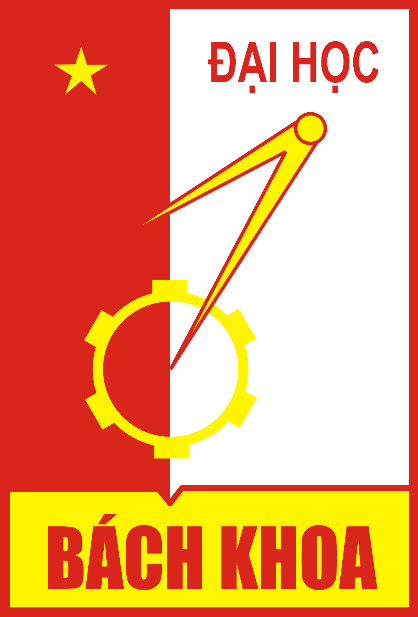
**ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**



BÁO CÁO THỰC HÀNH

IT3103 – 744527 – 2024.1

**BÀI THỰC HÀNH – LAB 1**

|  |  |
| --- | --- |
| **Họ và tên SV:** | **Trịnh Hữu An** |
| **MSSV:** | **20225743** |
| **Lớp:** | **Việt Nhật 03 – K67** |
| **GVHD:** | **Lê Thị Hoa** |
| **HTDH:** | **Đặng Mạnh Cường** |

Hà Nội, tháng 9 năm 2024

**Mục Lục**

# First Programs

## The Very First Java Programs

### Write, compile the first Java application:

**Step 1:** **Create a new file.** From the Notepad interface, choose File → New File.

**Step 2: Save the file.** From the Notepad interface, choose File → Save. Browse the desired directory, change the file name to “*HelloWorld.java”* and hit the “*Save*” button.

**Step 3: Write the source code.** The source code is shown in Figure 5.

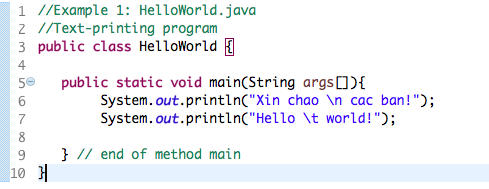


Figure . The First Java Application

**Step 4: Compile.** On a Command Prompt or a Terminal, change the current working directory[[1]](#footnote-2) into the directory where we have saved the source code. Then issue the following commands**.**

**$ javac HelloWorld.java**

**$ java HelloWorld**

### Write, compile the first dialog Java program

**Step 1:** **Create a new file.** From the Notepad interface, choose File → New File.

**Step 2: Save the file.** From the Notepad interface, choose File → Save. Browse the desired directory, change the file name to “*FirstDialog.java,”* and click the “*Save*” button.

**Step 3: Write the source code.** The source code is shown in Figure 6



Figure . The First Dialog Java Application

**Step 4: Compile.** On a Command Prompt or a Terminal, change the current working directory into the directory where we have saved the source code. Issue the following commands.

**$ javac FirstDialog.java**

**$ java FirstDialog**

### Write, compile the first input dialog Java application

**Step 1:** **Create a new file.** From the Notepad interface, choose File → New File.

**Step 2: Save the file.** From the Notepad interface, choose File → Save. Browse the desired directory, change the file name to “*HelloNameDialog.java,”* and click the “*Save*” button.

**Step 3: Write the source code.** The source code is shown in Figure 7

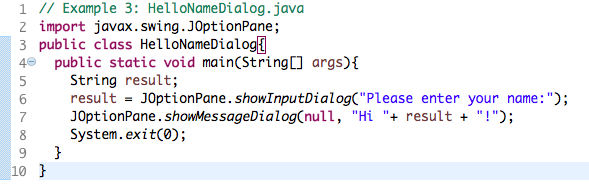


Figure . The First Input Dialog Java Application

**Step 4: Compile.** On a Command Prompt or a Terminal, change the current working directory into the directory where we have saved the source code. Issue the following commands.

**$ javac HelloNameDialog.java**

**$ java HelloNameDialog**

### Write, compile, and run the following example:

**Step 1:** **Create a new file.** From the Notepad interface, choose File → New File.

**Step 2: Save the file.** From the Notepad interface, choose File → Save. Browse the desired directory, change the file name to “*ShowTwoNumbers.java,”* and click the “*Save*” button.

