

Start the lab

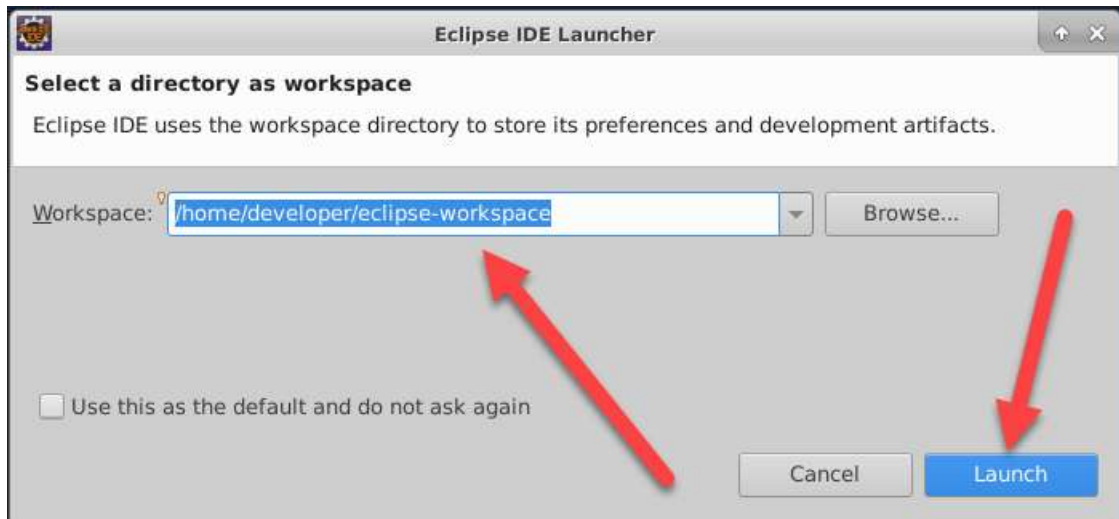
Lab2: Using the Scanner class for Console applications



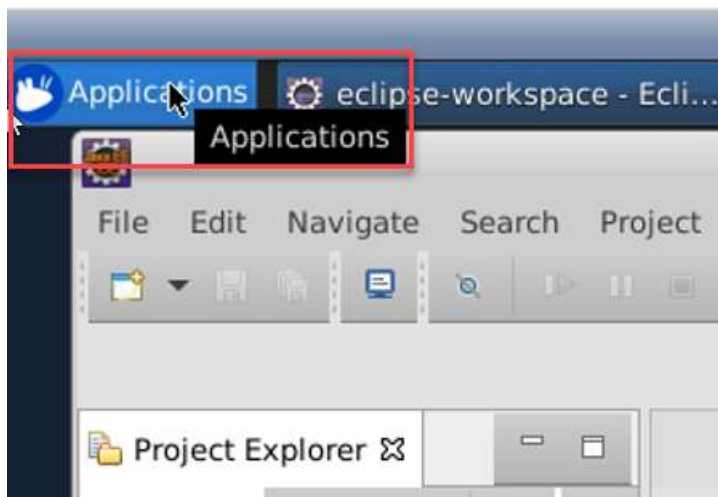
Let Eclipse start



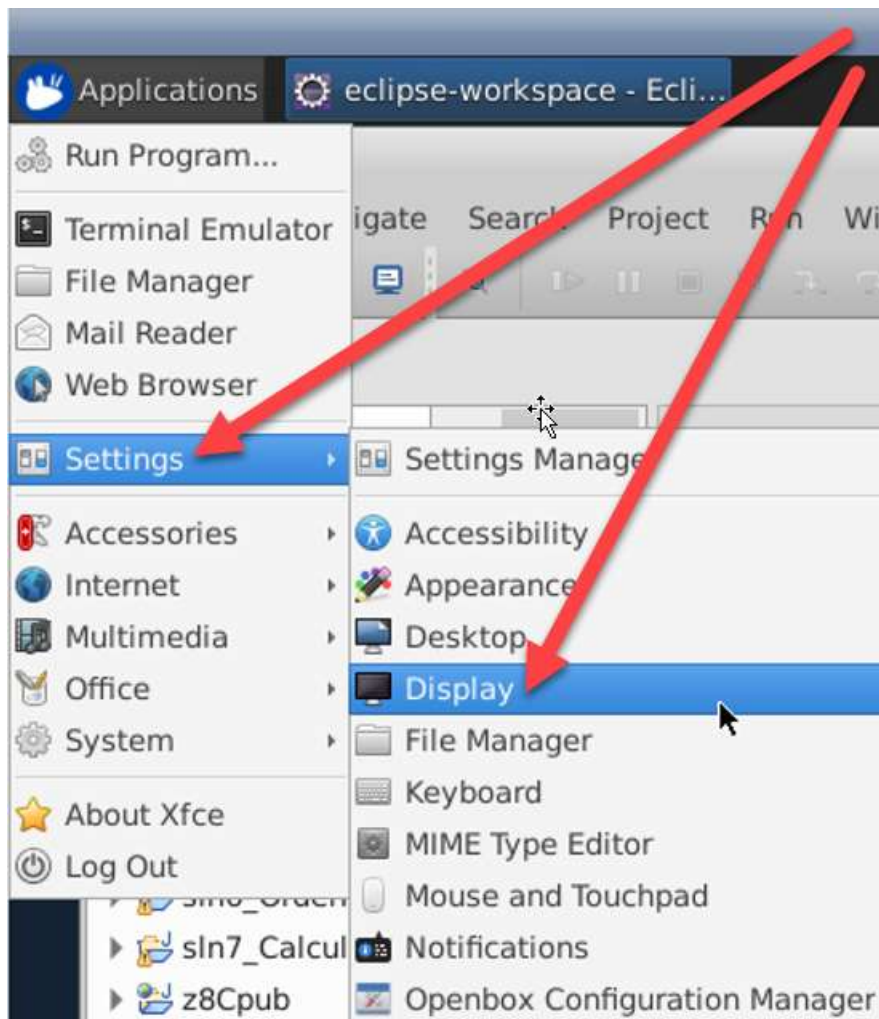
Leave the workspace unchanged, and click **Launch**.



