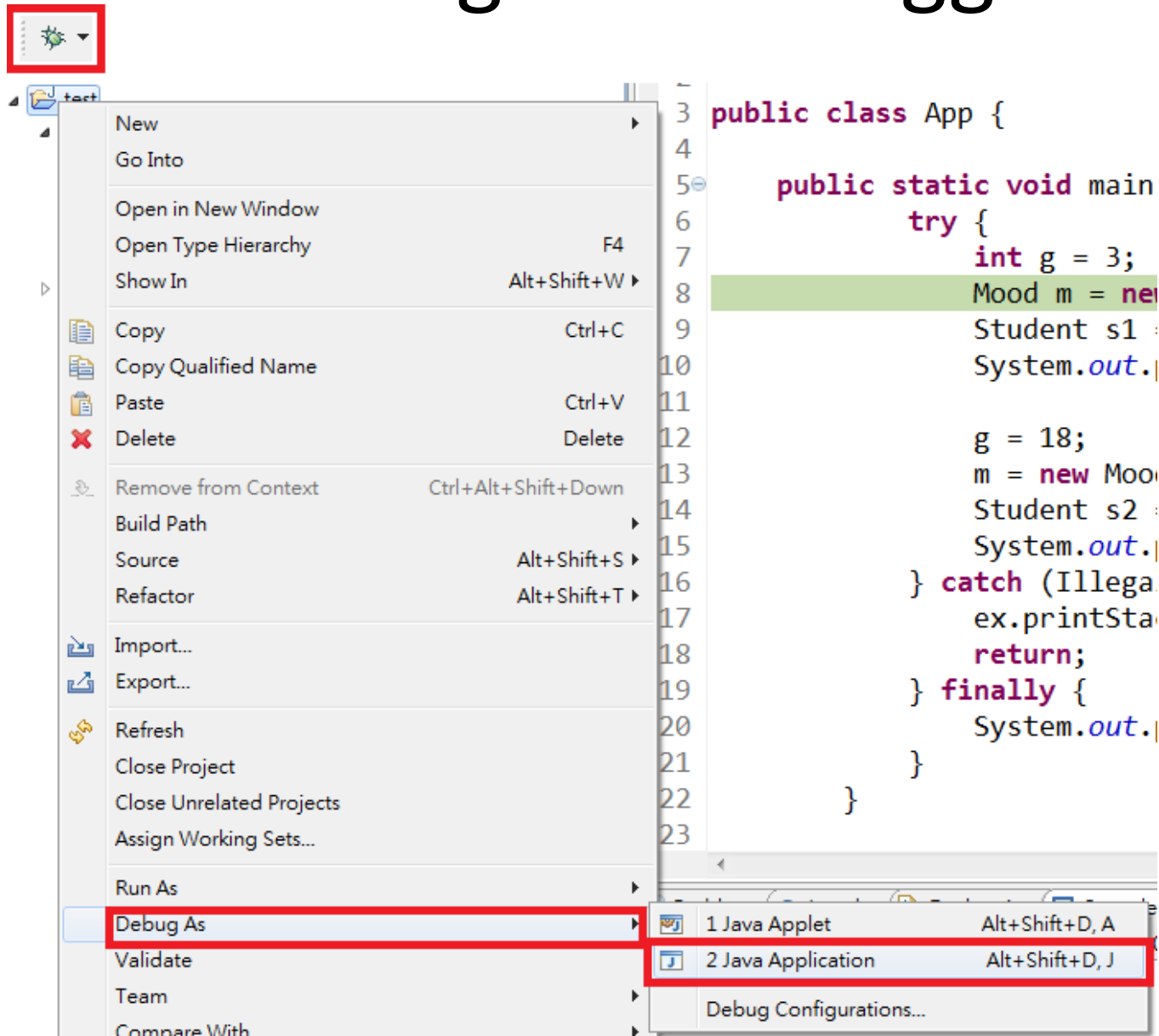
Preferences...

Breakpoint Properties...

Setting Breakpoints

- You should set at least one breakpoint before starting debugging.
- If you have not defined any breakpoints, it will run your program as normal. To debug the program you need to define breakpoints.