**Step 3: Write the source code.** The source code is shown in Figure 8

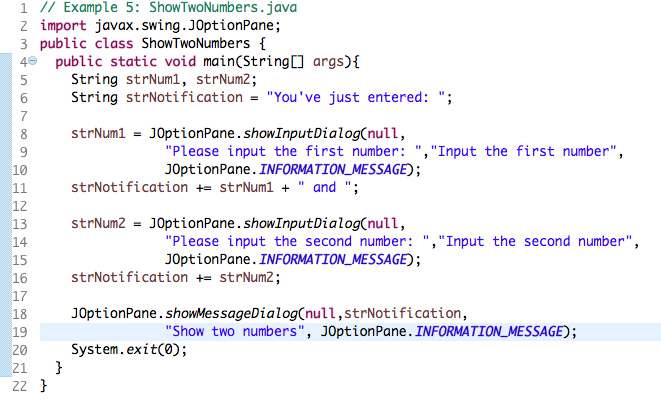


Figure . Java Application showing two entered numbers and their sum

**Step 4: Compile.** On a Command Prompt or a Terminal, change the current working directory into the directory where we have saved the source code. Issue the following commands.

**$ javac ShowTwoNumbers.java**

**$ java ShowTwoNumbers**

### Write a program to calculate sum, difference, product, and quotient of 2 double numbers which are entered by users.

**Notes**

* To convert from String to double, you can use

**double num1 = Double.parseDouble(strNum1)**

* Check the divisor of the division

### Write a program to solve:

*For simplicity, we only consider the real roots of the equations in this task.*

* **The first-degree equation (linear equation) with one variable**

**Note**: A first-degree equation with one variable can have a form such as .

You should handle the case where the user input value 0 for a.

* **The system of first-degree equations (linear system) with two variables**

**Note**: A system of first-degree equations with two variables and can be written as follows.

You should handle the case where the values of the coefficients produce infinitely many solutions and the case where they produce no solution.

**Hint**:

Use the following determinants:

* **The second-degree equation with one variable**

**Note**: A second-degree equation with one variable (i.e., quadratic equation) can have a form such as , where x is the variable, and a, b, and c are coefficients ().

You should handle the case where the values of the coefficients produce a double root & the case where they produce no root. You should also handle the case where the user input value 0 for a.

**Hint**:

Use the discriminant

# Introduction to Eclipse / Netbean

In the previous section, we have written our very first Java applications in a programming text editor such as Notepad. From this lab forward, we use an integrated development environment, so called IDE, which is like a text editor, but provides various features such as modifying, compiling, and debugging software. Some of the most popular IDEs for Java are JetBrains IntelliJ, NetBeans, and Eclipse. In this course, we use Eclipse for our demonstrations.

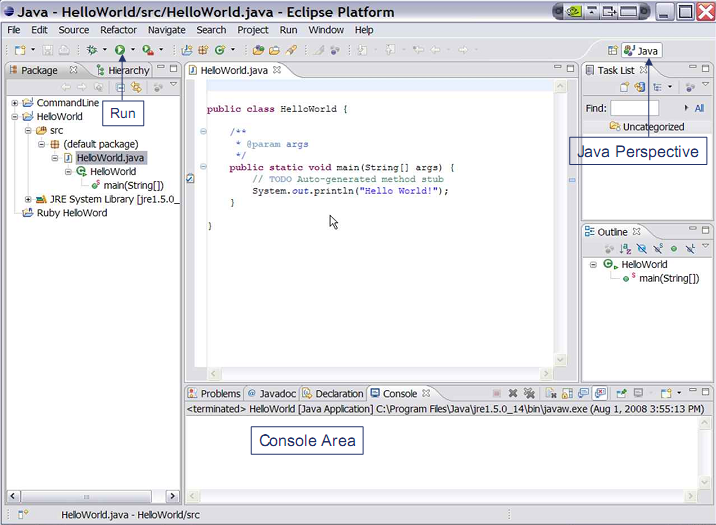
****

Figure . Eclipse IDE

***Installation guide:***

**Note:** You should install Java 8 or a later version before installing an IDE.

In this instruction guide, we need no installer; we just download the ZIP file and unzip them.

- Netbeans: Download the binary file at the following link. Read README.html for more details. The application is inside the ***bin* directory.**

<https://www.apache.org/dyn/closer.cgi/netbeans/netbeans/11.2/netbeans-11.2-bin.zip>

If you want to use pre-Apache Netbeans versions, you can see them [here](https://netbeans.org/downloads/8.2/) (this may not compatible with later Java version).

- Eclipse: We recommend ***Eclipse IDE for Enterprise Java Developers***. Download the suitable binary file at the following link. <https://www.eclipse.org/downloads/packages/>