In the upper left hand corner, click the **Applications** button.

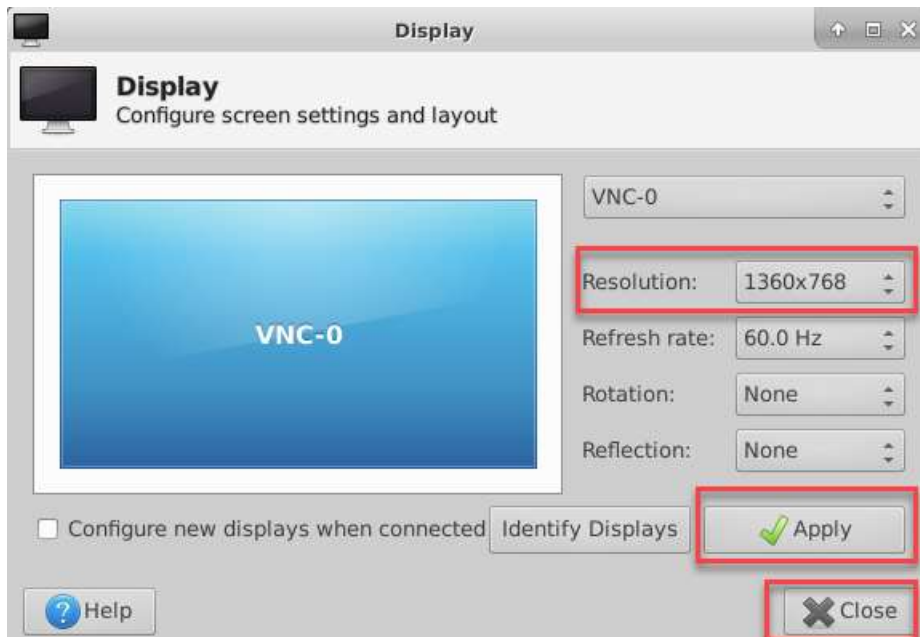


Click **Settings** -> **Display**

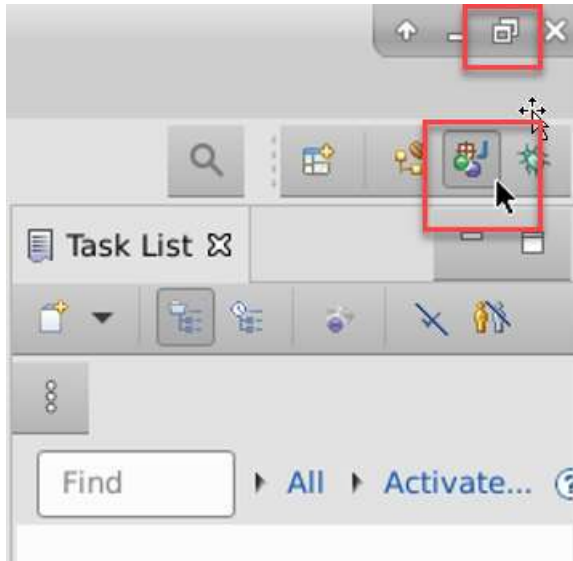


IMPORTANT: This example uses resolution **1360x768**. However, a different setting may work better for you.

