Object-Oriented Programming

Lab 01: Environment Setup and Java Basics

In this lab, you will practice with:

- Set up development environment
- Writing some Java's code and compile it with Java JDK on the Eclipse IDE

Keywords: JDK, JRE, Java installation, programming text editor, IDE, git, github

0 Assignment Submission

For this lab class, you will have to turn in your work twice, specifically:

- **Right after the lab class**: for this deadline, you should include any work you have done within the lab class time to github.
- **10PM the day after the class**: for this deadline, you should include **all** the two programs (2.2.5 & 2.2.6) and six applications in the exercise section (6.1, 6.2, 6.3, 6.4, 6.5 & 6.6) of this lab, into a directory namely "**Lab01**" and push it to your **master** branch of the valid repository.

Each student is expected to turn in his or her own work and not give or receive unpermitted aid. Otherwise, we would apply extreme methods for measurement to prevent cheating. Please write down answers for all questions into a text file named "**answers.txt"** and submit it within your repository.

1 Getting Started

1.1 Java Development Kit

Java Platform, Standard Edition Development Kit (JDK) is a development environment for building applications, applets, and components using the Java programming language. There are many releases of the platform (the latest being JDK 15) that are available to download. Among them, JDK 8 is the most widely-used version of Java and it is also the last long-term support (LTS) release that contains JavaFX (which we will be working extensively with later on in this course). For the above reasons**, it is required that JDK 8 is installed for all the labs in this course**. However, if you have installed a later version, you can still install JavaFX separately (there will be an installation guide in the JavaFX lab in this case).

The installation steps for JDK 8 are illustrated as follows:

Step 1: Check if JDK has been pre-installed

1. Open a Command Prompt (on Windows. Press Windows+R to open the "Run" box. Type "cmd" and then click "OK" to open a regular Command Prompt) or a Terminal (on Linux or macOS).
2. Issue the following command.
   ```
   $ javac -version
   ```
3. In case a JDK version number is returned (e.g., JDK x.x.x), then JDK has already been installed. When the JDK version is prior to 1.8, a message "*Command 'javac' not found*", or a message "*'javac' is not recognized as an internal or external command, operable program or batch file.*", proceed to **step 2** to install Oracle JDK 8. Otherwise, proceed to 1.2.

Note: Linux usually chooses OpenJDK as its default JDK since OpenJDK is open source. However, Oracle JDK is not completely compatible with OpenJDK and it is recommended to use Oracle JDK.

Step 2: Download Oracle JDK 8

1. Go to Java SE Development Kit (JDK) 8 download site at the following link. https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html
2. Download the installation file, choose the appropriate one for your operating system, under "*Java SE Development Kit 8u241*" section. The recommended file for Linux is Compressed Archive file. We may need an Oracle Account to download for the Oracle JDK License has changed for releases since April 16, 2019.

Step 3: Install and Configure

- **Windows:**
  1. Install Oracle JDK 8. Run the downloaded installer and follow the instructions.
  2. Configure. Launch Control Panel → System and Security → System → Advanced system settings → Environment Variables in Advanced tab. In the lower list "System variables", you need to look for two variables:
     - The first one is `JAVA_HOME`, check if it already exists, if not, add new by choosing "New…" and set the variable name as `JAVA_HOME`. Then, you need to set its value as the path to where the JDK installation is located, which is the "jdk-x" folder under the "Java" folder, e.g. `C:\Program Files\Java\jdk1.8.0_241`.
     - The second variable is `Path`, you need to modify it by adding the following entry to it: `%JAVA_HOME%\bin`

- **Linux:**
  1. Create installation directory. We shall install Oracle JDK 8 under "*/usr/local/java*" directory.

```
$ cd /usr/local
$ sudo mkdir java
```

  2. Extract the downloaded package (e.g., jdk-8u241-linux-x64.tar.gz) to the installation directory.

```
$ cd /usr/local/java
$ sudo tar xzvf ~/Downloads/jdk-8u241-linux-x64.tar.gz
// x: extract, z: for unzipping gz, v: verbose, f: filename
```

  3. Inform the Linux to use this JDK/JRE

```
// Setup the location of java, javac and javaws
$ sudo update-alternatives --install "/usr/bin/java" "java"
"/usr/local/java/jdk1.8.0_241/bin/java" 1
        // --install symlink name path priority
$ sudo update-alternatives --install "/usr/bin/javac" "javac"
"/usr/local/java/jdk1.8.0_241/bin/javac" 1
$ sudo update-alternatives --install "/usr/bin/javaws" "javaws"
"/usr/local/java/jdk1.8.0_241/bin/javaws" 1


// Use this Oracle JDK/JRE as the default
$ sudo update-alternatives --set java /usr/local/java/jdk1.8.0_241/bin/java
        // --set name path
$ sudo update-alternatives --set javac /usr/local/java/jdk1.8.0_241/bin/javac
$ sudo update-alternatives --set javaws /usr/local/java/jdk1.8.0_241/bin/javaws
```

- **MacOS:** Double-click the DMG file and follow the instructions.