# Javadocs help:

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

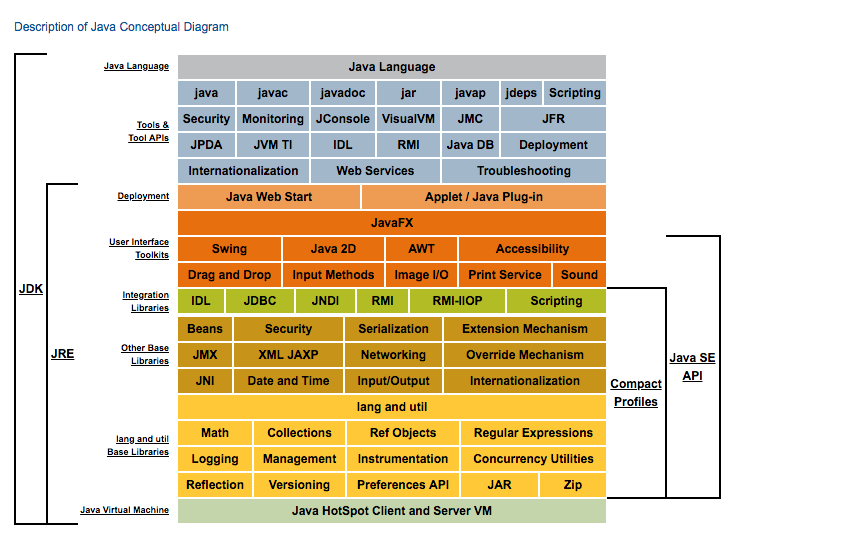


Figure . Java Conceptual Diagram

* Click the link Java SE API

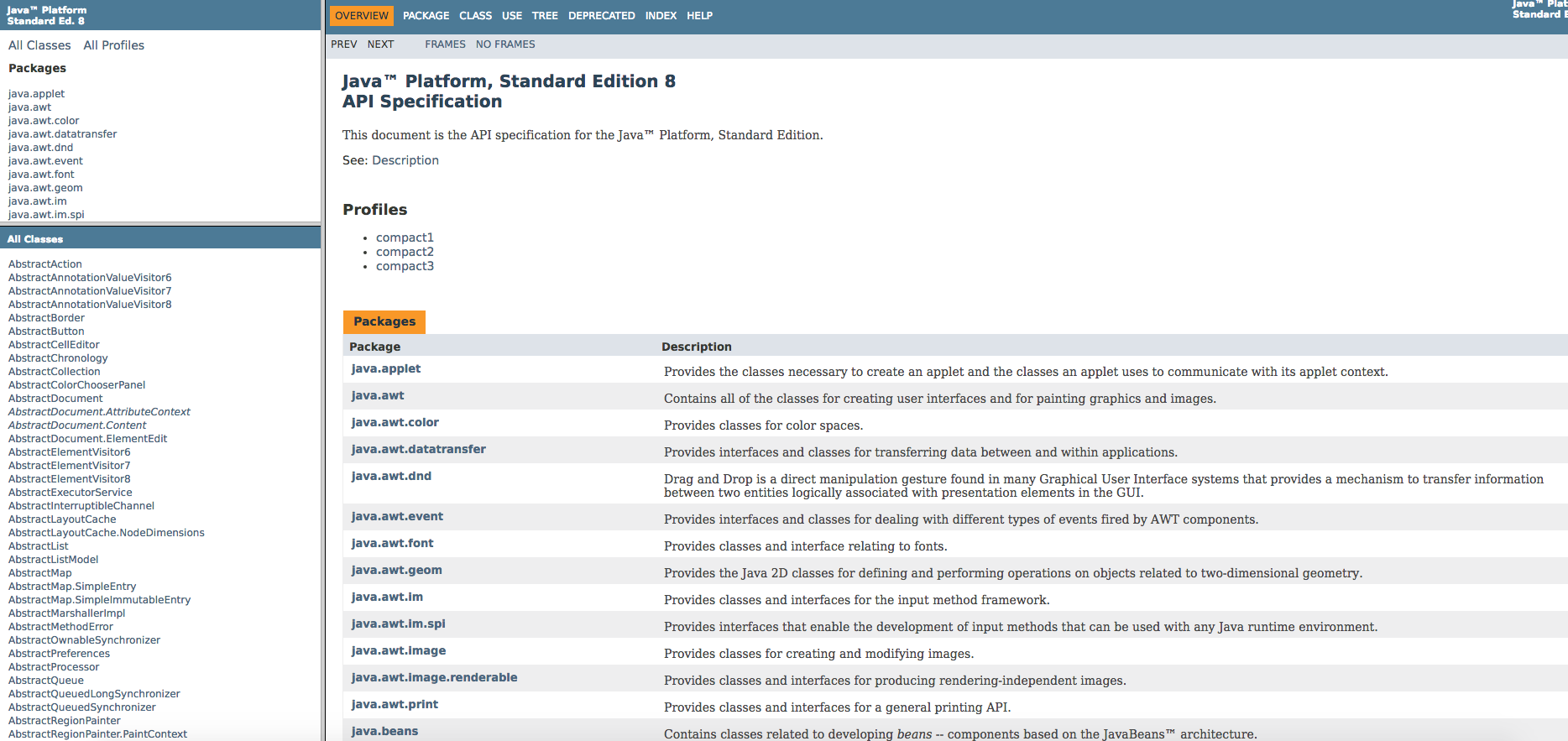


Figure . Java SE API

* 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)

# Your first Java project

1. From the Eclipse install directory, run Eclipse IDE.
2. In Eclipse IDE Launcher window, choose your workspace directory where you want to save the project(s). If you want to use the chosen directory as the default, check the box. Then, click *Launch* button.