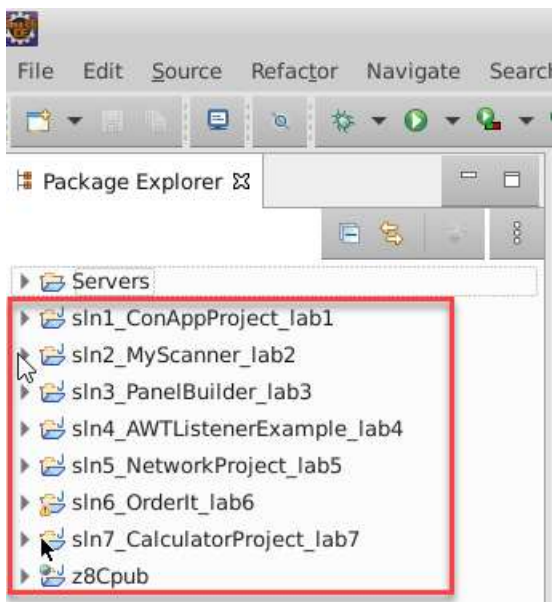
Click **Apply** and **Close**.



In the upper right corner, maximize **Eclipse**. Also, important, click the **Java Perspective** button. It must be the **Java perspective**.



Solution projects are listed in the Project Explorer. Key in all lines of code. However, if needed, use the solutions for troubleshooting.

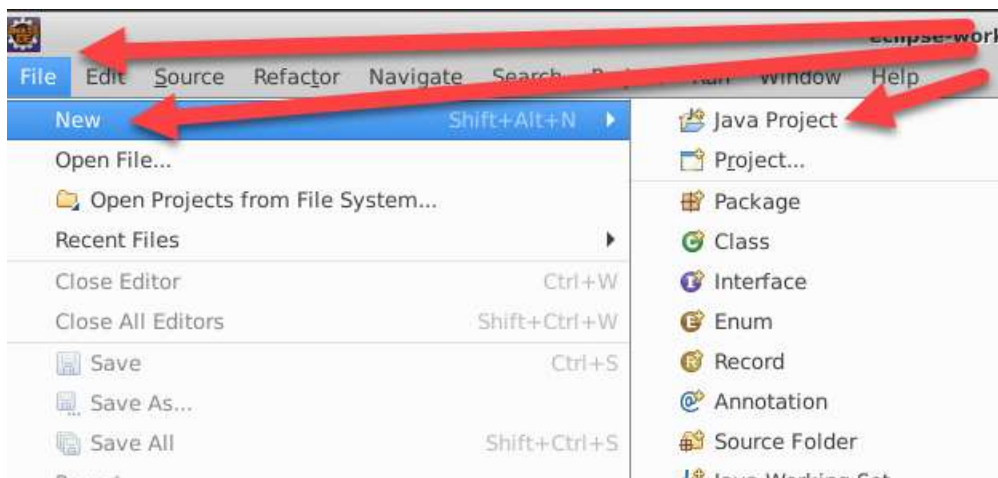


For this first lab, use solution project **sln2_MyScanner_lab2** for reference. It is possible to copy and paste the Java code. However, avoid copy and paste as much as possible.



Create a new Java Project called **MyScanner**

Click the **File** menu -> **New** -> **Java Project**



Name the project **MyScanner** and click **Finish**.

New Java Project

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

Project name:

☒ Use default location
Location: [Browse...](#)

JRE—
☒ Use an execution environment JRE:
☐ Use a project specific JRE:
☐ Use default JRE 'java-8-openjdk-amd64' and workspace compiler preferences [Configure JREs...](#)

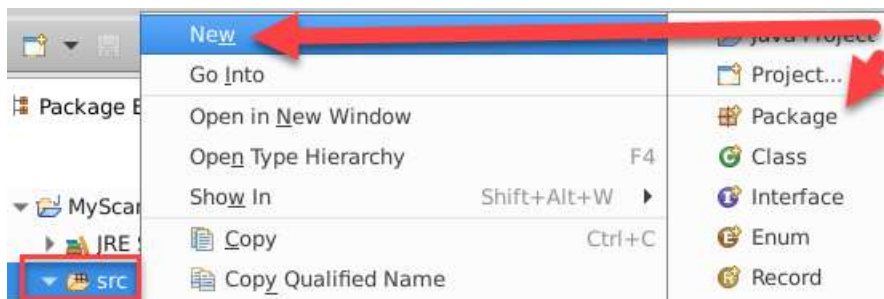
Project layout—
☐ Use project folder as root for sources and class files
☒ Create separate folders for sources and class files [Configure default...](#)

Working sets—
☐ Add project to working sets [New...](#)
Working sets: [Select...](#)

[?](#) [< Back](#) [Next >](#) [Cancel](#) [Finish](#)