Step 4: Verify the JDK Installation. Issue the following command.

```
$ javac -version
```

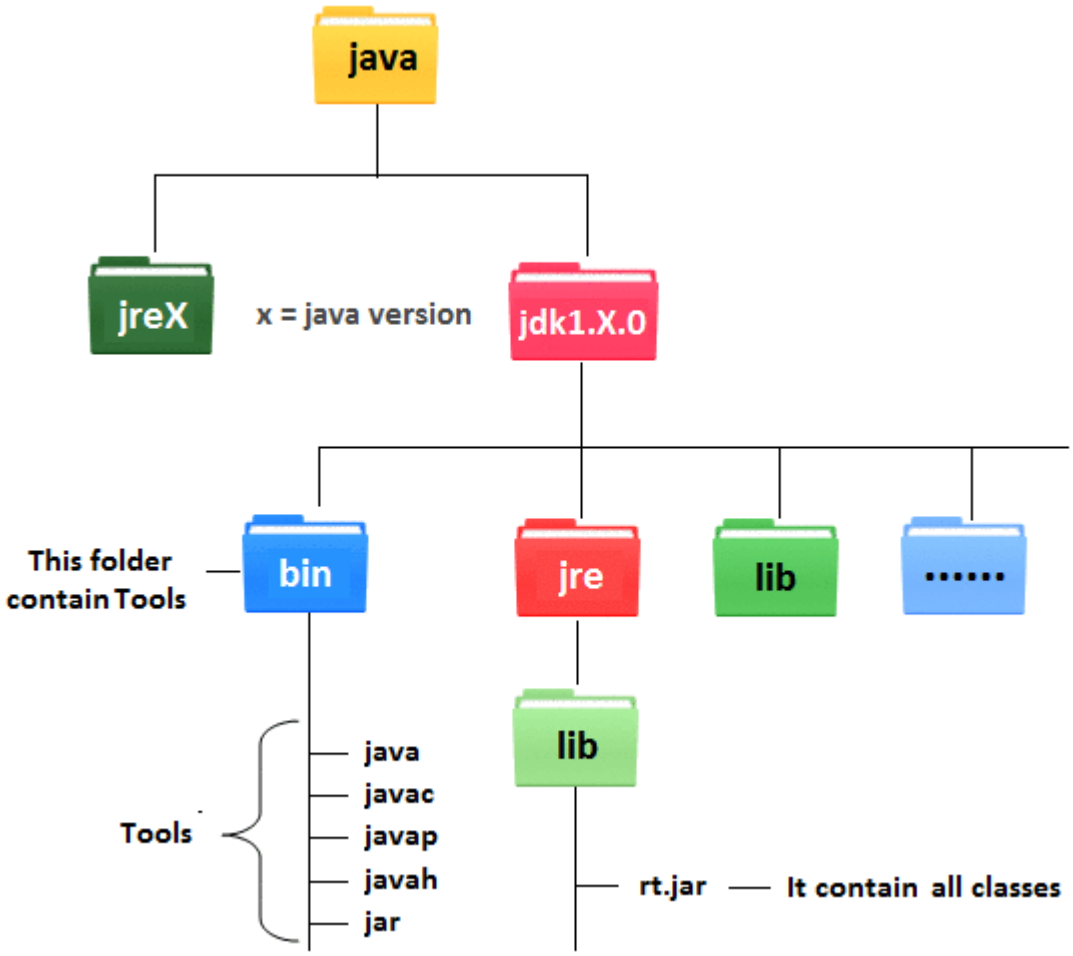


Figure 1. Development Environment.

1.2 Programming Text Editor

For the next section (section 2), you can use any text editor to write your java source code. Here, the Notepad text editor is used as an illustration.

2 First Programs

2.1 Java Programming Steps

The steps in writing a Java program are illustrated in the following steps and in Figure 2.

Step 1: Write the source code such as the code shown in Figure 3. and save in, e.g., "HelloWorld.java" file.

Step 2: Compile the source code into Java portable bytecode (or machine code) using the JDK's Java compiler by issuing the following command.

```
$ javac HelloWorld.java
```

Step 3: Run the compiled bytecode using the JDK's Java Runtime by issuing the following command.

```
$ java HelloWorld
```