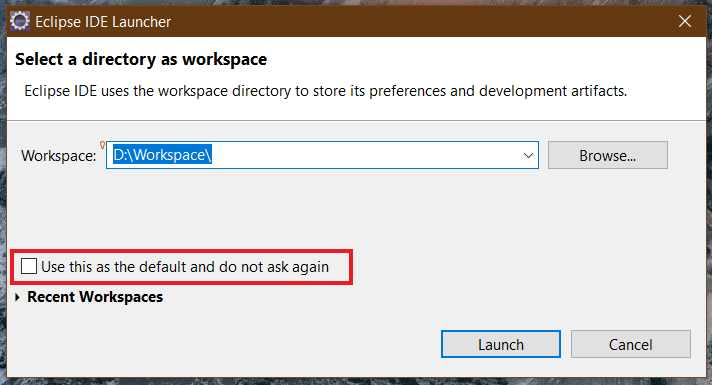
****

Figure . Eclipse Launcher Window

1. To create a new Java project, choose *File → New →Project…*

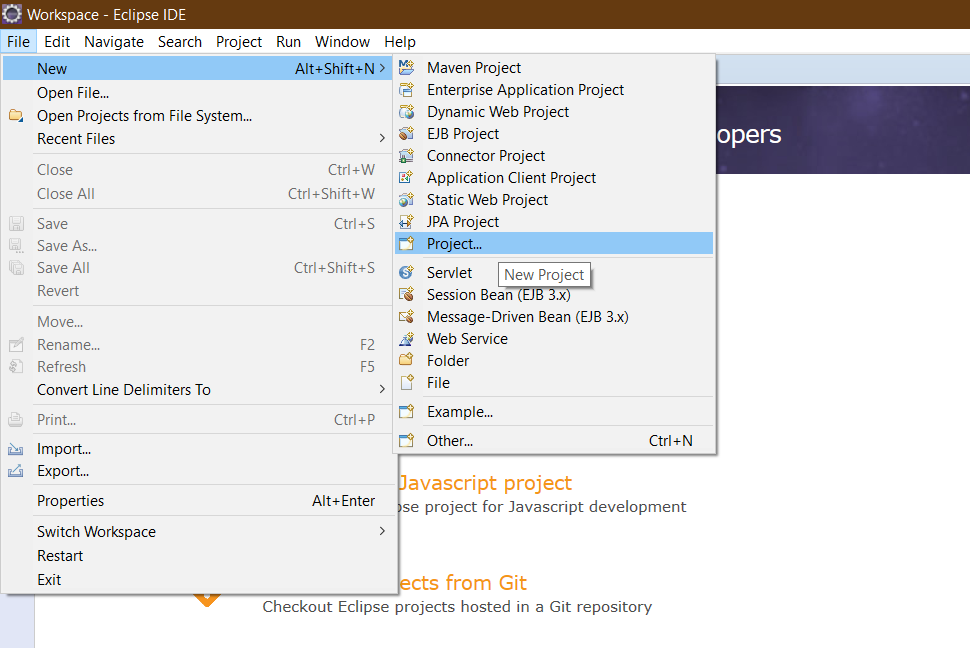


Figure . Create new Java project

1. On the pop-up window, choose *Java Project,* then click *Next >* button. If you cannot find it, type the filter text.

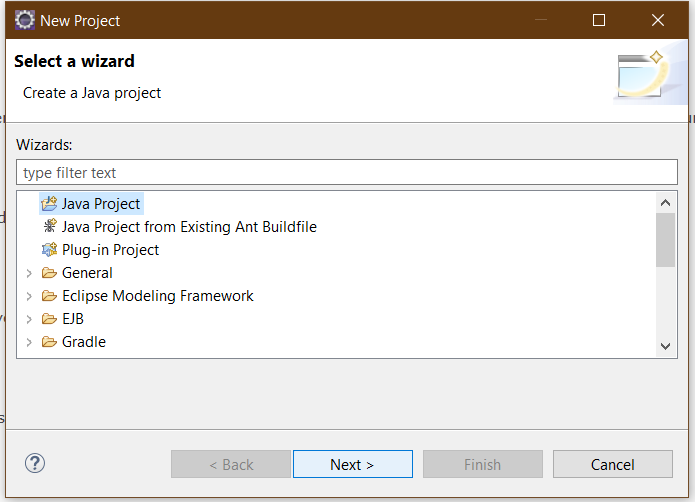


Figure . New Project Window

