

Java2 Lab 1

(Computer system design and application)

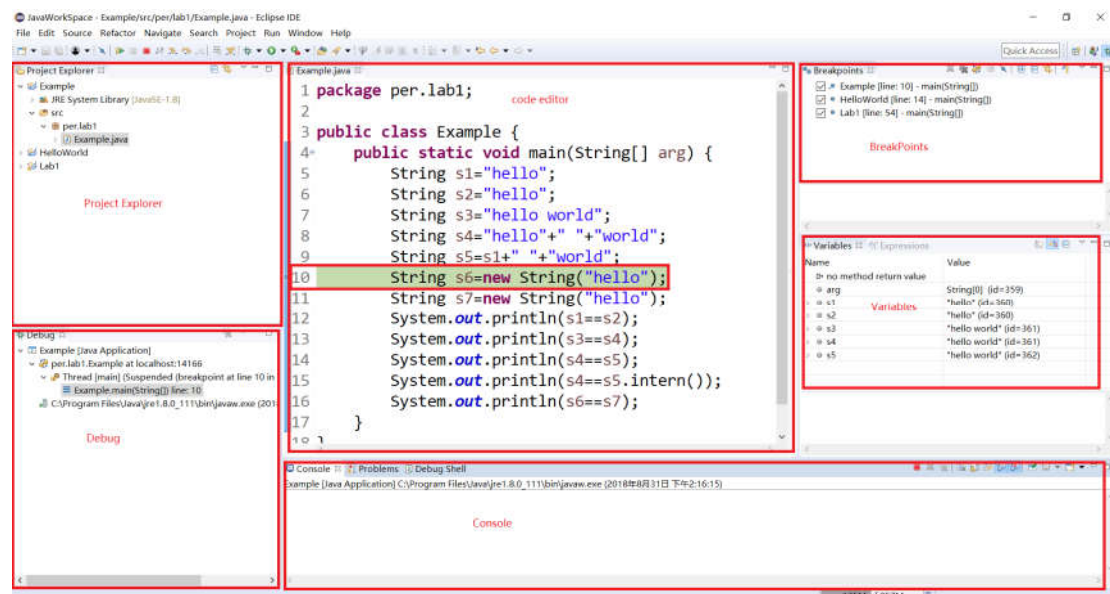
[Experimental Objective]

1. Learn Some tips for debugging program
2. Learn how to install a decompiler to see the source code of a jar file.
3. Understand the difference between compile-time exception and run-time exception

[Tips for debugging program]

The visual debugging interface

Window	Description
Debug	Mainly displays the current thread method call stack, and the line number of the code
Breakpoints	A list of breakpoint; we can easily enable, disable, add or delete a breakpoint, and set breakpoint condition.
Variables	Displays the local variables of the current method. We can modify the variable values.
CodeEditor	Edit the code
Console	Output the logs and results



Other auxiliary windows : Expression, Type hierarchy, Call hierarchy, Search, etc.

● BreakPoints

1. **line breakpoints** : There are two types of line breakpoint: Hit Count and Conditional.
2. **method breakpoints**: Method breakpoints are placed at the first line of the method declaration. By default, the thread interrupts when the method enters, and you can choose Entry or Exit to interrupt when the method enters or exits. One special thing about


method breakpoints is that they can work in the source code of the JDK. Because the JDK removes debugging information, normal breakpoints can't work in it, but method breakpoints can, so the call stack of the method can be viewed in this way.


3. watch breakpoints: watch breakpoints are set on instance variables or static variables of classes. The conditions for breakpoints are Access, Modification and Hit count. Among them, Access and Modification must select at least one. Hit count is optional. When you select Access or Modification, each time a variable is accessed or modified, it is interrupted at the point where it is accessed or modified. If Hit count is selected, it will be interrupted once the variable is accessed or modified for N times.

4. exception breakpoints: suspend when an exception occurs.