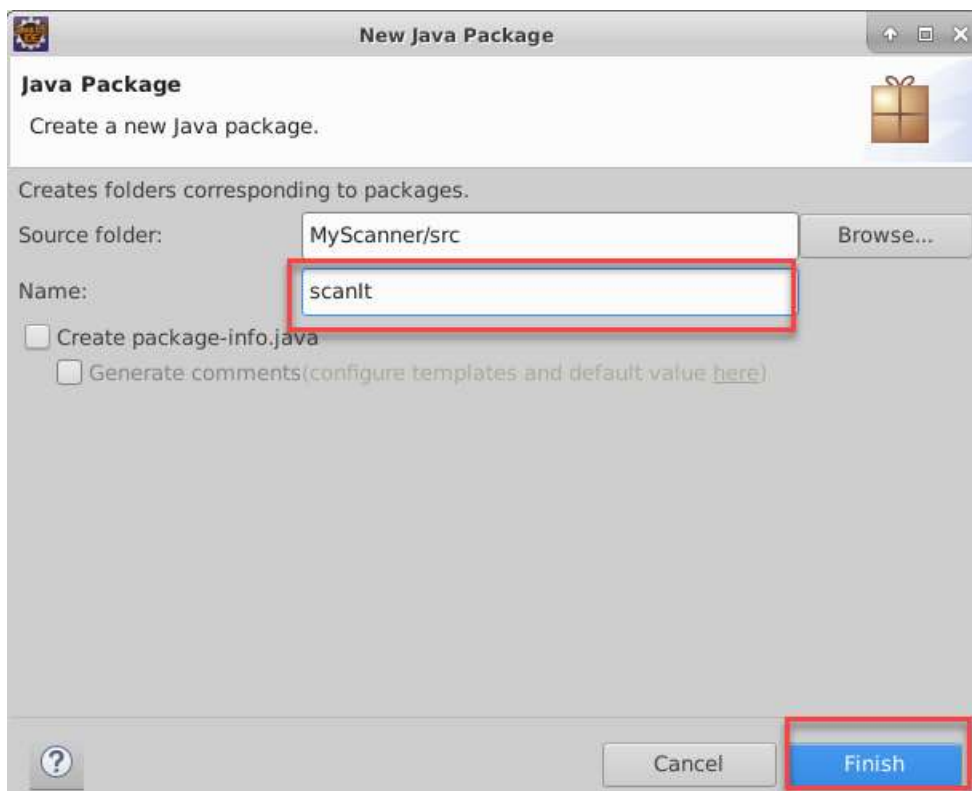
Create a package called addit

Expand the **MyScanner** and right click on the **src** folder, select **New -> Package**



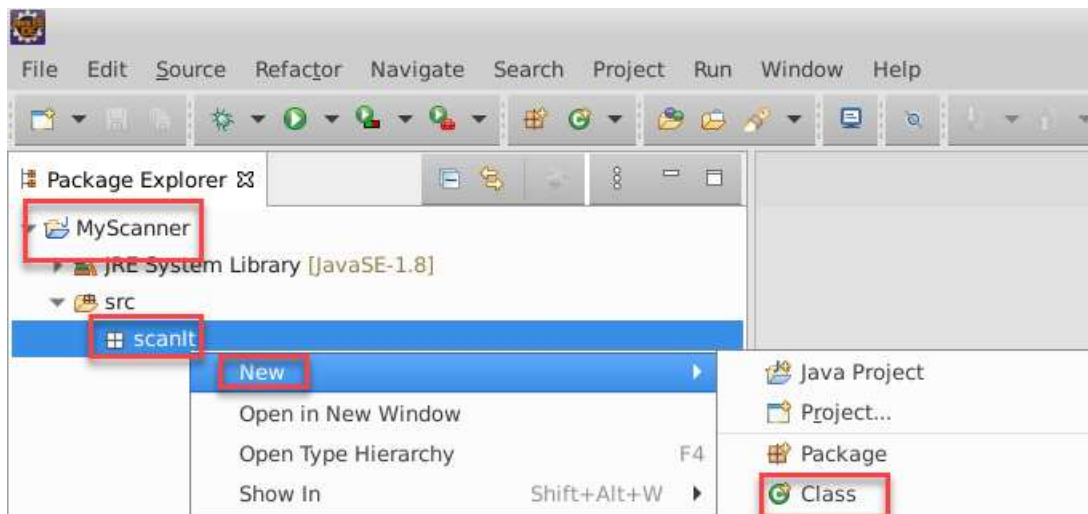
When the Java Package dialog opens, enter:

Name: **scanIt**



Click **Finish**

Right click on the **scanIt** package and select, New -> Class



In the Java Class dialog, enter the following:

Name: **ScannerConsole**

Check: **public static void main(String[] args)**

Click: **Finish**

Java Class
Create a new Java class.

Source folder: MyScanner/src Browse...

Package: scanIt Browse...

☐ Enclosing type: Browse...

Name: **ScannerConsole**

Modifiers: ☒ public ☐ package ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass: java.lang.Object Browse...