Starting the Debugger



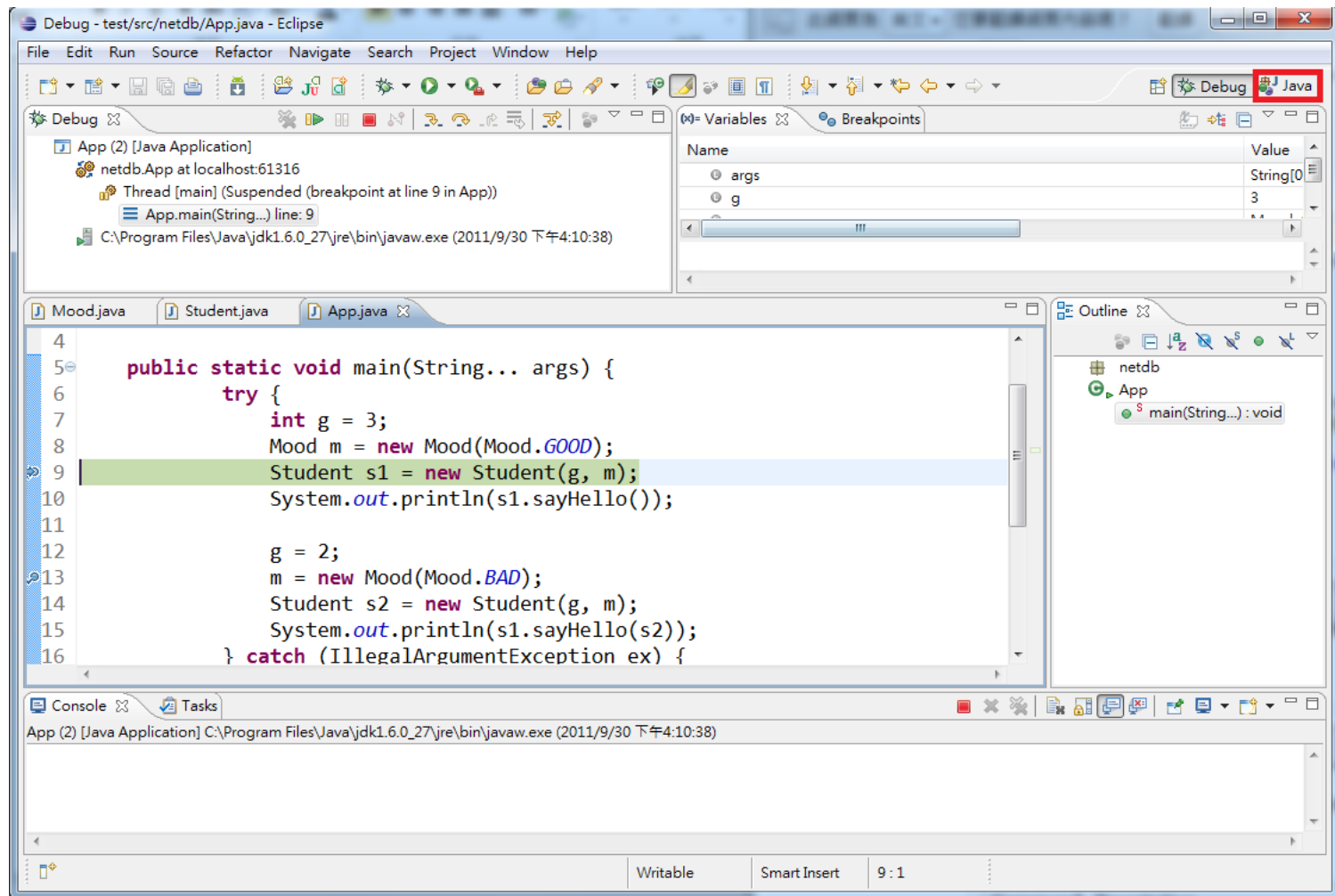
Debug Perspective

- Starting the debugger the first time, Eclipse will ask you if you want to switch to the debug perspective. Answer "yes".