5. class load breakpoints: we can type a class load breakpoint on abstract or non-abstract class, it suspends the thread/vm when the class is first loaded or when the first subclass is first loaded.


● Debug


 Skip All Breakpoints :Skip All Breakpoints are set, there will be a slash at all breakpoints, indicating that the breakpoint will be skipped and that the thread will not be suspended at that breakpoint.

 Drop to Frame : lets the program go back to the first line of the current method to restart execution, re-execute the Java stack frame, select a specified stack frame, and then click Drop to Frame to re-enter the specified stack frame.


Attention:

1. cannot drop to the method in the method stack that has been executed.
2. it will not change the original value of a global data.

 Step Return : run out of the current method, during the execution of the called method, using Step Return will jump out of the method and return to the method that called the method after executing all the code of the current method.

 Step Over : When a function encounters a sub-function, it does not go into the sub-function, but stops after the whole execution of the sub-function, that is, execute the sub-function as a whole step

 Step Into: when encountered sub-function, enter and continue to single step execution.

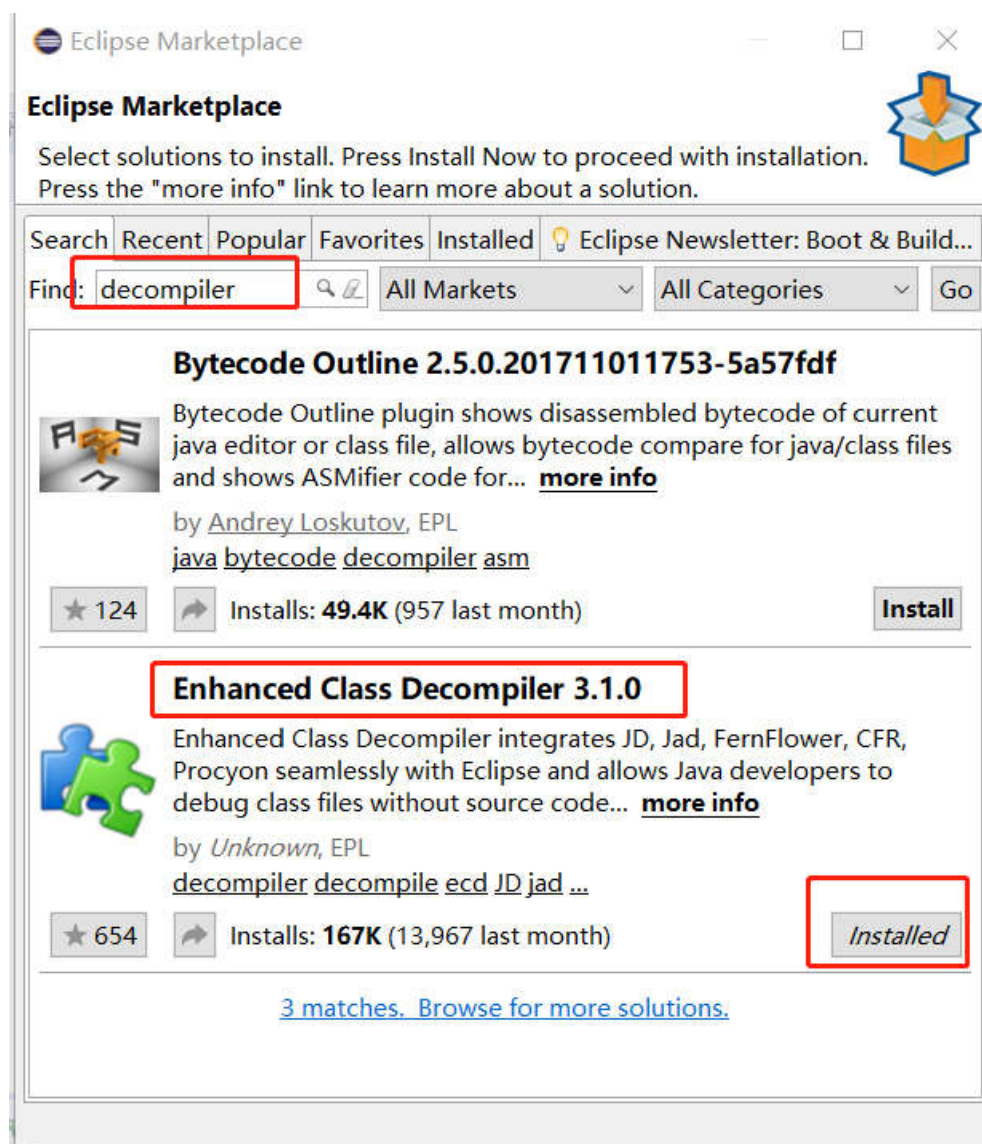
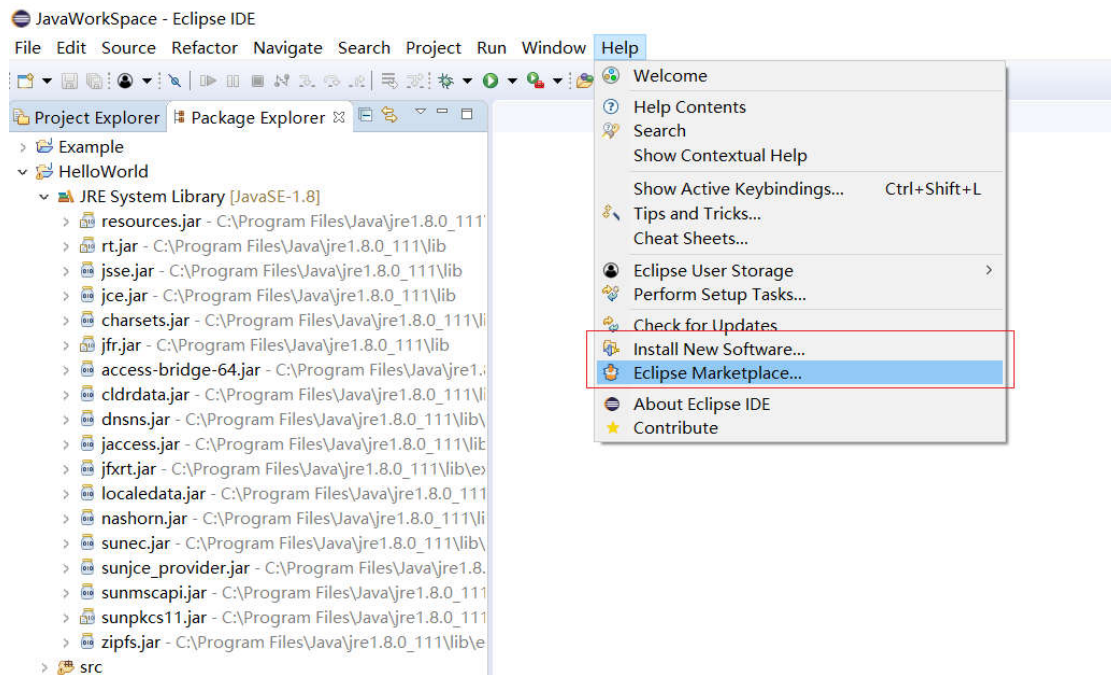
 Resume: resume the thread that is suspended, and jump directly from the current location to the next breakpoint.

