How to debug in Eclipse
Eclipse Debugging Features

- Breakpoints
- Step into
- Step over
- Step return
- Step Filters
- Watches
- Run to line
- Suspend/Resume/Terminate
Breakpoints

• Breakpoints cause the thread of execution to suspend and return to the debugger

• Setting breakpoints in Eclipse is easy, just double-click the left margin
Step Into

• Executes single (highlighted) statement

• If statement contains call to method, it steps into the method

• To use:
  – Must be in Debug Perspective
  – Click from Debug Toolbar or press F5
Step Over

• Executes single (highlighted) statement
• If statement contains method call, the entire method is executed without stepping through it

• To use:
  – Must be in Debug Perspective
  – Click from Debug toolbar or press F6
Step Return

- All statements in current method are executed
- Return to where method was called
- To use:
  - Must be in Debug Perspective
  - Click from Debug toolbar or press F7
Step with Filters

• User can specify which methods Step Filter will execute and return from

• Usually you would want steps to be made only into methods of your own classes

• To set filters:
  
  – Go through Window > Preferences > Java > Debug > Step Filtering
Watches

- Watches allow you to view (and sometimes change) the value of a variable during execution
- In Eclipse, watches are set automatically on variables in scope
- Watches can be found in the Eclipse Debug perspective in the “Variables” window
- Eclipse allows you to change the value of variables dynamically during runtime
Run to Line

• Equivalent to setting a temporary breakpoint
Suspend/Resume/Terminate

• Suspend pauses the current thread of execution and breaks into the debugger

• Resume resumes a suspended application

• Terminate stops the program and debugging process
DEMO
Create a Java Project

Create a Java project in the workspace or in an external location.

**Project name:** Lab1a

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

**Directory:** C:\Users\robin\eclipse\workspace\Lab1a

**JRE:**
- Use default JRE (Currently 'jre1.6.0_05')
- Use a project specific JRE: jre1.6.0_05
- Use an execution environment JRE: JavaSE-1.6

**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

The wizard will automatically configure the JRE and the project layout based on the existing source.
Java Settings
Define the Java build settings.

Details

- **Create new source folder**: use this if you want to add a new source folder to your project.
- **Link additional source**: use this if you have a folder in the file system that should be included as additional source folder.
- **Configure inclusion and exclusion filters**: specify patterns to the inclusion and exclusion filters instead of including and excluding each folder or file manually.
- **Remove source folder 'src' from build path**: Children of the folder will not be seen.

**Default output folder:**

```
Lab1/bin
```

Browse...
package label;

public class Label {

}
```java
String fileName = args[0];
Scanner inputStream = null;
inputStream =
    new Scanner(new FileInputStream(fileName));

String s;
int sum = 0,
    i = 0;
int [] nums;
nums = new int [20];

while (inputStream.hasNextLine())
{
```
String j;
int sum = 0,
    i = 0;
int [] nums;
nums = new int [20];

while (inputStream.h...
```java
import java.text.*;

public class Lab1a
{
    public static void main(String[] args) throws IOException
    {
        String fileName = args[0];
        Scanner inputStream = null;

        inputStream =
            new Scanner(new FileInputStream(fileName));

        String j;
        int sum = 0,
            i = 0;
        int [] nums;
        nums = new int [20];

        while (inputStream.hasNextLine())
        {
```
```java
public static void main(String[] args) throws IOException {
    String fileName = args[0];
    Scanner inputStream = null;

    inputStream =
        new Scanner(new FileInputStream(fileName));

    String j;
    int sum = 0,
        i = 0;

    <terminated> Lab1 [Java Application] C:\Program Files\Java\jdk1.6.0_05\bin\javaw.exe (Oct 17, 2008 9:51:23 PM)
    S is 5% of the sum
    10 is 10% of the sum
    20 is 20% of the sum
    30 is 30% of the sum
    30 is 30% of the sum
    5 is 5% of the sum

    Slides Prepared by: Farzana Rahman
while (inputStream.hasNextLine())
{
    j = inputStream.nextLine();
    nums[i] = Integer.parseInt(j);
    sum = sum + nums[i];
    i++;

    if (i == 20)
        System.out.println("Too many input values.");
}

inputStream.close();

System.out.println("\nThe sum of the nonzero numbers = " + sum + ");
Functions of F5, F6, F7, and F8 keys in Debug mode

- You can use F5 / F6, F7 and F8 to step through your coding.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F5</td>
<td>Goes to the next step in your program. If the next step is a method / function this command will jump into the associated code.</td>
</tr>
<tr>
<td>F6</td>
<td>F6 will step over the call, e.g. it will call a method / function without entering the associated code.</td>
</tr>
<tr>
<td>F7</td>
<td>F7 will go to the caller of the method/ function. So this will leave the current code and go to the calling code.</td>
</tr>
<tr>
<td>F8</td>
<td>Use F8 to go to the next breakpoint. If no further breakpoint is encountered then the program will normally run</td>
</tr>
</tbody>
</table>
Thank You