1. On the *New Java Project* window, let the *Project name* be “**JavaBasics**”. Then, click *Finish* button.

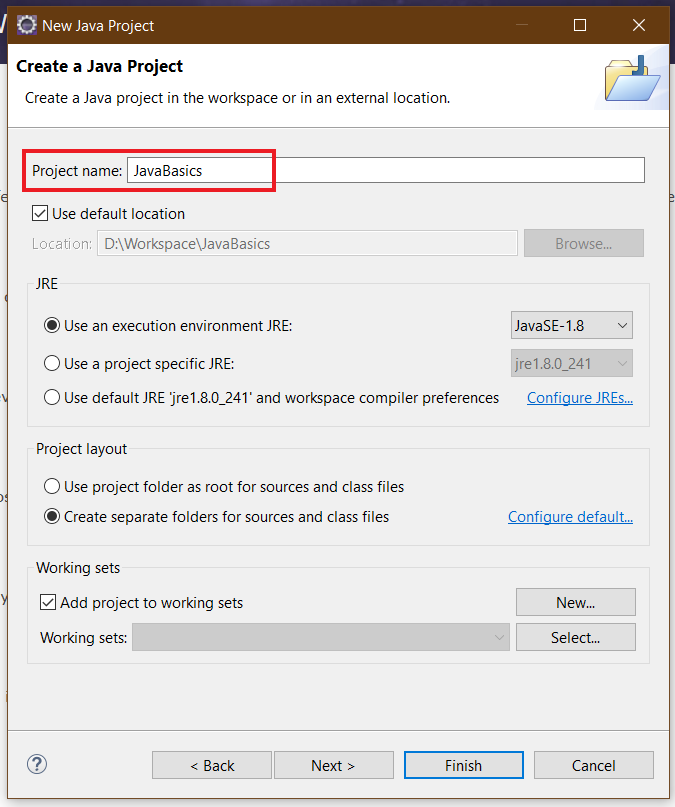


Figure . New Java Project Window

1. On the pop-up window, choose *Open Perspective*.

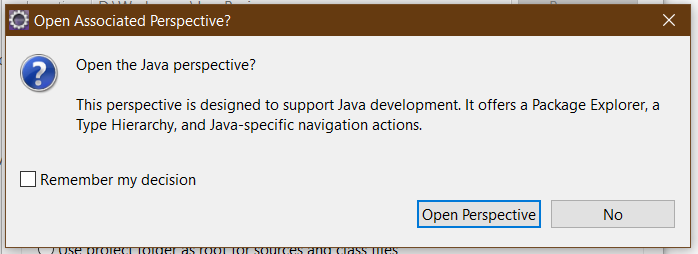


Figure . Open Associated Perspective Window

1. Close the Welcome page; then the Java perspective shows up.

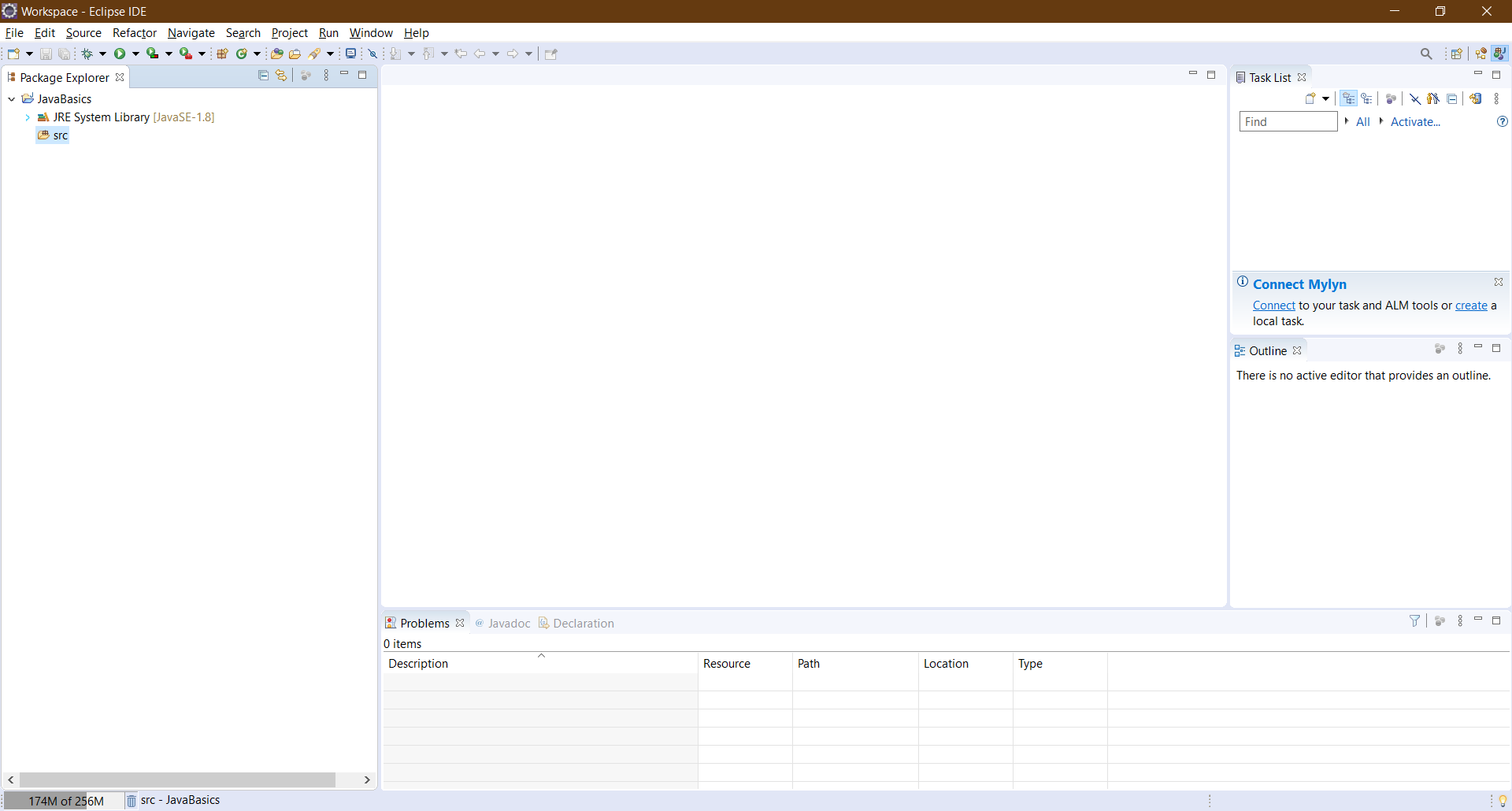


Figure . Java Perspective