 Terminate :Terminate debugging of local programs through Terminate command

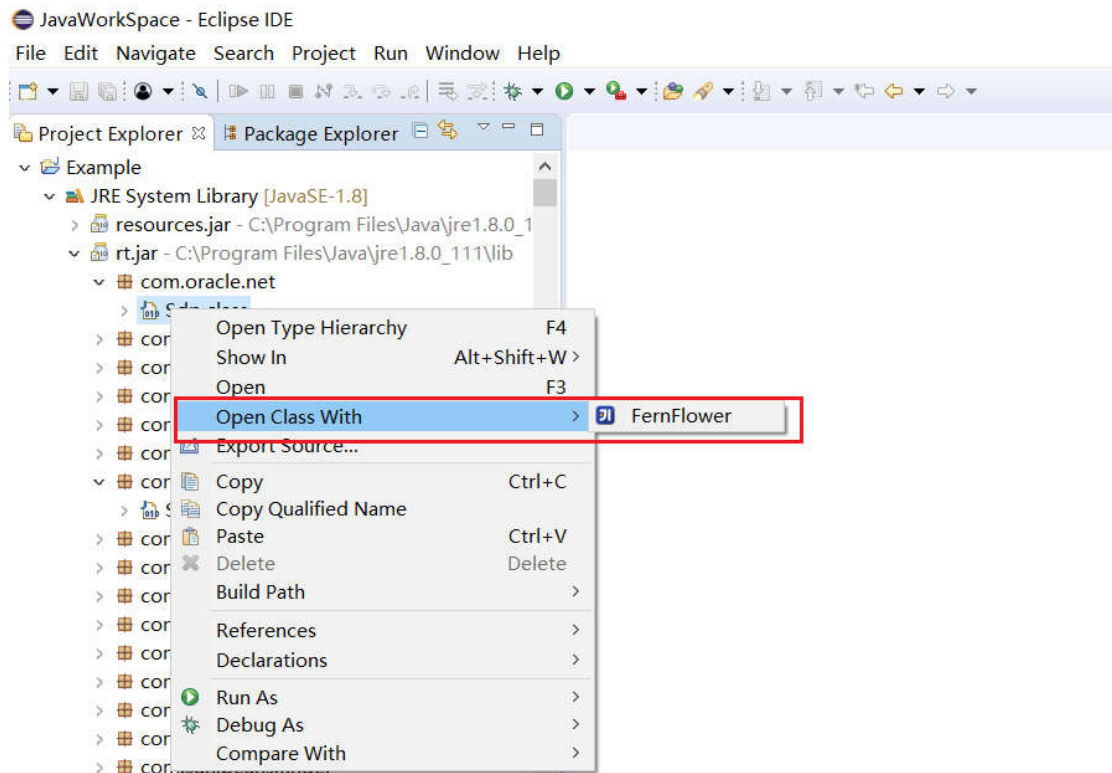
[Decompiler]

When we want to look at the source code in the jar package, we need to decompile the .class file in the jar package with Decompiler to get the source code

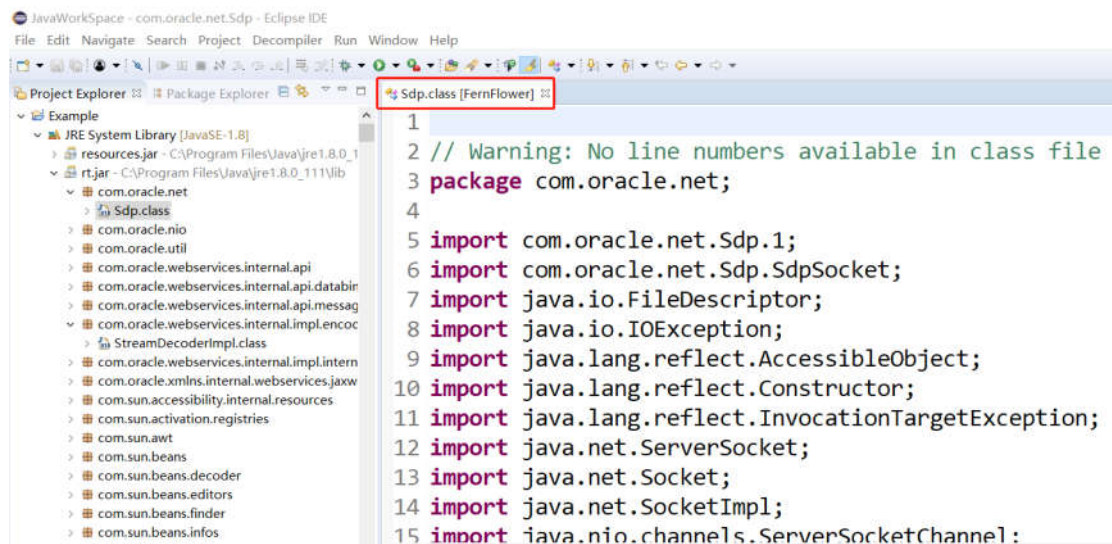
Installation: Main menu->Help->Install New Software/Eclipse Marketplace