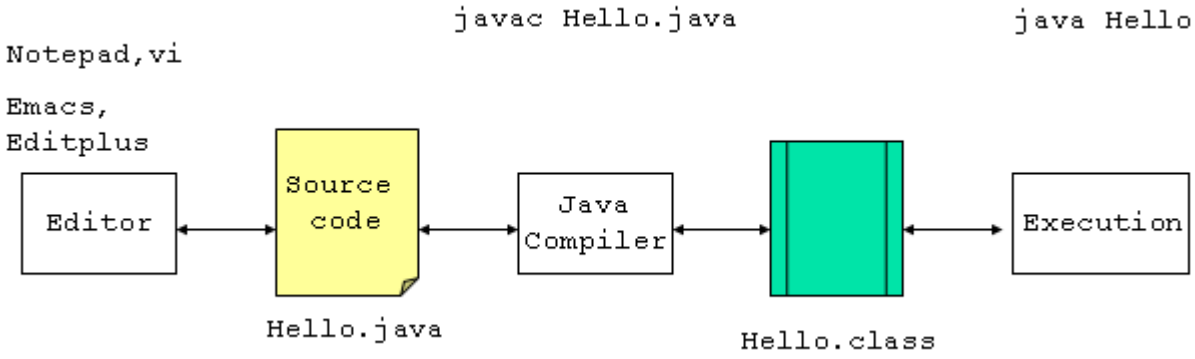


Figure 2. Compile a Java application by command line

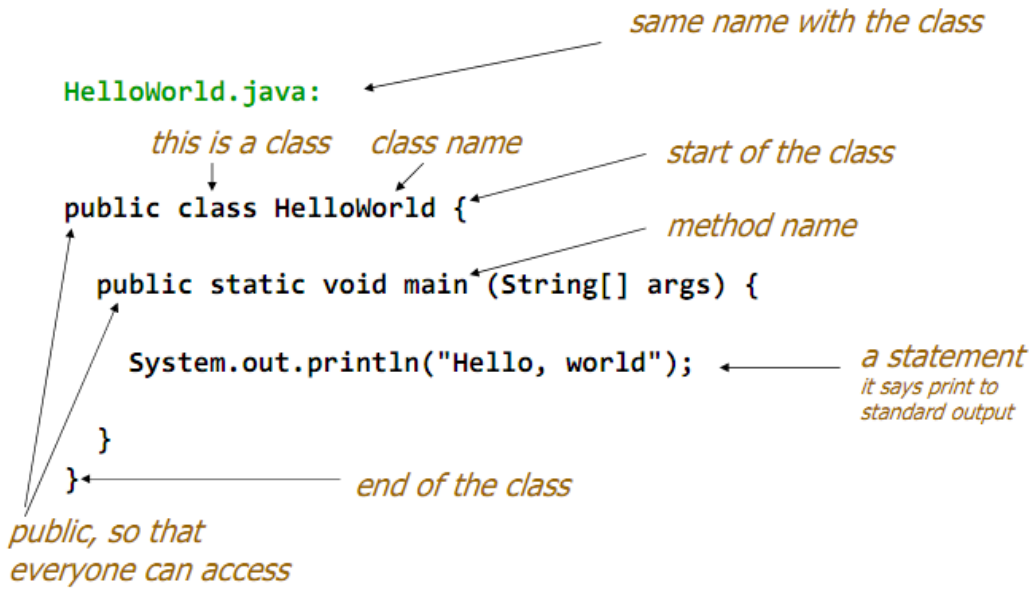


Figure 3. The first Java application

The result is shown in Figure 4.

```
%> javac HelloWorld.java
%> java HelloWorld
Hello, world
```

Figure 4. Result of the first Java application

For better illustration, we can watch the following demo videos.
https://www.youtube.com/watch?v=G1ubVOl9IBw
https://www.youtube.com/watch?v=2Xa3Y4xz8_s

2.2 The Very First Java Programs

2.2.1 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.

```
1  //Example 1: HelloWorld.java
2  //Text-printing program
3  public class HelloWorld {
4
5     public static void main(String args[]){
6           System.out.println("Xin chao \n cac ban!");
7           System.out.println("Hello \t world!");
8
9     } // end of method main
10 }
```

Figure 5. The First Java Application

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

```
$ javac HelloWorld.java
$ java HelloWorld
```

- **Step 5:** After completing this assignment, you should add and commit all of your work to the master branch. Follow these two steps to achieve the desired result:
  - **Step 1:** Add all changes in the current directory and its subdirectories to the staging area
  - (master) $ git add .
  - **Step 2:** Commit the change in the local repository.
  - (master) $ git commit –m "Complete assignment 2.2.1"

-

2.2.2 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.

[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:*"

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

```
1  // Example 2: FirstDialog.java
2  import javax.swing.JOptionPane;
3  public class FirstDialog{
4      public static void main(String[] args){
5          JOptionPane.showMessageDialog(null,"Hello world! How are you?");
6          System.exit(0);
7      }
8  }
```

Figure 6. 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
```

- **Step 5:** As step 5 in section 2.2.1, add and commit your work using '**git add**' and '**git commit -m <message>**'.

2.2.3 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

```
1  // Example 3: HelloNameDialog.java
2  import javax.swing.JOptionPane;
3  public class HelloNameDialog{
4    public static void main(String[] args){
5      String result;
6      result = JOptionPane.showInputDialog("Please enter your name:");
7      JOptionPane.showMessageDialog(null, "Hi "+ result + "!");
8      System.exit(0);
9    }
10 }
```

Figure 7. 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
```

- **Step 5:** As step 5 in section 2.2.1, add and commit your work using '**git add**' and '**git commit -m <message>**'.

2.2.4 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

```
1  // Example 5: ShowTwoNumbers.java
2  import javax.swing.JOptionPane;
3  public class ShowTwoNumbers {
4    public static void main(String[] args){
5      String strNum1, strNum2;
6      String strNotification = "You've just entered: ";
7
8      strNum1 = JOptionPane.showInputDialog(null,
9                  "Please input the first number: ","Input the first number",
10                 JOptionPane.INFORMATION_MESSAGE);
11     strNotification += strNum1 + " and ";
12
13     strNum2 = JOptionPane.showInputDialog(null,
14                 "Please input the second number: ","Input the second number",
15                 JOptionPane.INFORMATION_MESSAGE);
16     strNotification += strNum2;
17
18     JOptionPane.showMessageDialog(null,strNotification,
19                 "Show two numbers", JOptionPane.INFORMATION_MESSAGE);
20     System.exit(0);
21   }
22 }
```

