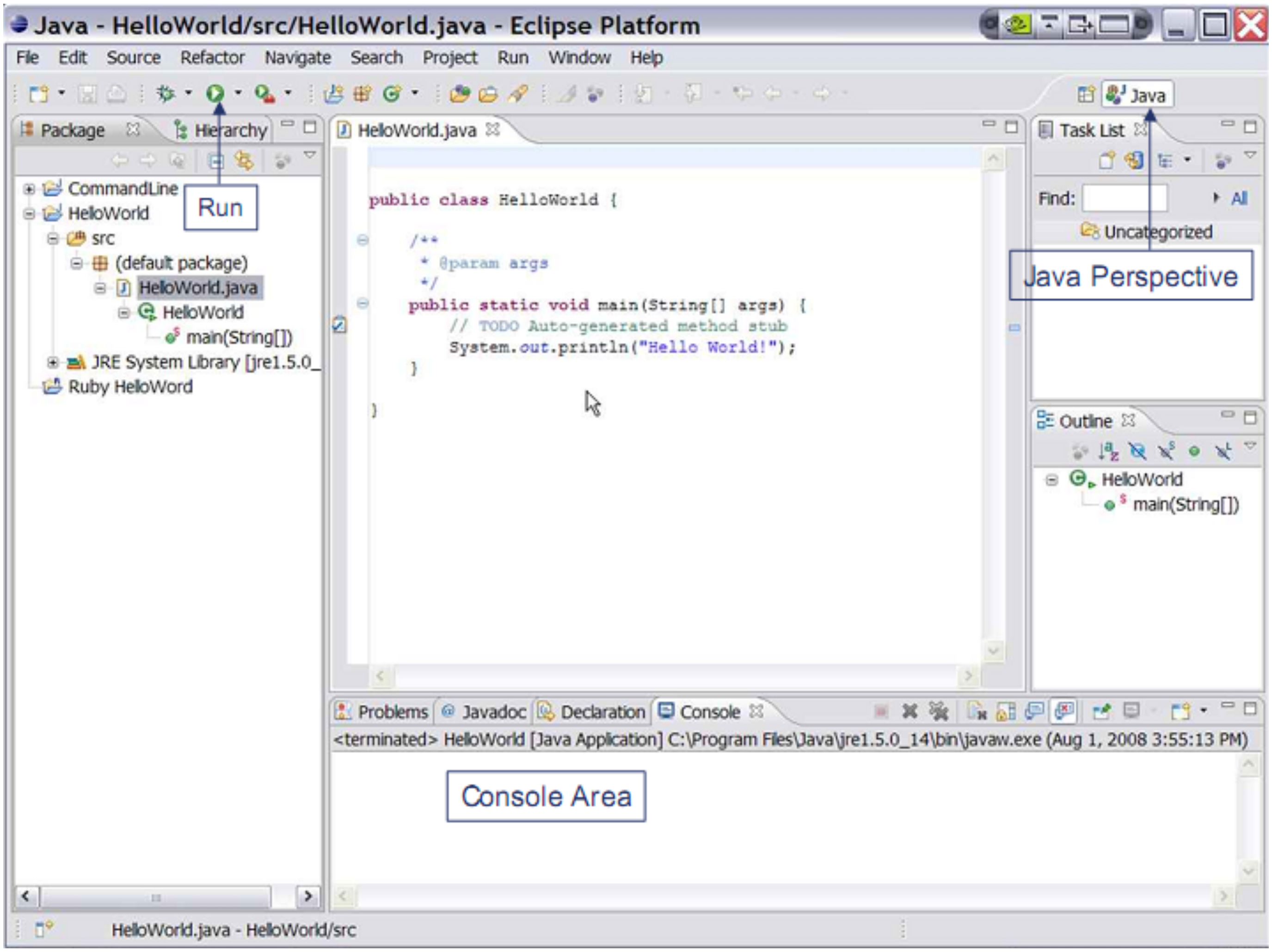
Object-Oriented Language and Theory

NGUYEN Thi Thu Trang, trangntt@soict.hust.edu.vn

Lab 2: Java basics and UML

* Introduction to Eclipse / Netbean



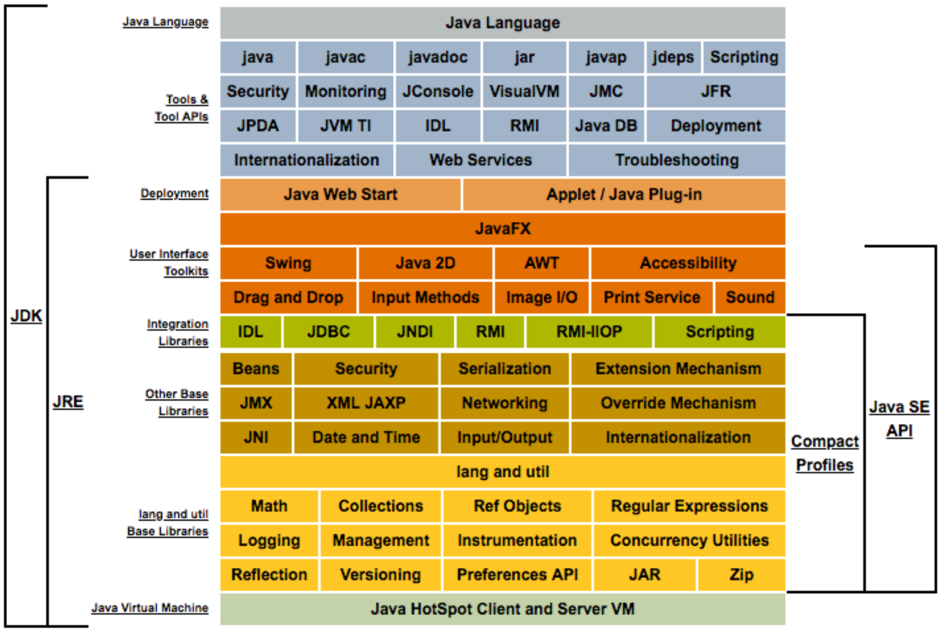
```
public class HelloWorld {

    /**
     * @param args
     */
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.println("Hello World!");