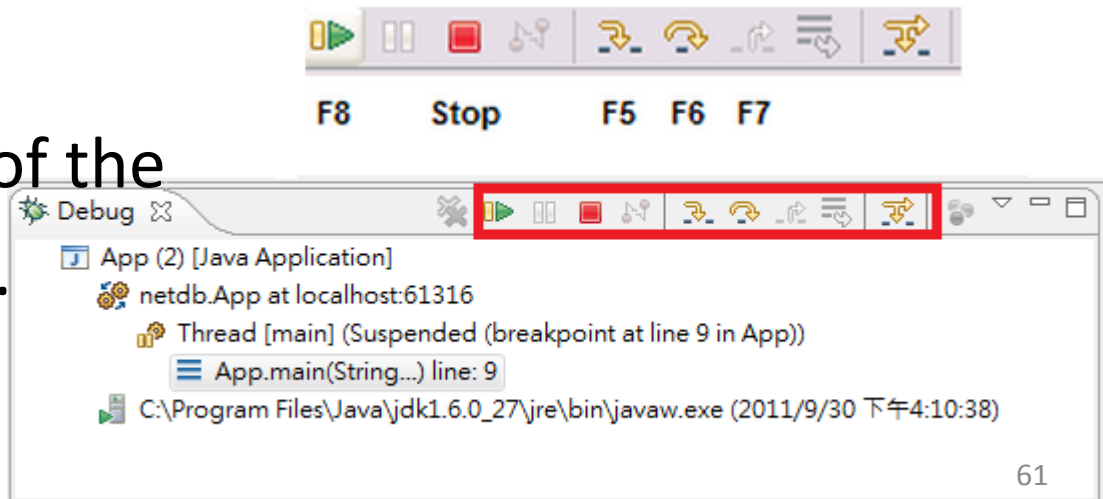
Debug Perspective

- And you will switch to the debug mode.



Step Through the Code

- F5: Step Into
 - If the next step is a method / function this command will jump into the associated code.
- F6: Step Over
 - without entering the associated code.
- F7: Step Return
 - go to the caller of the method/ function.



Outline

- Overview
- Start a Debug Session
 - How To Start
 - Setting Breakpoints
 - Starting the Debugging
- Stack Frame
- Variables(inspect、 watch)
- Exercise

Stack Frame

- The current stack is displayed in the "Debug"

The screenshot shows the Java IDE's Debug window. The left pane displays the 'Thread [main] (Suspended)' with a list of stack frames: 'Student.sayHello() line: 24' and 'App.main(String...) line: 10'. A red box highlights these frames, with a red arrow pointing to the 'Stack' window on the right. The 'Stack' window shows the current stack of active methods: 's1.sayHello(target)' and 'App.main(args)'. The 'App.main(args)' frame is highlighted in green, indicating it is the current frame. The 'Stack' window also shows the arguments for each frame: 'target:Student' and 'tGrade:int' for 's1.sayHello(target)', and 'args:String[]' and 's1:Student' for 'App.main(args)'. The 'App.main(args)' frame has a value of 22 for 'args:String[]' and a null value for 's1:Student'. The 'Student.sayHello()' frame has a value of 15 for 'tGrade:int' and a null value for 'target:Student'.

Stack

Method	Arguments	Value
s1.sayHello(target)	target:Student tGrade:int	● 15
App.main(args)	args:String[] s1:Student	22 ●

Stack frame

```
20      this.mood = mood;
21    }
22
23    public String sayHello() {
24      return "Hello, I am in " + mood + " mood";
25    }
```

Variables (1/2)

- The view "Variables" displays fields and local variables from the current stack.