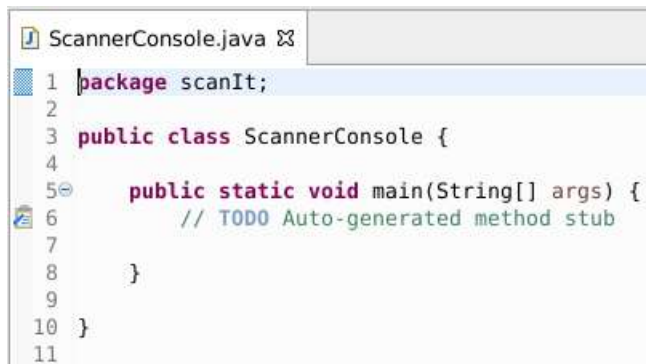
Interfaces: Add... Remove

Which method stubs would you like to create?
☒ public static void main(String[] args)
☐ Constructors from superclass
☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))
☐ Generate comments

Cancel **Finish**

Eclipse will open the **ScannerConsole.java** class file.



The screenshot shows the Eclipse IDE with the file `ScannerConsole.java` open. The code is as follows:

```
1 package scanIt;
2
3 public class ScannerConsole {
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7     }
8 }
9
10
11
```

Add the library that support command line input. It is just 1 library.

import java.util.Scanner;

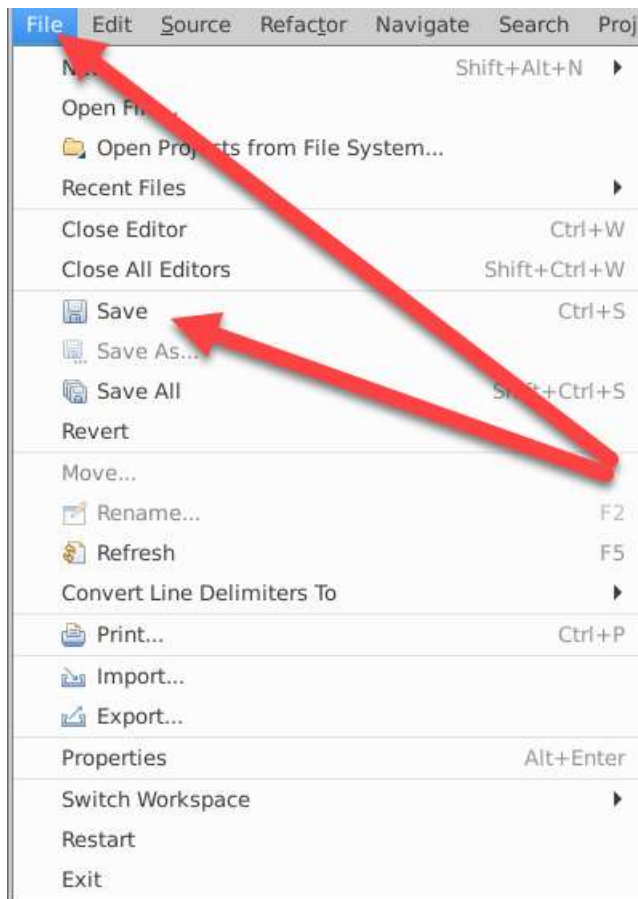
These libraries support command line input, and converting the input to data.



The screenshot shows the Eclipse IDE with the file `ScannerConsole.java` open. The code is now updated with the import statement. A red arrow points to the line `import java.util.Scanner;` on line 3.

```
1 package scanIt;
2
3 import java.util.Scanner;
4
5 public class ScannerConsole {
6
7     public static void main(String[] args) {
8         // TODO Auto-generated method stub
9     }
10 }
11
12 }
```

Click the **File** menu -> **Save**



It is time to instantiate class. We do so in the main method.