    }

}
```

```
<terminated> HelloWorld [Java Application] C:\Program Files\Java\jre1.5.0_14\bin\javaw.exe (Aug 1, 2008 3:55:13 PM)
```

* Javadocs help:

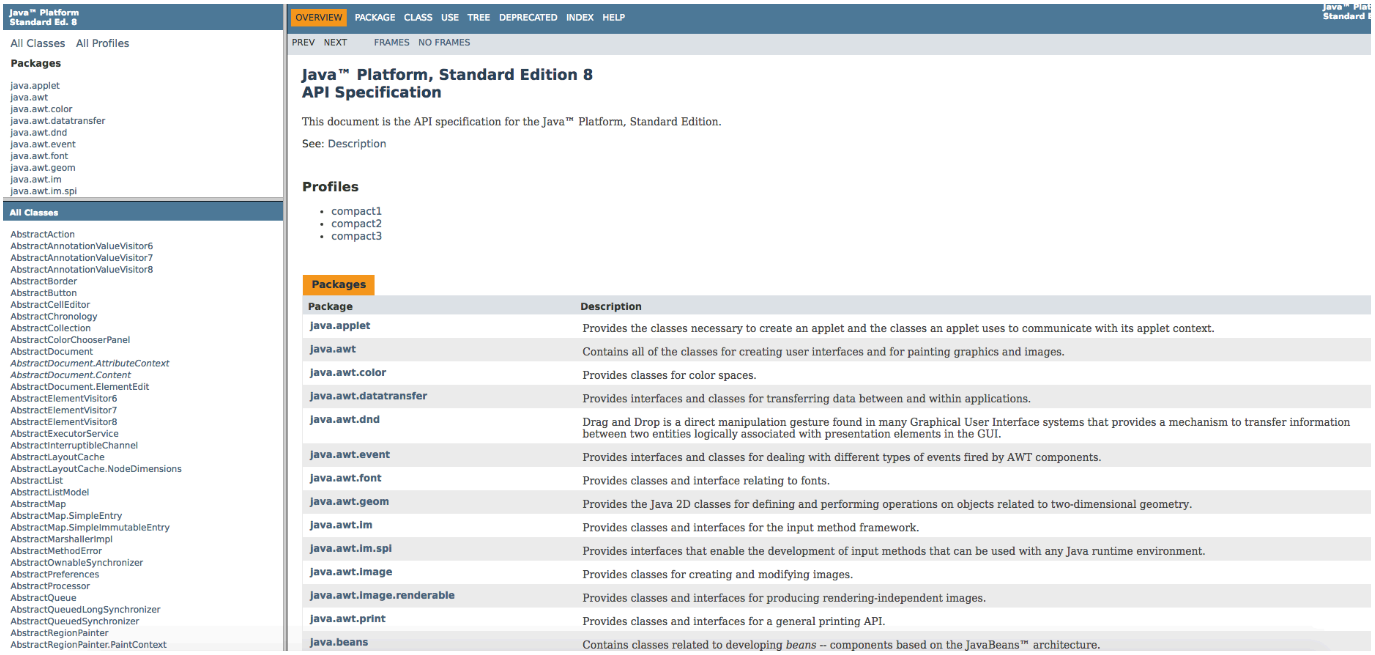
- Open index.html in the docs folder (download from https://www.oracle.com/technetwork/java/javase/documentation/jdk8-doc-downloads-2133158.html)