Figure 8. 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
```

- **Step 5:** As step 5 in section 2.2.1, add and commit your work using '**git add**' and '**git commit -m <message>**'.

2.2.5 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
- Don't forget to add and commit your work using '**git add** .' and '**git commit -m <message>**' command

2.2.6 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 $ax + b = 0$ $(a \neq 0)$.

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

Don't forget to add and commit your work using '**git add** .' and '**git commit -m <message>**' command

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

Note: A system of first-degree equations with two variables x_1 and x_2 can be written as follows.

$$\{a_{11}x_1 + a_{12}x_2 = b_1 \; a_{21}x_1 + a_{22}x_2 = b_2$$

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:

$$D = |a_{11}\ a_{12}\ a_{21}\ a_{22}\ | = a_{11}a_{22} - a_{21}a_{12} D_1 = |b_1\ a_{12}\ b_2\ a_{22}\ | = b_1a_{22} - b_2a_{12} D_2 = |a_{11}\ b_1\ a_{21}\ b_2\ | = a_{11}b_2 - a_{21}b_1$$

- **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 $ax^2 + bx + c = 0$, where x is the variable, and a, b, and c are coefficients $(a \neq 0)$.

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 $\Delta = b^2 - 4ac$

After completing the code in section 2, you should push all of your changes, including assignment 2.2.1 to 2.2.6 to the **master** branch of the valid repository you have created.

Hint: You should use "git push origin <name of the branch that you want to push to the remote repository>", in this case is **"master"** , to push all of your works to the **master** branch.

3 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.

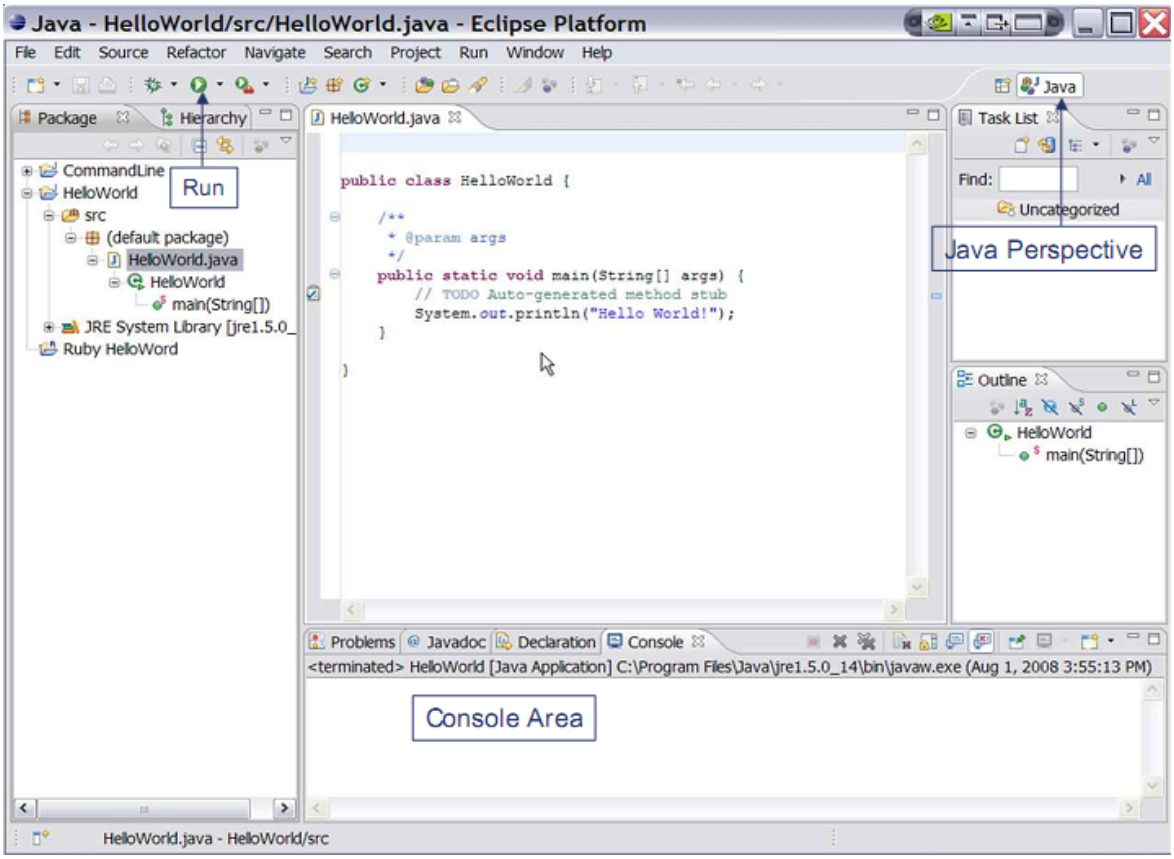


Figure 9. Eclipse IDE

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 (this may not be compatible with later Java versions).

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

4 Javadocs help:

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

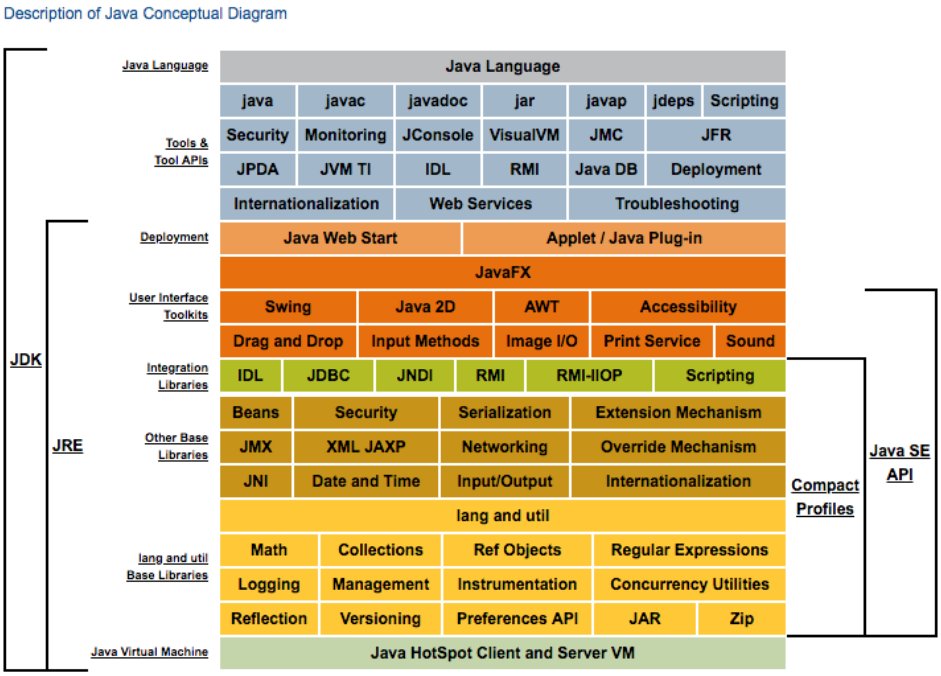


Figure 10. Java Conceptual Diagram

- Click the link Java SE API

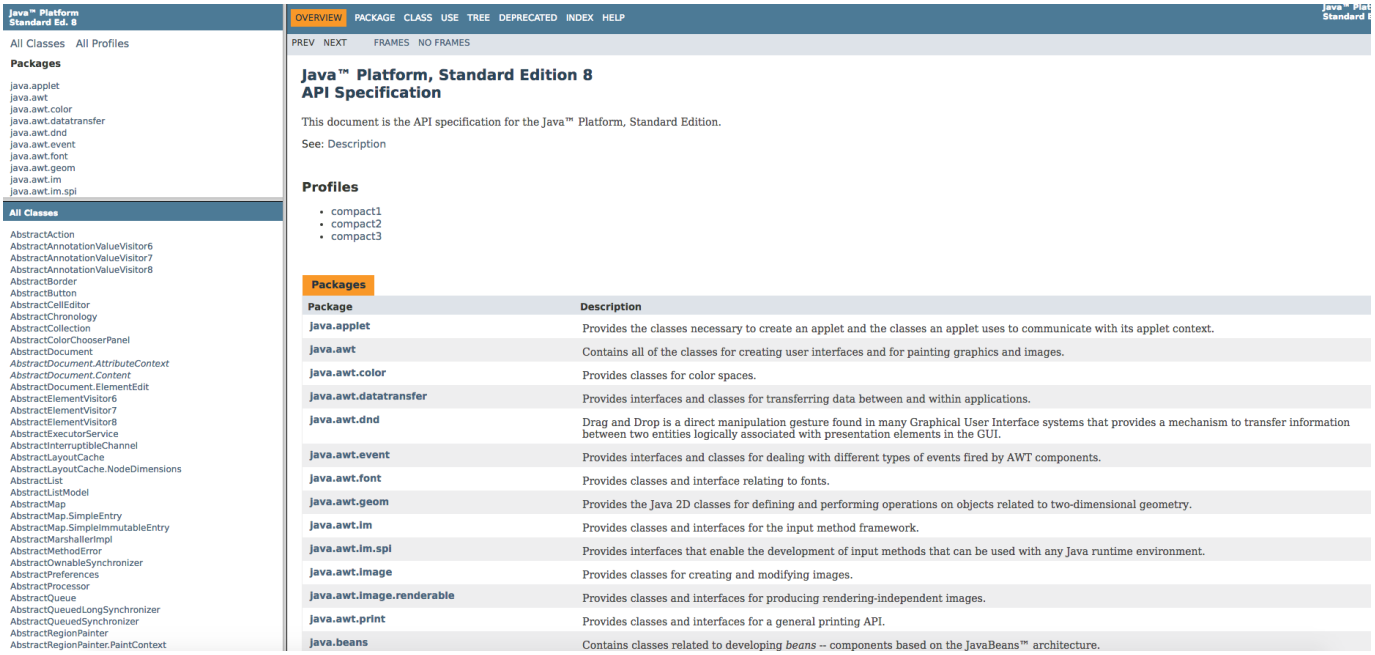


Figure 11. Java SE API

- 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)

5 Your first Java project

1. From the Eclipse install directory, run Eclipse IDE.
2. In the 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 the Launch button.

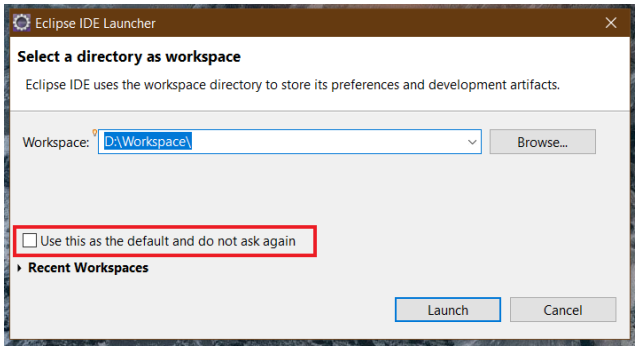


Figure 12. Eclipse Launcher Window

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

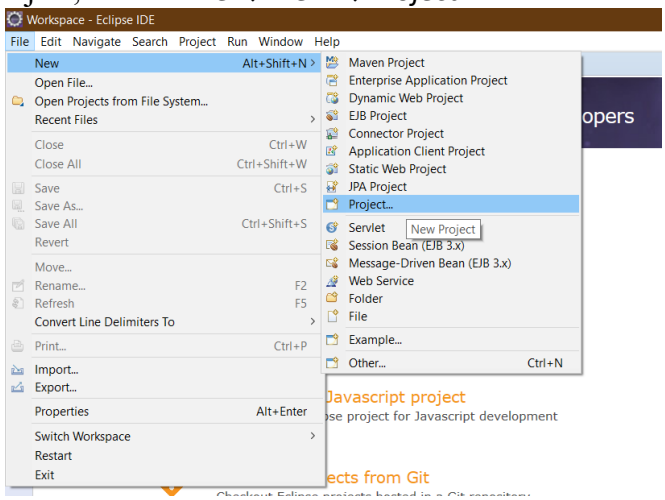