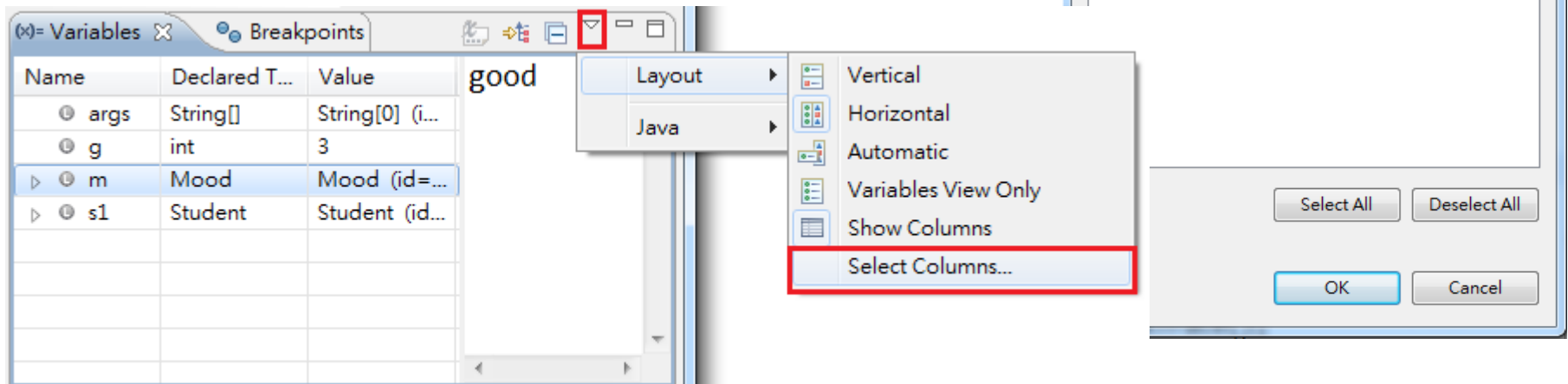
The screenshot illustrates the state of a Java application during execution. It is divided into four main sections:

- Stack:** A diagram showing the current method's frame. It contains local variables: `args:String[]`, `g:int` (value 3), `m:Mood`, and `s1:Student`. Arrows point from these variables to their corresponding objects in the heap.
- Heap:** A diagram showing objects created in memory. It includes:
 - `new Student(grade, mood)` with `grade:int` (3) and `mood:Mood` (pointing to a Mood object).
 - `new Mood(mood)` with `mood:int` (0).
 - A string array `{ "a1", "a2" }`.
- Variables:** A table displaying the current values of local variables and arguments.

Name	Value
args	String[0] (id=16)
g	3
m	Mood (id=18)
s1	Student (id=31)
- Source Code:** The IDE shows the `App` class with the `main` method. The current line of execution is `int g = 3;` (line 7). The code includes creating a `Mood` object, a `Student` object, and printing a message.
- Method Area:** A diagram showing the loaded classes and methods. It includes `netdb`, `App`, and the `main(String...): void` method.

Variables (2/2)

- Via the menu you can change the variables layout and also customize the displayed columns.



Inspect

```
57  
58 public void printScoreList() {  
59     for (int i = 0; i < getEnrollStu  
60         System.out.println(scoreList[i]  
61     }  
62  
63 private void swap(int i, int j) {  
64     double tmp;  
65     tmp = scoreList[i];  
66     scoreList[i] = scoreList[j];  
67     scoreList[j] = tmp;  
68 }  
69
```