Description of Java Conceptual Diagram



JDK
JRE
Java Language: Java Language
Tools & Tool APIs:
java | javac | javadoc | jar | javap | jdeps | Scripting
Security | Monitoring | JConsole | VisualVM | JMC | JFR
JPDA | JVM TI | IDL | RMI | Java DB | Deployment
Internationalization | Web Services | Troubleshooting
Deployment: Java Web Start | Applet / Java Plug-in
User Interface Toolkits:
JavaFX
Swing | Java 2D | AWT | Accessibility
Drag and Drop | Input Methods | Image I/O | Print Service | Sound
Integration Libraries: IDL | JDBC | JNDI | RMI | RMI-IIOP | Scripting
Other Base Libraries:
Beans | Security | Serialization | Extension Mechanism
JMX | XML JAXP | Networking | Override Mechanism
JNI | Date and Time | Input/Output | Internationalization
lang and util Base Libraries:
lang and util
Math | Collections | Ref Objects | Regular Expressions
Logging | Management | Instrumentation | Concurrency Utilities
Reflection | Versioning | Preferences API | JAR | Zip
Java Virtual Machine: Java HotSpot Client and Server VM
Compact Profiles
Java SE API

- Click the link Java SE API



Java™ Platform
Standard Ed. 8

All Classes All Profiles

Packages

java.applet
java.awt
java.awt.color
java.awt.datatransfer
java.awt.dnd
java.awt.event
java.awt.font
java.awt.geom
java.awt.im
java.awt.im.spi

All Classes

AbstractAction
AbstractAnnotationValueVisitor6
AbstractAnnotationValueVisitor7
AbstractAnnotationValueVisitor8
AbstractBorder
AbstractButton
AbstractCellEditor
AbstractChronology
AbstractCollection
AbstractColorChooserPanel
AbstractDocument
AbstractDocument.AttributeContext
AbstractDocument.Content
AbstractDocument.ElementEdit
AbstractElementVisitor6
AbstractElementVisitor7
AbstractElementVisitor8
AbstractExecutorService
AbstractInterruptibleChannel
AbstractLayoutCache
AbstractLayoutCache.NodeDimensions
AbstractList
AbstractListModel
AbstractMap
AbstractMap.SimpleEntry
AbstractMap.SimpleImmutableEntry
AbstractMarshallerImpl
AbstractMethodError
AbstractOwnableSynchronizer
AbstractPreferences
AbstractProcessor
AbstractQueue
AbstractQueuedLongSynchronizer
AbstractQueuedSynchronizer
AbstractRegionPainter
AbstractRegionPainter.PaintContext

OVERVIEW PACKAGE CLASS USE TREE DEPRECATED INDEX HELP
PREV NEXT FRAMES NO FRAMES
Java™ Plat
Standard E

Java™ Platform, Standard Edition 8
API Specification

This document is the API specification for the Java™ Platform, Standard Edition.