Figure 13. Create new Java project

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

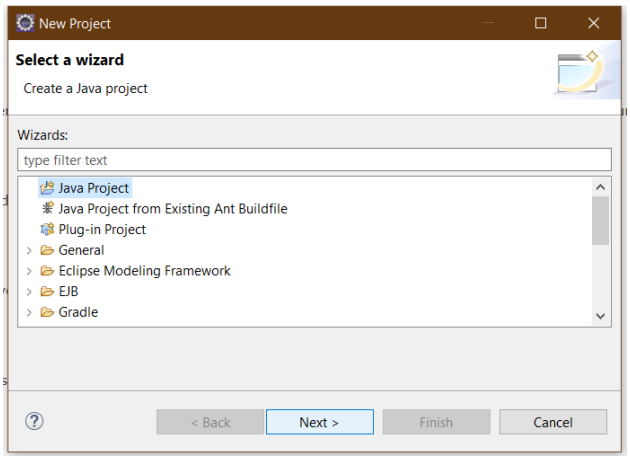


Figure 14. New Project Window

5. On the *New Java Project* window, let the *Project name* be "**JavaBasics**". Then, click the Finish button.

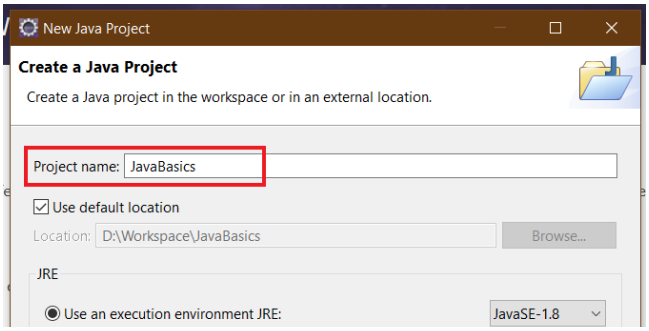


Figure 15. New Java Project Window

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

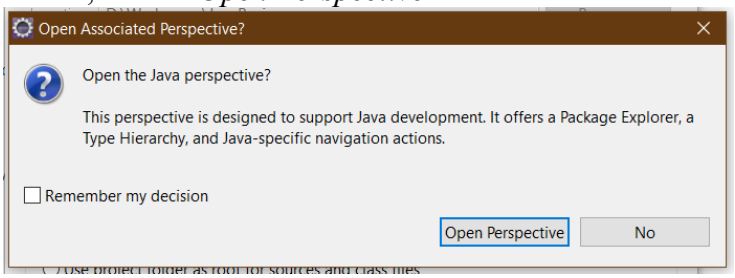


Figure 16. Open Associated Perspective Window

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

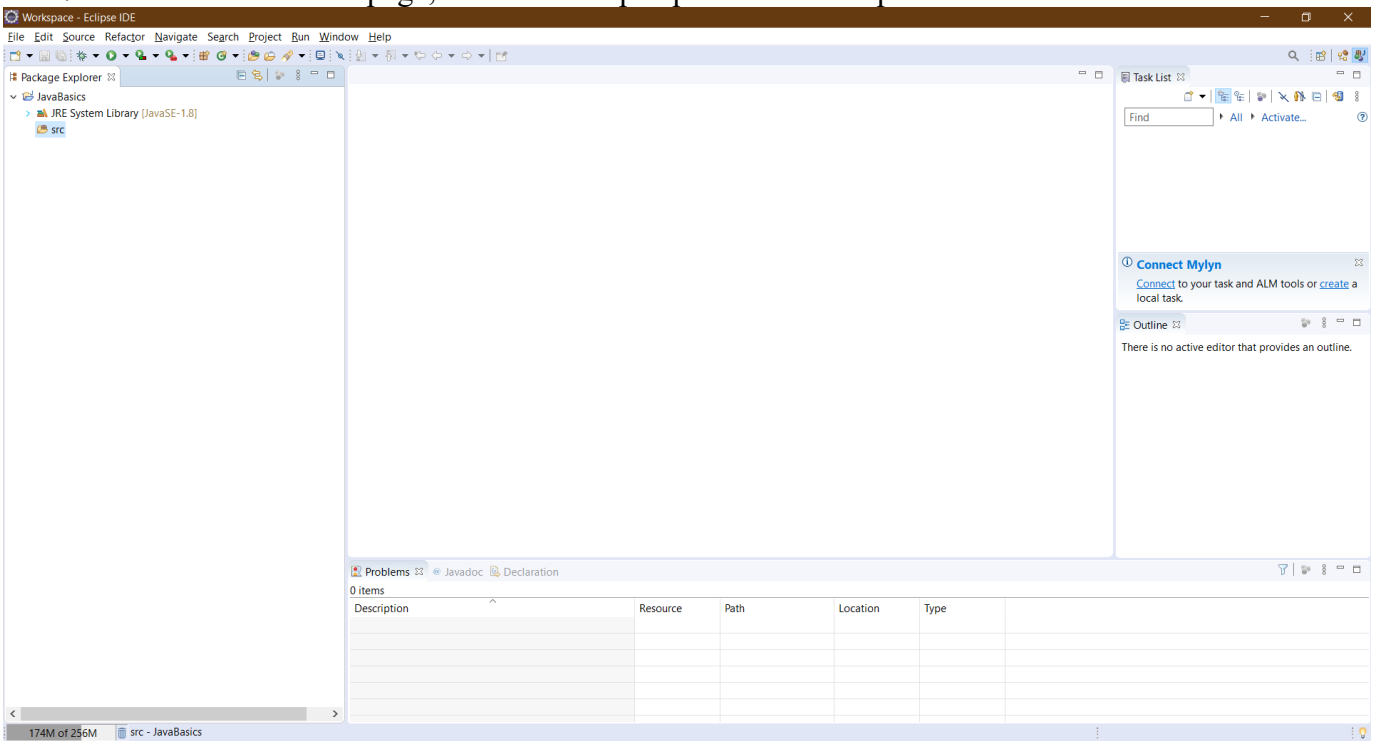


Figure 17. Java Perspective

6 Exercises

6.1 Write, compile and run the ChoosingOption program:

Note: We use the JavaBasics project for this exercise.

Step 1: Create a class.

- Choose *File → New → Class*

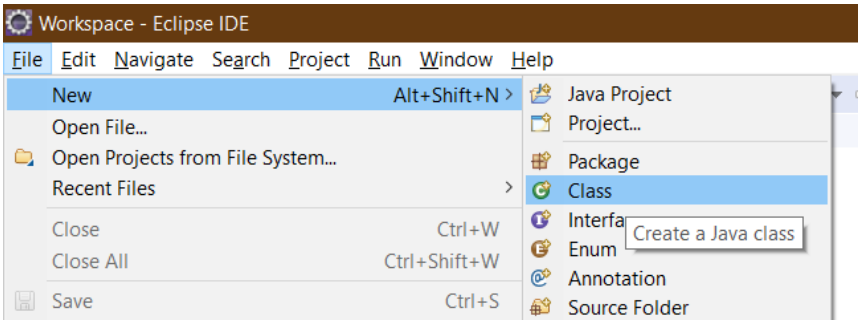


Figure 25. Class creating

- On the pop-up window, set the *Name* same as the class name in the Figure 26, which is "**ChoosingOption**"

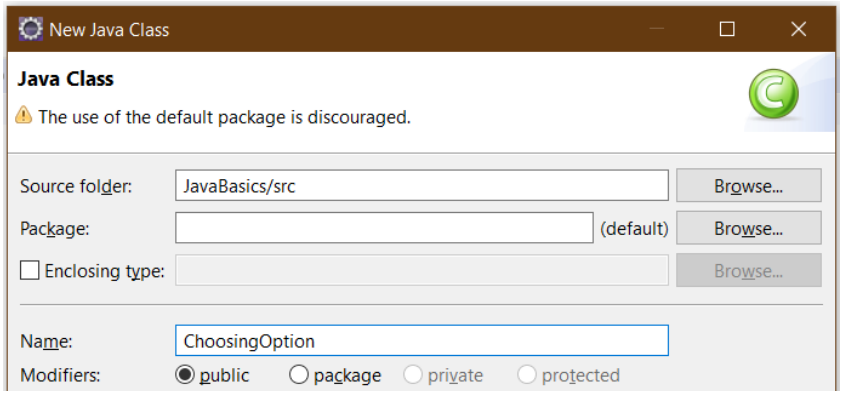


Figure 26. New Java Class Window

We have a new class namely *ChoosingOption* created as shown in Figure 27.

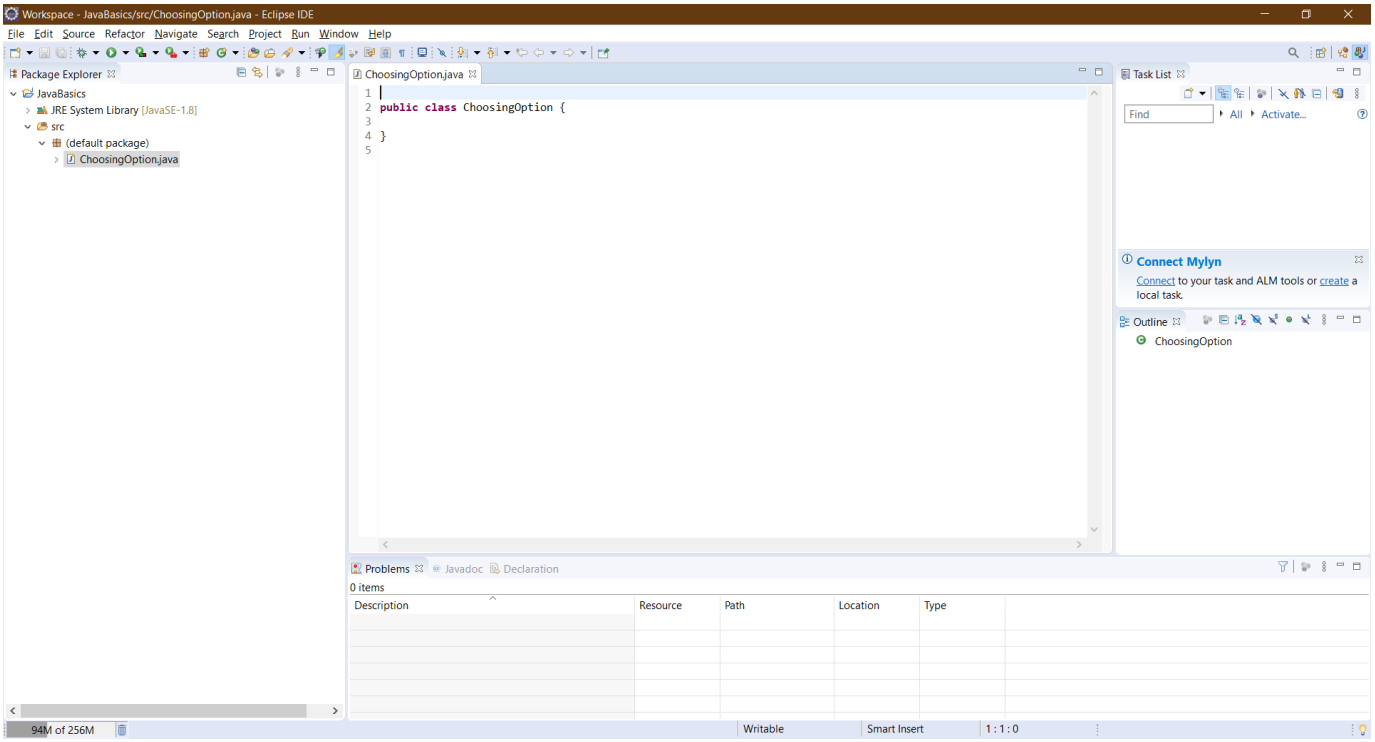


Figure 27. A New Class created

Step 2: Write the program. The source code is illustrated in Figure 28.