# Exercises

## Write, compile and run the ChoosingOption program:

**Note:** We use JavaBasics project for this exercise.

**Step 1: Create a class.**

* Choose *File → New → Class*

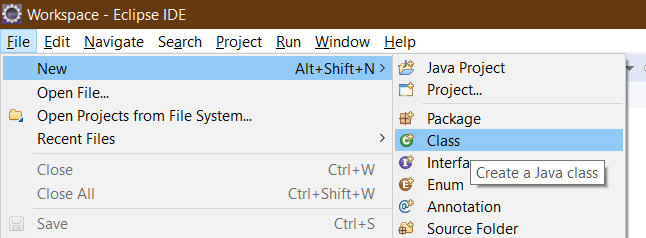
****

Figure . Class creating

* On the pop-up window, set the *Name* same as the class name in the Figure 19, which is “**ChoosingOption**”

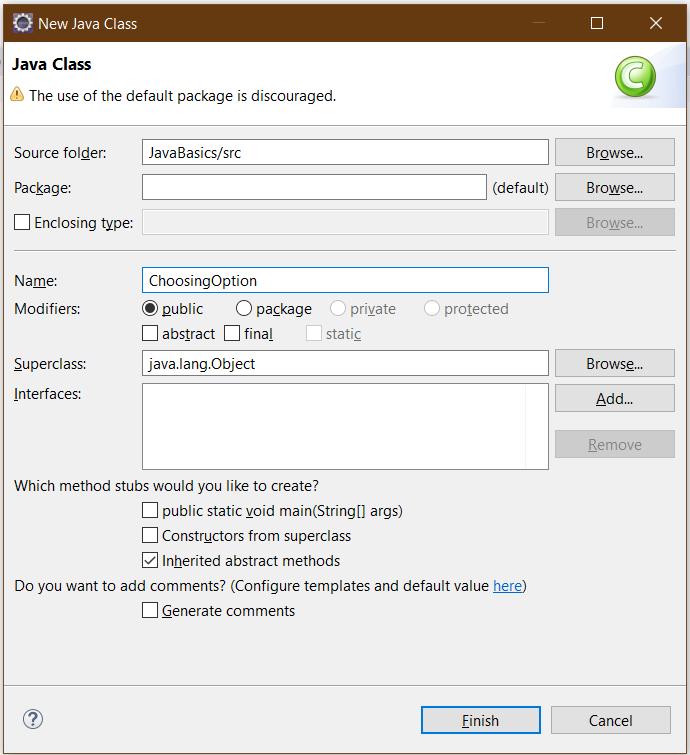
****

Figure . New Java Class Window

We have a new class namely *ChoosingOption* created as shown in Figure 20.