See: Description

Profiles

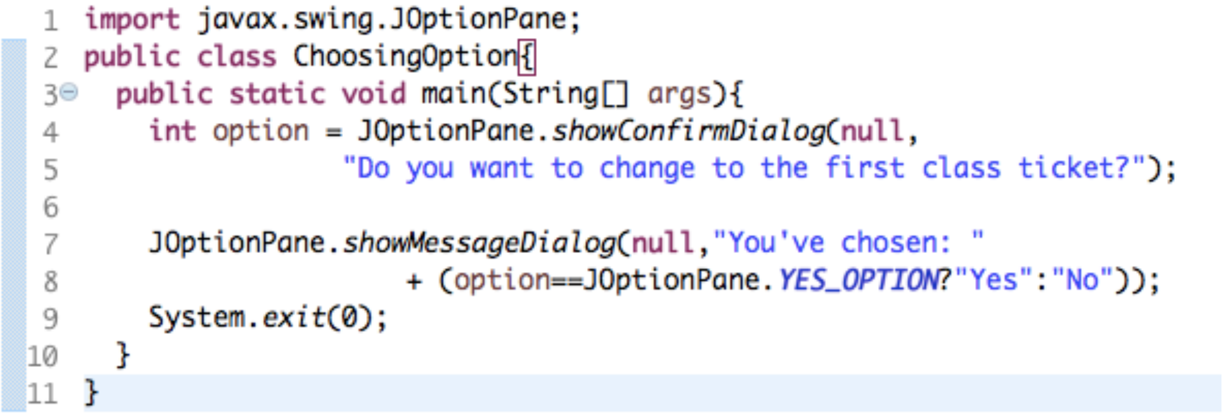
- compact1
- compact2
- compact3

Packages

Package	Description
java.applet	Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.
java.awt	Contains all of the classes for creating user interfaces and for painting graphics and images.
java.awt.color	Provides classes for color spaces.
java.awt.datatransfer	Provides interfaces and classes for transferring data between and within applications.
java.awt.dnd	Drag and Drop is a direct manipulation gesture found in many Graphical User Interface systems that provides a mechanism to transfer information between two entities logically associated with presentation elements in the GUI.
java.awt.event	Provides interfaces and classes for dealing with different types of events fired by AWT components.
java.awt.font	Provides classes and interface relating to fonts.
java.awt.geom	Provides the Java 2D classes for defining and performing operations on objects related to two-dimensional geometry.
java.awt.im	Provides classes and interfaces for the input method framework.
java.awt.im.spi	Provides interfaces that enable the development of input methods that can be used with any Java runtime environment.
java.awt.image	Provides classes for creating and modifying images.
java.awt.image.renderable	Provides classes and interfaces for producing rendering-independent images.
java.awt.print	Provides classes and interfaces for a general printing API.
java.beans	Contains classes related to developing *beans* -- components based on the JavaBeans™ architecture.

- The top left frame: all packages in Java API
- The bottom left frame: corresponding classes in the chosen above package
- The right frame: Detail information
- Click to a frame, and find the necessary information (Ctrl + F)