In the main method add the code below

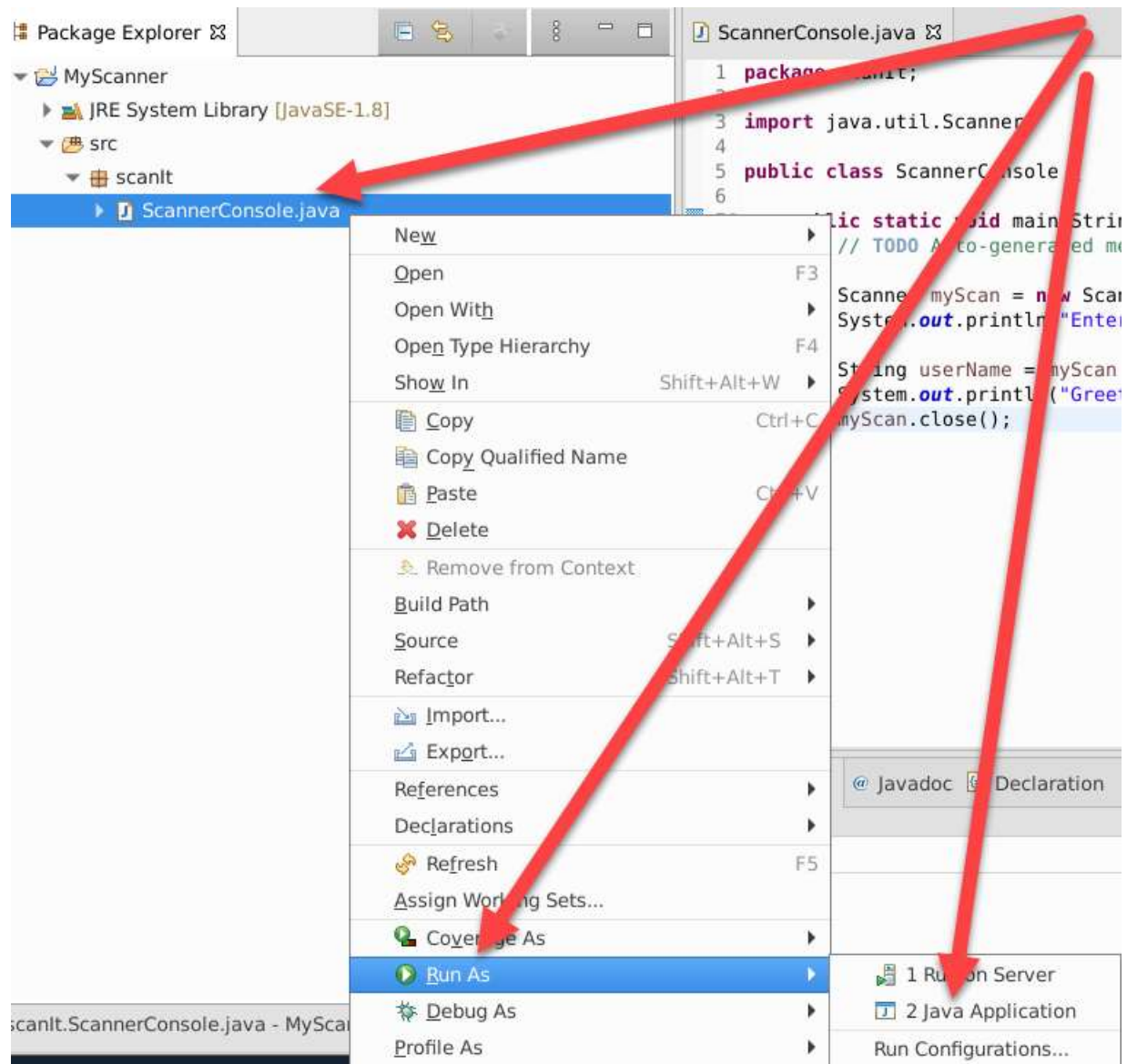
Notice there is no **throws IOException**

```
ScannerConsole.java
1 package scanIt;
2
3 import java.util.Scanner;
4
5 public class ScannerConsole {
6
7     public static void main(String[] args) {
8         // TODO Auto-generated method stub
9
10        Scanner myScan = new Scanner(System.in); //Instantiate myScan as a Scanner object
11        System.out.println("Enter username");    // Instruction to the user output to the console
12
13        String userName = myScan.nextLine();      //take command line input and store it in a String
14        System.out.println("Greetings " + userName); //output the results to the console.
15        myScan.close();                          //Delete the myScan object to free the memory.
16
17    }
18
19 }
20
```

Click **File** menu -> **Save**

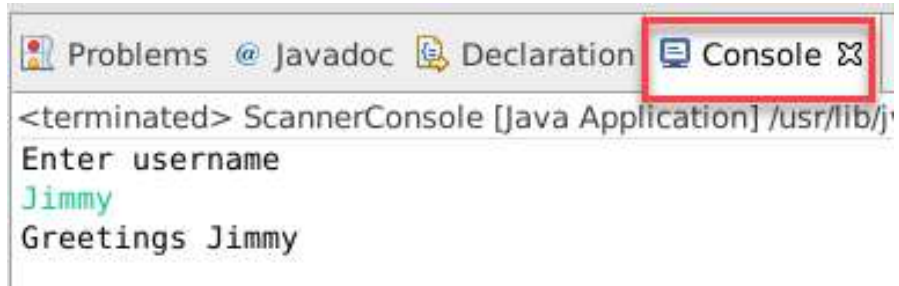
Run the application

In the Package Explorer view, right click the ScannerConsole.java file, click **Run As -> Java Application**



Click in the **Console** view, to focus it.

Enter a name from the keyboard



```
<terminated> ScannerConsole [Java Application] /usr/lib/j
Enter username
Jimmy
Greetings Jimmy
```

The Scanner methods below are available for different data. Therefore, there is less of a need to convert data.

nextInt(), returns int data.

nextDouble(), returns double, or real number data.

Update the application by adding all the lines shown below.