Once the installation is complete, we will see an additional *Open class with* option in the right-click menu of the .class file.

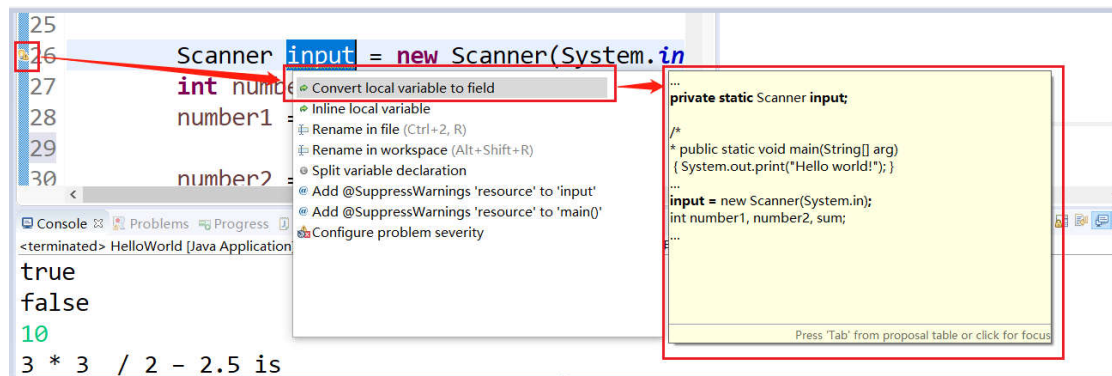


Open class with FernFlower, we can see the source code of the .class file.



Quick solution for error and warning

Warning/Error: once there is a warning/error in our code, you can click the yellow bulb logo to get a quick solution.



Link with editor.

Switch between perspectives. We can also customize perspective window.

[Example]

There are runtime exceptions/unchecked exceptions and non-runtime exceptions/checked exceptions.

Compile-time error

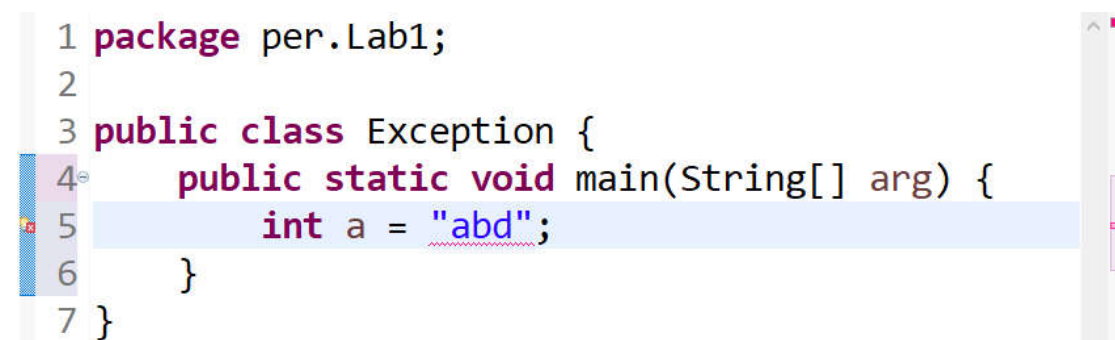
Compile the following code in CMD window

```
public class Exception{
    public static void main(String[] arg){
        int a = "abd";
    }
}
```