```
1  import javax.swing.JOptionPane;
2  public class ChoosingOption{
3    public static void main(String[] args){
4      int option = JOptionPane.showConfirmDialog(null,
5                   "Do you want to change to the first class ticket?");
6
7      JOptionPane.showMessageDialog(null,"You've chosen: "
8                        + (option==JOptionPane.YES_OPTION?"Yes":"No"));
9      System.exit(0);
10   }
11 }
```

Figure 28. Choosing Option Application

Step 3: Save and Launch.

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

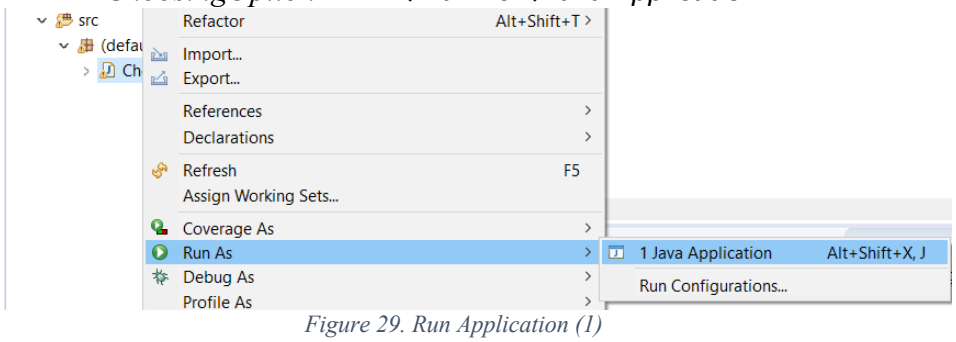


Figure 29. Run Application (1)

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

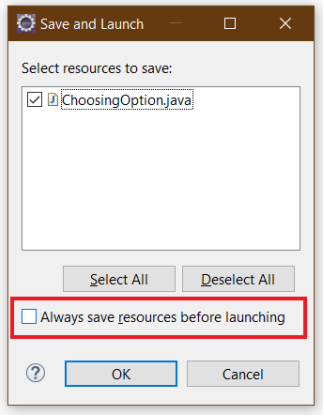


Figure 30. Save and Launch

Step 4: Add, commit, and push all your work to the "**'master'** branch.

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).

6.2 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 32.

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 31

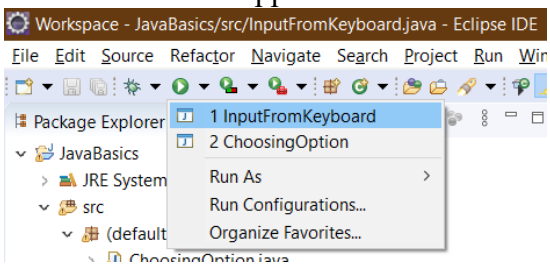


Figure 31. Run Application (2)