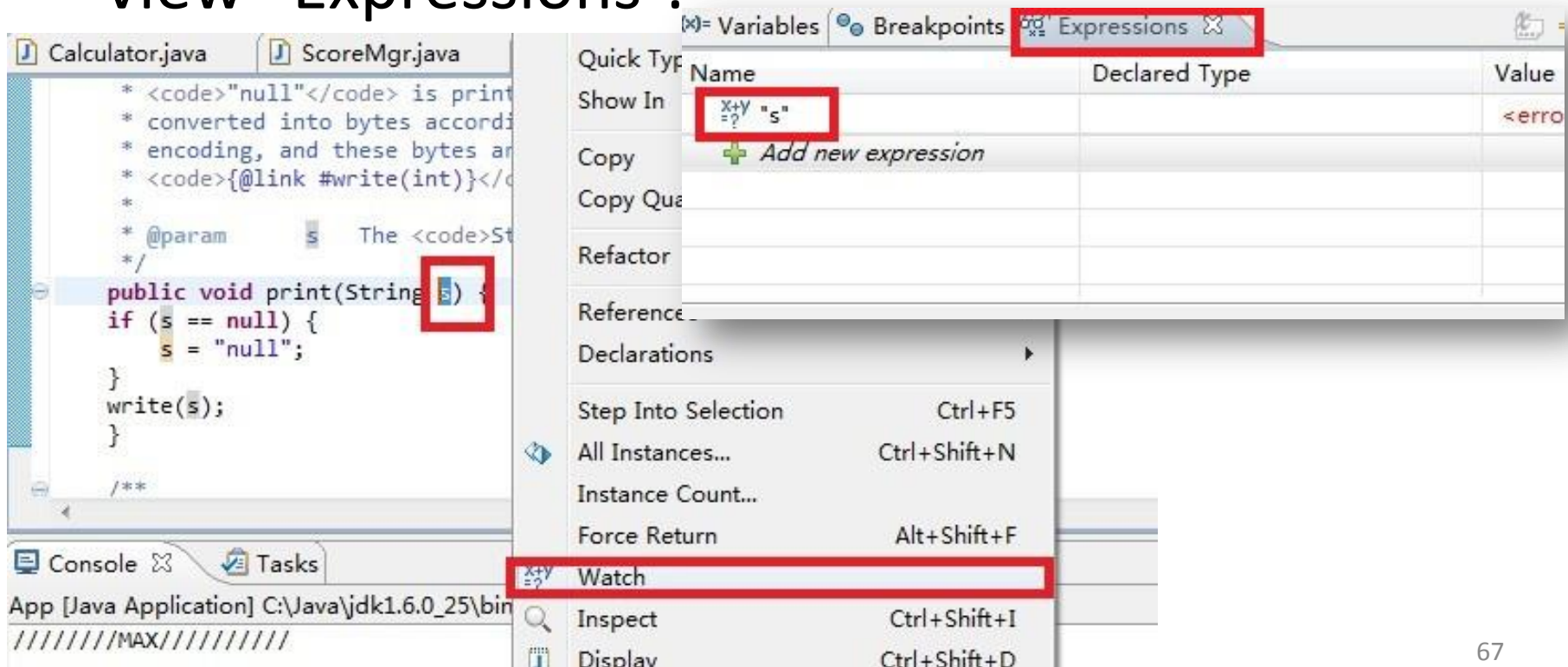
References
Declarations
Add to Snippets...
Step Into Selection Ctrl+F5
All Instances... Ctrl+Shift+N
Instance Count...
Force Return Alt+Shift+F
Watch
Inspect Ctrl+Shift+I
Display Ctrl+Shift+D
Execute Ctrl+U
Run to Line Ctrl+R
Run As

```
System.out.println(scoreList[i]);
```

scoreList= (id=22)
▲ [0]= 96.9
▲ [1]= 65.5
▲ [2]= 77.8
▲ [3]= 63.6
[96.9, 65.5, 77.8, 63.6]