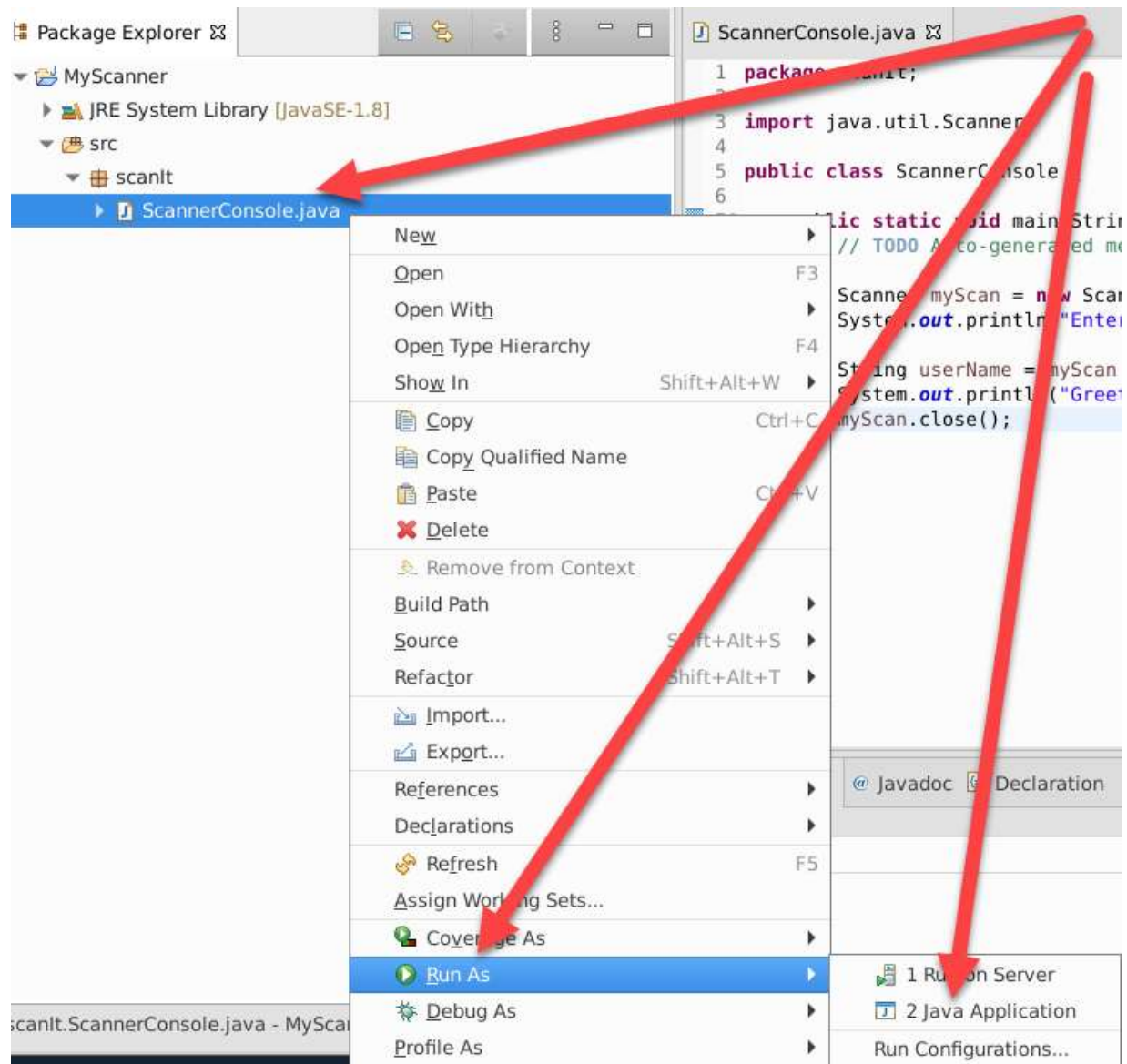
Add the getTotal method shown below. It returns double data.

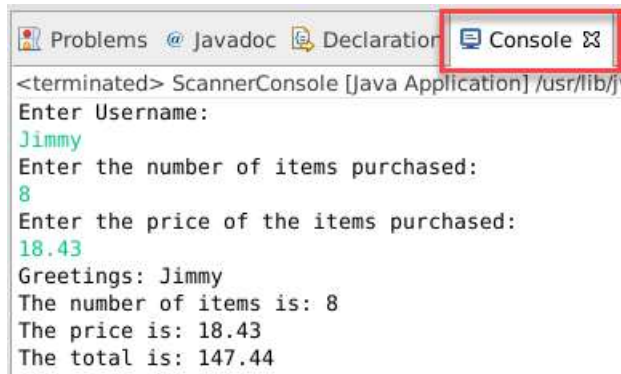
Add the additional code shown, below, in the main method. The code is command interaction with the user.

```
ScannerConsole.java
1 package scanIt;
2
3 import java.util.Scanner;
4
5 public class ScannerConsole {
6
7     public double getTotal(double cst, int amt) {
8
9         double cost = cst;
10        int num = amt;
11        double tot = cost * num;
12
13        return tot;
14    }
15
16    public static void main(String[] args) {
17        // TODO Auto-generated method stub
18        ScannerConsole sc = new ScannerConsole(); //instantiate ScannerConsole class and create object sc
19
20        Scanner myScan = new Scanner(System.in); //Create object myScan of type Scanner. Get command line input
21        System.out.println("Enter Username: "); //Instructing output to the user
22        String userName = myScan.nextLine(); //call method nextline on myScan. It returns a String.
23
24        System.out.println("Enter the number of items purchased: "); //Instructing output to the user
25        int items = myScan.nextInt(); //call method nextint on myScan. It returns int data.
26
27        System.out.println("Enter the price of the items purchased: "); //Instructing output to the user
28        double price = myScan.nextDouble(); //call method nextdouble on myScan. It returns double data.
29
30        double total = sc.getTotal(price, items); //call getTotal on sc object which is of type ScannerConsole.
31
32        System.out.println("Greetings: " + userName); //Output username to the console
33        System.out.println("The number of items is: " + items); //Output items to the console
34        System.out.println("The price is: " + price); //Output the price to the console
35        System.out.println("The total is: " + total); //Output the total to the console
36        myScan.close(); //delete the myScan object from memory.
37    }
38 }
```