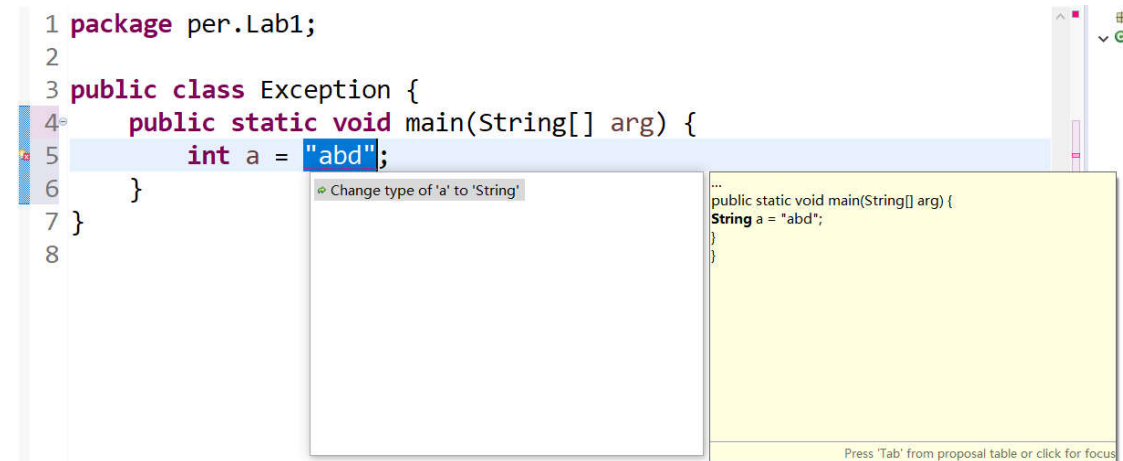
Result:

```
D:\demo>javac Exception.java
Exception.java:3: 错误: 不兼容的类型: String无法转换为int
    int a = "abd";
           ^
1 个错误
```

If we edit the code in eclipse, the error is directly point out by the editor.



And by click the yellow bulb at the left side of the line, we can get a quick fix solution.



Run-time error

Compile the following code in CMD window

```

public class Exception2{
    public static void main(String args[]) {
        Object a = null;
        String aString = a.toString();
    }
}

```

Result:

```

D:\demo>javac Exception2.java

D:\demo>

```

Run the .class file in CMD window

Result:

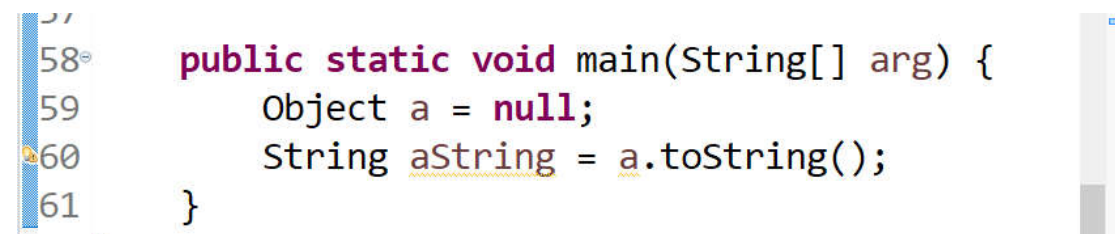
```

D:\demo>java Exception2
Exception in thread "main" java.lang.NullPointerException
    at Exception2.main(Exception2.java:5)

D:\demo>

```

If we edit the code in eclipse, the error has not been point out by the editor.



Run result:

```

Exception in thread "main" java.lang.NullPointerException
    at per.Lab1.Lab1.main(Lab1.java:60)

```

[Assignment]

No assignment for the 1st class.