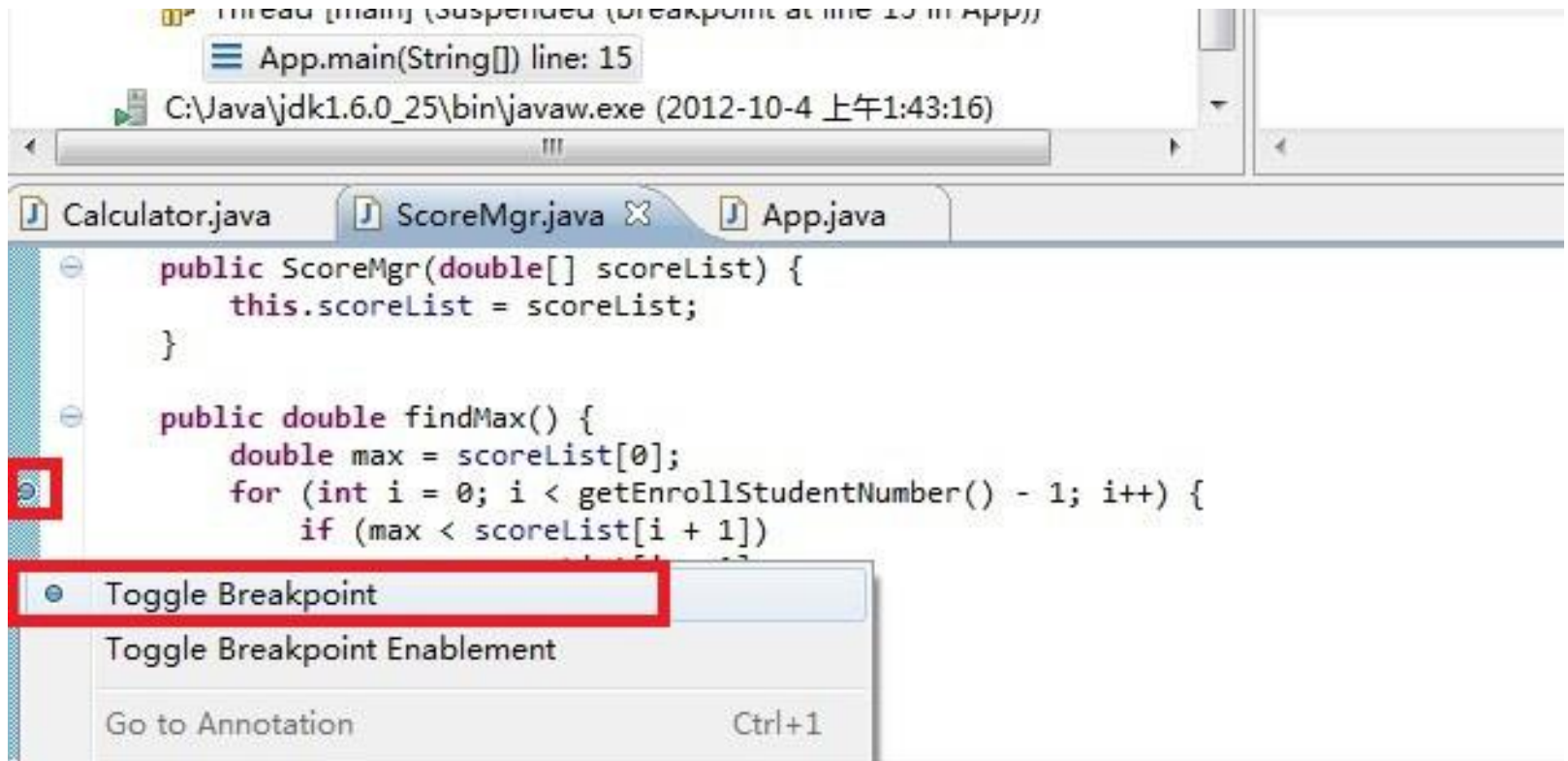
Watch

- You can add variables to watch and watch them all through the debug process in the view “Expressions”.



Tips

- You can add and remove breakpoints in debug mode by the same way.