```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 + ".");

    }
}
```

Markers | Properties | Servers | Data Source Explorer | Snippets | Problems | Console | Searc

```
<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.
```

Figure 32. InputFromKeyboard Application

6.3 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.

6.4 ***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.*

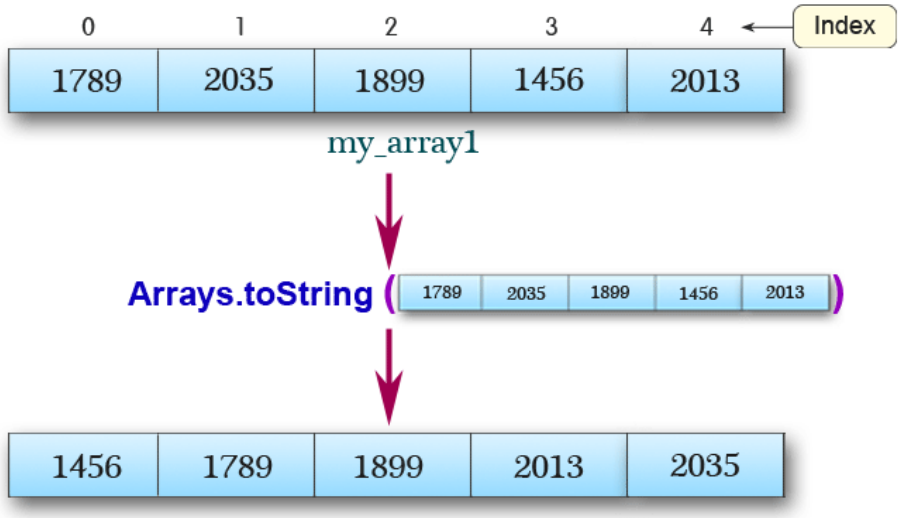
Note: You must create a new Java project for this exercise.

- 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 input 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, the year 1800 is not a leap year, yet the 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

6.5 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.

6.6 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.

After completing all of your tasks, You should push all of your work you've done in this lab into a "**master**" branch.

7 References

Hock-Chuan, C. (2020, January). *How to Install JDK 13 (on Windows, macOS & Ubuntu) and Get Started with Java Programming*. Retrieved from Nanyang Technological University: https://www3.ntu.edu.sg/home/ehchua/programming/howto/JDK_HowTo.html