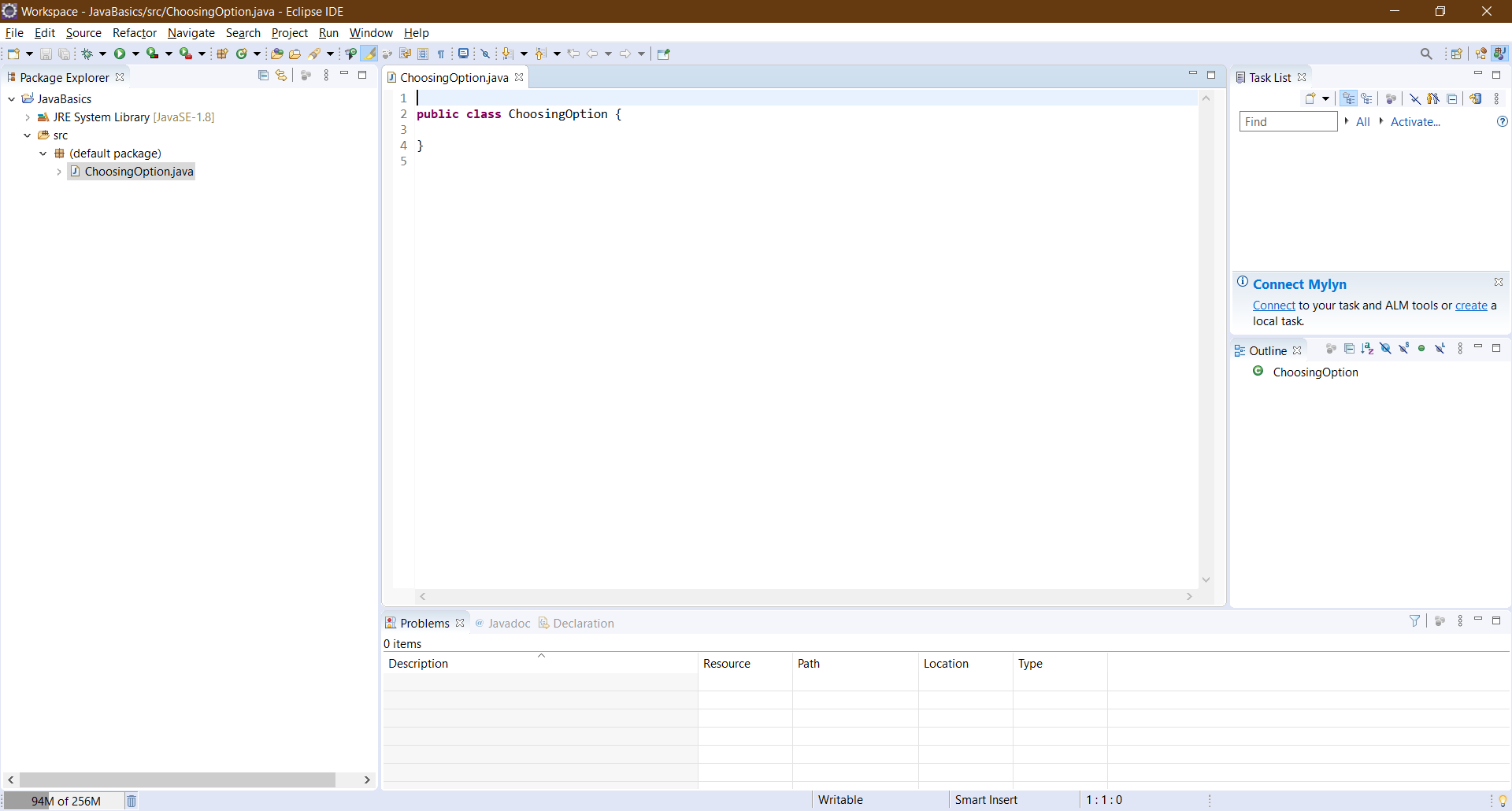


Figure . A New Class created

**Step 2: Write the program.** The source code is illustrated in Figure 21.

****

Figure . Choosing Option Application

**Step 3: Save and Launch.**

* Right-click on the *ChoosingOption* class *→Run As →Java Application*

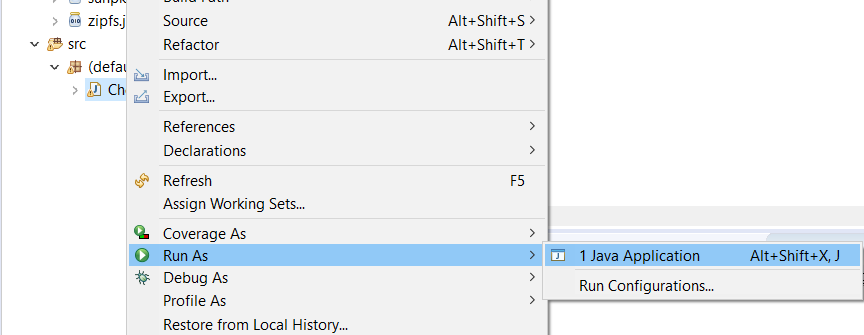


Figure . Run Application (1)