Reference

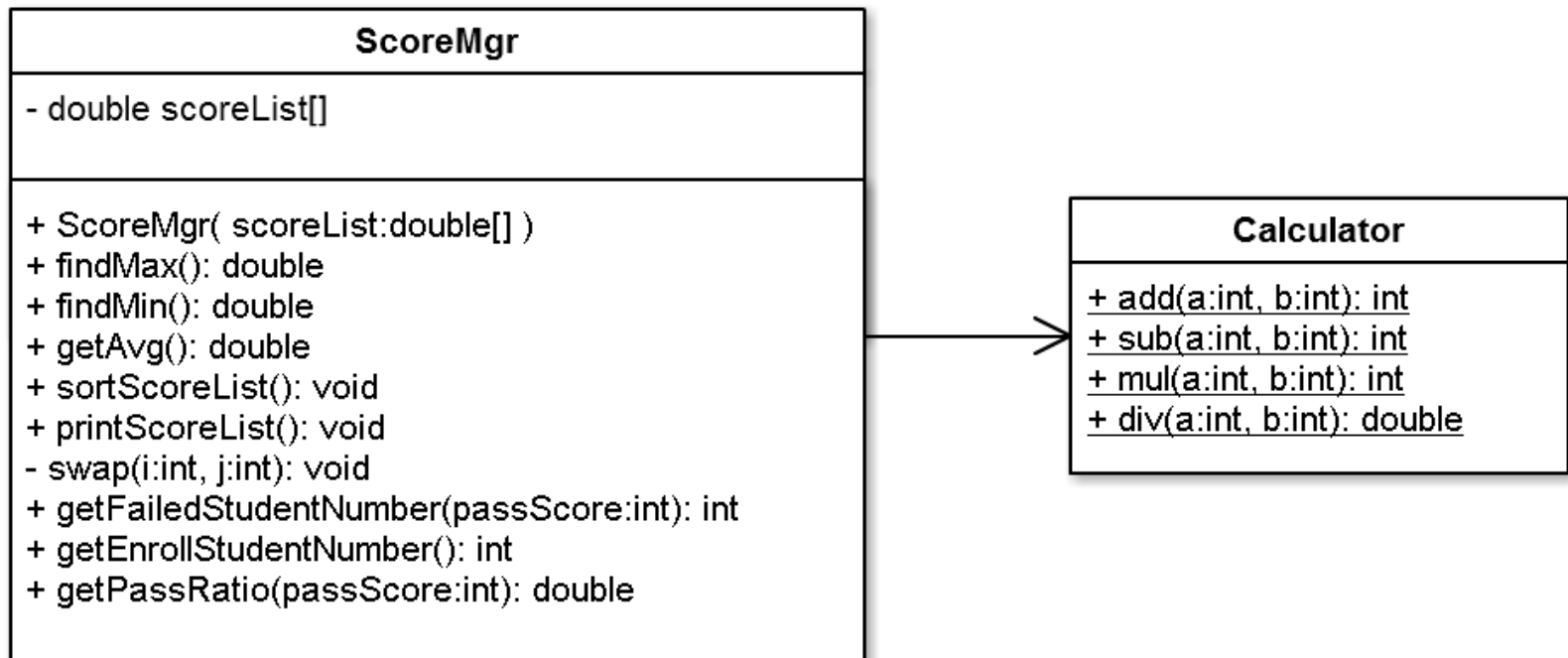
- <http://www.vogella.de/articles/EclipseDebugging/article.html>

Outline

- Overview
- Start a Debug Session
 - How To Start
 - Setting Breakpoints
 - Starting the Debugging
- Stack Frame
- Variables(inspect、 watch)
- Exercise

Exercise (1/4)

- Debug the *ScoreMgr.class* and *Calculator.class*.



Exercise (2/4)

```
public class App {  
    public static void main(String[] args) {  
        double score[]={96.9, 25.59, 75.6, 69.9, 65, 85.75, 88};  
        ScoreMgr scoremgr = new ScoreMgr(score);  
  
        System.out.println("//////////MAX//////////");  
        System.out.println(scoremgr.findMax());  
        System.out.println("//////////MIN//////////");  
        System.out.println(scoremgr.findMin());  
        System.out.println("//////////AVG//////////");  
        System.out.println(scoremgr.getAvg());  
        System.out.println("///FailedStudent#///");  
        System.out.println(scoremgr.getFailedStudentNumber(75));  
        System.out.println("/////PassRatio////////");  
        System.out.println(scoremgr.getPassRatio(75));  
        System.out.println("////////UnSortList////////");  
        scoremgr.printScoreList();  
        System.out.println("////////SortedList////////");  
        scoremgr.sortScoreList();  
        scoremgr.printScoreList();  
  
    }  
}
```

```
//////////MAX//////////  
96.9  
//////////MIN//////////  
25.59  
//////////AVG//////////  
72.39142857142858  
///FailedStudent#///  
3  
/////PassRatio////////  
0.5714285714285714  
////////UnSortList////////  
96.9 25.59 75.6 69.9  
65.0 85.75 88.0  
////////SortedList////////  
96.9 88.0 85.75 75.6  
69.9 65.0 25.59
```


Exercise (3/4)

- There are 4 bugs in this project.
- Use debugger to help you find out all of them and correct it!

Exercise (4/4)

- Checkout BuggyScorer project here on the SVN: (p.18)
 - https://netdb.cs.nthu.edu.tw/svn/courses/software_studio/2013_fall/samples/buggyscorer
- Disconnect the project (p.46)
- Import the project to your SVN account (p.24)
- Correct all the bugs and commit the project (p.22)
- Please show TAs the places you've corrected and the final result of your project.