1. Write, compile and run the ChoosingOption program:

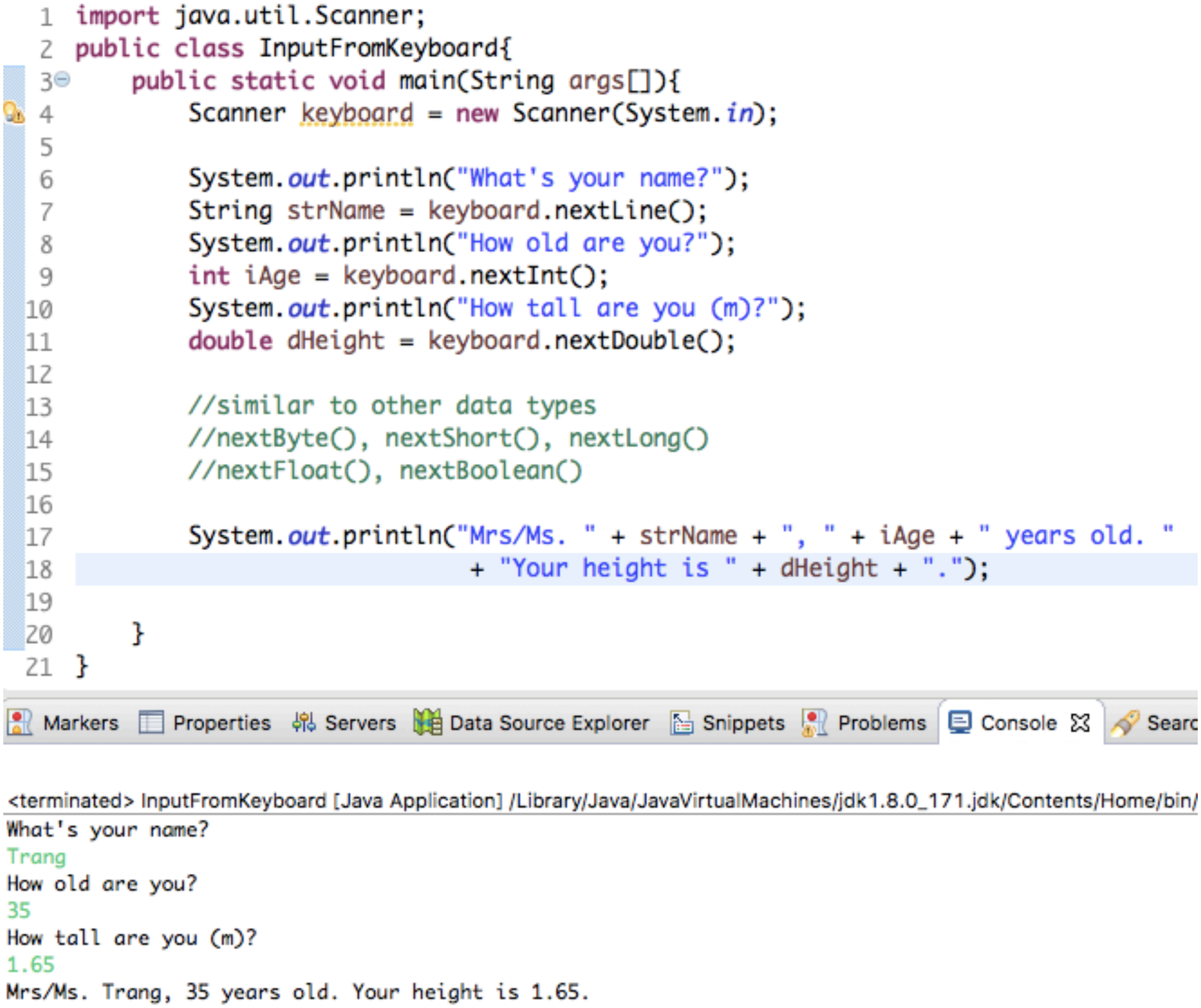


```java
import javax.swing.JOptionPane;
public class ChoosingOption{
  public static void main(String[] args){
    int option = JOptionPane.showConfirmDialog(null,
                  "Do you want to change to the first class ticket?");

    JOptionPane.showMessageDialog(null,"You've chosen: "
                      + (option==JOptionPane.YES_OPTION?"Yes":"No"));
    System.exit(0);
  }
}
```

Questions:

- What happens if users choose “Cancel”?
- How to customize the options to users, e.g. only two options: “Yes” and “No”, OR “I do” and “I don’t” (Suggestion: Use Javadocs or using Eclipse/Netbean IDE help).

2. Write a program for input/output from keyboard



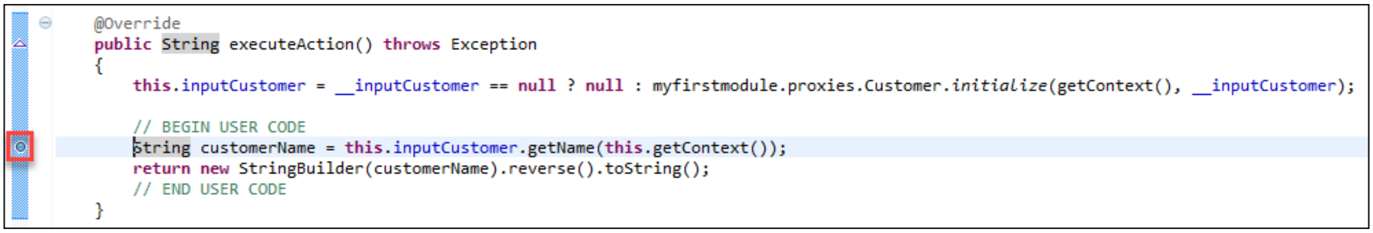
```java
import java.util.Scanner;
public class InputFromKeyboard{
    public static void main(String args[]){
        Scanner keyboard = new Scanner(System.in);

        System.out.println("What's your name?");
        String strName = keyboard.nextLine();
        System.out.println("How old are you?");
        int iAge = keyboard.nextInt();
        System.out.println("How tall are you (m)?");
        double dHeight = keyboard.nextDouble();

        //similar to other data types
        //nextByte(), nextShort(), nextLong()
        //nextFloat(), nextBoolean()

        System.out.println("Mrs/Ms. " + strName + ", " + iAge + " years old. "
                            + "Your height is " + dHeight + ".");

    }
}
```

```
<terminated> InputFromKeyboard [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_171.jdk/Contents/Home/bin/
What's your name?
Trang
How old are you?
35
How tall are you (m)?
1.65
Mrs/Ms. Trang, 35 years old. Your height is 1.65.
```