* Choose *Always save resources before launching*, then click *OK* button

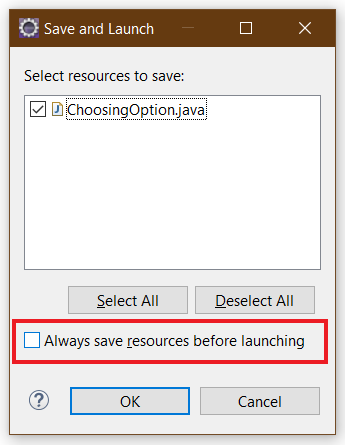


Figure . Save and Launch

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).

## Write a program for input/output from keyboard

**Note:** We use the JavaBasics project for this exercise.

**Step 1: Create a class.**

* Choose *File → New → Class*
* On the pop-up window, set the *Name* as “**InputFromKeyboard**”

**Step 2: Write the program.** The source code is illustrated in Figure 25.

**Step 3: Save and Launch.**

* Method 1: Right-click on the *InputFromKeyboard* class *→Run As →Java Application.*
* Method 2: Click the button and choose the application as shown in the Figure 24

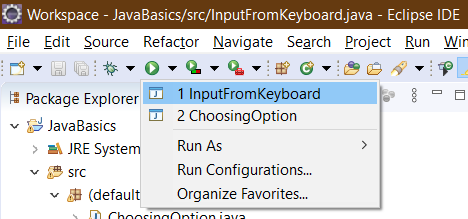
****

Figure . Run Application (2)



Figure . InputFromKeyboard Application

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

**E.g. n=5:**

\*

\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*

**Note**: You must create a new Java project for this exercise.

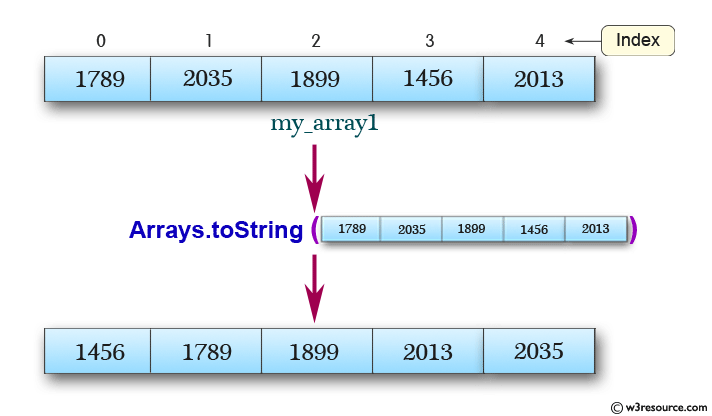
## **Write a program to display the number of days of a month**, which is entered by users (both month and year). If it is an invalid month/year, ask the user to enter again.

**Note**: You must create a new Java project for this exercise.

* The user can either enter a month in its full name, abbreviation, in 3 letters, or in number. To illustrate, the valid inputs of *January* are January, Jan., Jan, and 1.
* The user must enter a year in a non-negative number and enter all the digits. For instance, the valid inputs of year *1999* is only 1999, but not 99, “one thousand nine hundred ninety-nine”, or anything else.
* A year is either a common year of 365 days or a leap year of 366 days. Every year that is divisible by 4 is a leap year, except for years that are divisible by 100, but not by 400. For instance, year 1800 is not a leap year, yet year 2000 is a leap year. In a year, there are twelve months, which are listed in order as follows.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Month** | January | February | March | April | May | June | July | August | September | October | November | December |
| Abbreviation | Jan. | Feb. | Mar. | Apr. | May | June | July | Aug. | Sept. | Oct. | Nov. | Dec. |
| **In 3 letters** | Jan | Feb | Mar | Apr | May | Jun | Jul | Aug | Sep | Oct | Nov | Dec |
| In Number | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| **Days of Month in Common Year** | 31 | 28 | 31 | 30 | 31 | 30 | 31 | 31 | 30 | 31 | 30 | 31 |
| Days of Month in Leap Year | 31 | **29** | 31 | 30 | 31 | 30 | 31 | 31 | 30 | 31 | 30 | 31 |

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



**Note**: You must create a new Java project for this exercise.

- The array can be entered by the user or a constant.

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

**Note**: You must create a new Java project for this exercise.

- The matrices can be entered by the user or constants.

1. In various operating systems, the *cd <desired directory name>* command (*cd* stands for ***c****hange* ***d****irectory*) allows us to change the current working directory to the desired directory. Besides, in Windows 10, to access another drive, we type the drive's letter, followed by ":". For instance, to change the current working drive to drive D, we issue the command “*d:*” [↑](#footnote-ref-2)