3. Use debug to run step by step or go to a checkpoint in a program

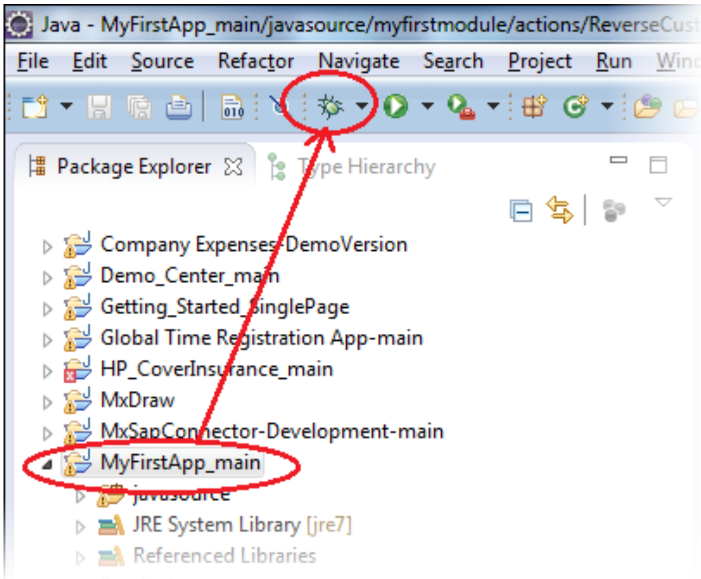
Video: https://www.youtube.com/watch?v=9gAjIQc4bPU&t=8s

3.1. Setting breakpoints: Place the cursor on the line that needs debugging, hold down Ctrl+Shift, and press B to enable a breakpoint. A blue dot in front of the line will appear.



```
@Override
public String executeAction() throws Exception
{
    this.inputCustomer = __inputCustomer == null ? null : myfirstmodule.proxies.Customer.initialize(getContext(), __inputCustomer);

    // BEGIN USER CODE
    String customerName = this.inputCustomer.getName(this.getContext());
    return new StringBuilder(customerName).reverse().toString();
    // END USER CODE
}
```

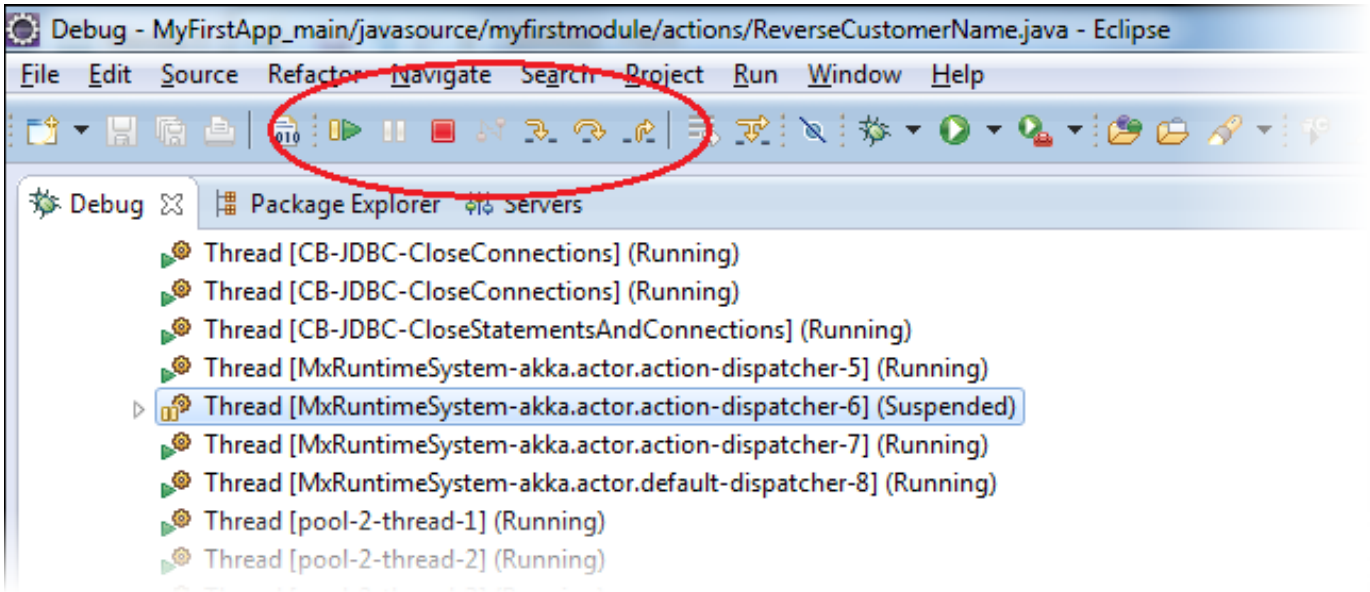
3.2. Debugging in Eclipse: Select the project root node in the package explorer and click the debug icon in the Eclipse toolbar. The application will now be started with Eclipse attached as debugger.


Java - MyFirstApp_main/javasource/myfirstmodule/actions/ReverseCust
File Edit Source Refactor Navigate Search Project Run Wind
Package Explorer
Type Hierarchy
Company Expenses-DemoVersion
Demo_Center_main
Getting_Started_SinglePage
Global Time Registration App-main
HP_CoverInsurance_main
MxDraw
MxSapConnector-Development-main
MyFirstApp_main
javasource
JRE System Library [jre7]
Referenced Libraries

- As soon as the deployment process is ready, open the application in your browser and trigger the Java action:
    - As an end-user of the application, you will see a progress bar on your application
    - As a developer, you will see the Eclipse icon flashing on the Windows task bar
- Open Eclipse. You should now see the “debug” perspective of Eclipse.

3.3. Step into or Step over or Step return/Resume

- Click Step into (or press F5) or Step over (or press F6) to move on the next step in the microflow.

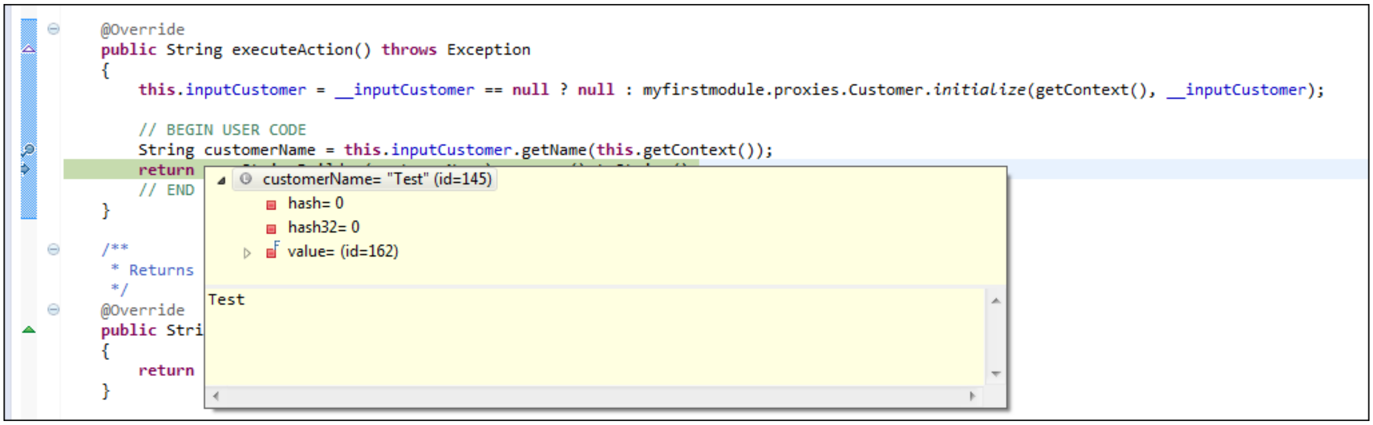

Debug - MyFirstApp_main/javasource/myfirstmodule/actions/ReverseCustomerName.java - Eclipse
File
Edit
Source
Refactor
Navigate
Search
Project
Run
Window
Help
Debug
Package Explorer
Servers

```
Thread [CB-JDBC-CloseConnections] (Running)
Thread [CB-JDBC-CloseConnections] (Running)
Thread [CB-JDBC-CloseStatementsAndConnections] (Running)
Thread [MxRuntimeSystem-akka.actor.action-dispatcher-5] (Running)
Thread [MxRuntimeSystem-akka.actor.action-dispatcher-6] (Suspended)
Thread [MxRuntimeSystem-akka.actor.action-dispatcher-7] (Running)
Thread [MxRuntimeSystem-akka.actor.default-dispatcher-8] (Running)
Thread [pool-2-thread-1] (Running)
Thread [pool-2-thread-2] (Running)
```

- With debugger options, the difference between "Step into" and "Step over" is only noticeable if you run into a function call :
  - "Step into" (F5) means that the debugger steps into the function
  - "Step over" (F6) just moves the debugger to the next line in the same Java action
- With "Step Return" (pressing F7), you can instruct the debugger to leave the function; this is basically the opposite of "Step into."
- Clicking "Resume" (F8) instructs the debugger to continue until it reaches another breakpoint.

3.4. Popup window

Place your cursor on any of the variables in the Java action to see its value in a pop-up window.



```
@Override
public String executeAction() throws Exception
{
    this.inputCustomer = __inputCustomer == null ? null : myfirstmodule.proxies.Customer.initialize(getContext(), __inputCustomer);

    // BEGIN USER CODE
    String customerName = this.inputCustomer.getName(this.getContext());
    return
    // END
}

/**
 * Returns
 */
@Override
public Stri
{
    return
}
```

```
customerName= "Test" (id=145)
    hash= 0
    hash32= 0
    value= (id=162)
Test
```

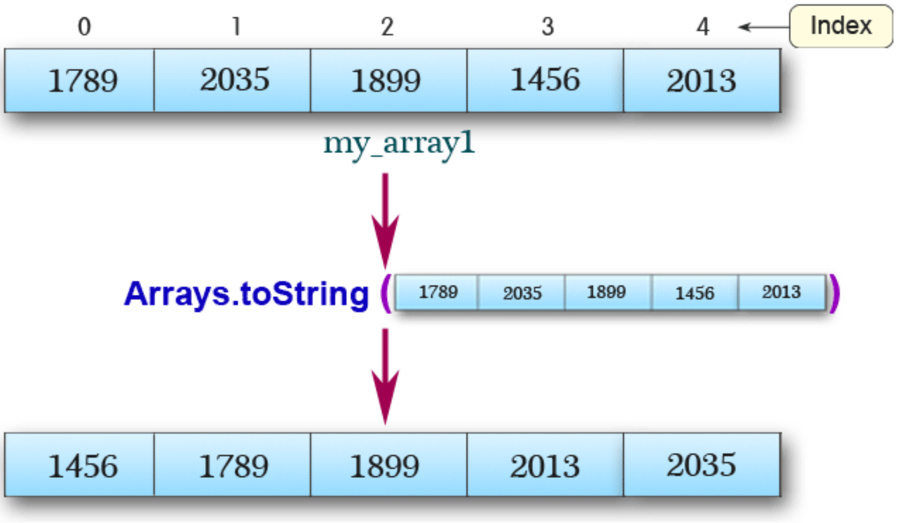
4. Write a program to display a triangle with a height of n stars (*), n is entered by users.

E.g. n=5:

```
    *
   ***
  *****
 ********
***********
```

5. Write a program to display the number of days of a month, which is entered by users (both month and year). If it is an invalid month/year, ask the user to enter again.

6. Write a Java program to sort a numeric array, and calculate the sum and average value of array elements.



7. Write a Java program to add two matrices of same size.