Run the application

In the Package Explorer view, right click the ScannerConsole.java file, click **Run As -> Java Application**



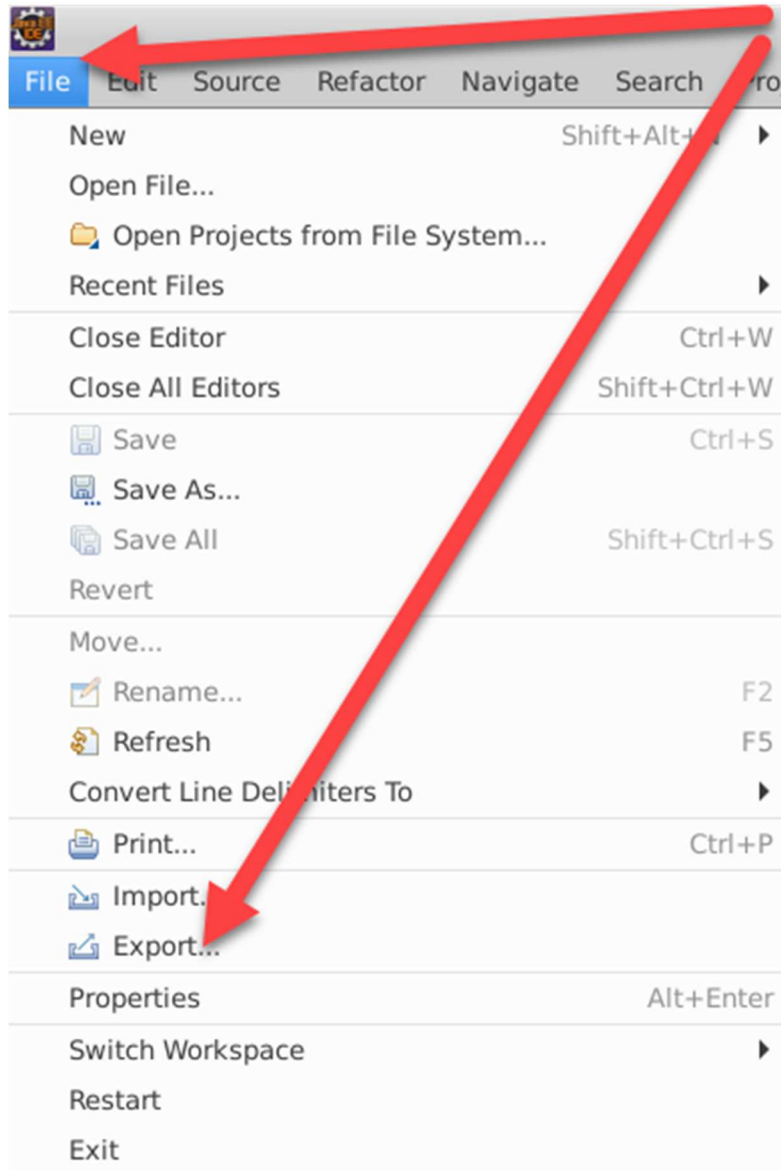


The screenshot shows an IDE's console window. The title bar at the top contains four tabs: 'Problems', 'Javadoc', 'Declaration', and 'Console'. The 'Console' tab is selected and highlighted with a red rectangular box. Below the tabs, the console output is displayed. It begins with the text '<terminated> ScannerConsole [Java Application] /usr/lib/'. This is followed by a series of prompts and user input: 'Enter Username:' followed by 'Jimmy' (in green); 'Enter the number of items purchased:' followed by '8' (in green); and 'Enter the price of the items purchased:' followed by '18.43' (in green). The program then outputs three lines: 'Greetings: Jimmy', 'The number of items is: 8', 'The price is: 18.43', and 'The total is: 147.44'.

```
<terminated> ScannerConsole [Java Application] /usr/lib/
Enter Username:
Jimmy
Enter the number of items purchased:
8
Enter the price of the items purchased:
18.43
Greetings: Jimmy
The number of items is: 8
The price is: 18.43
The total is: 147.44
```

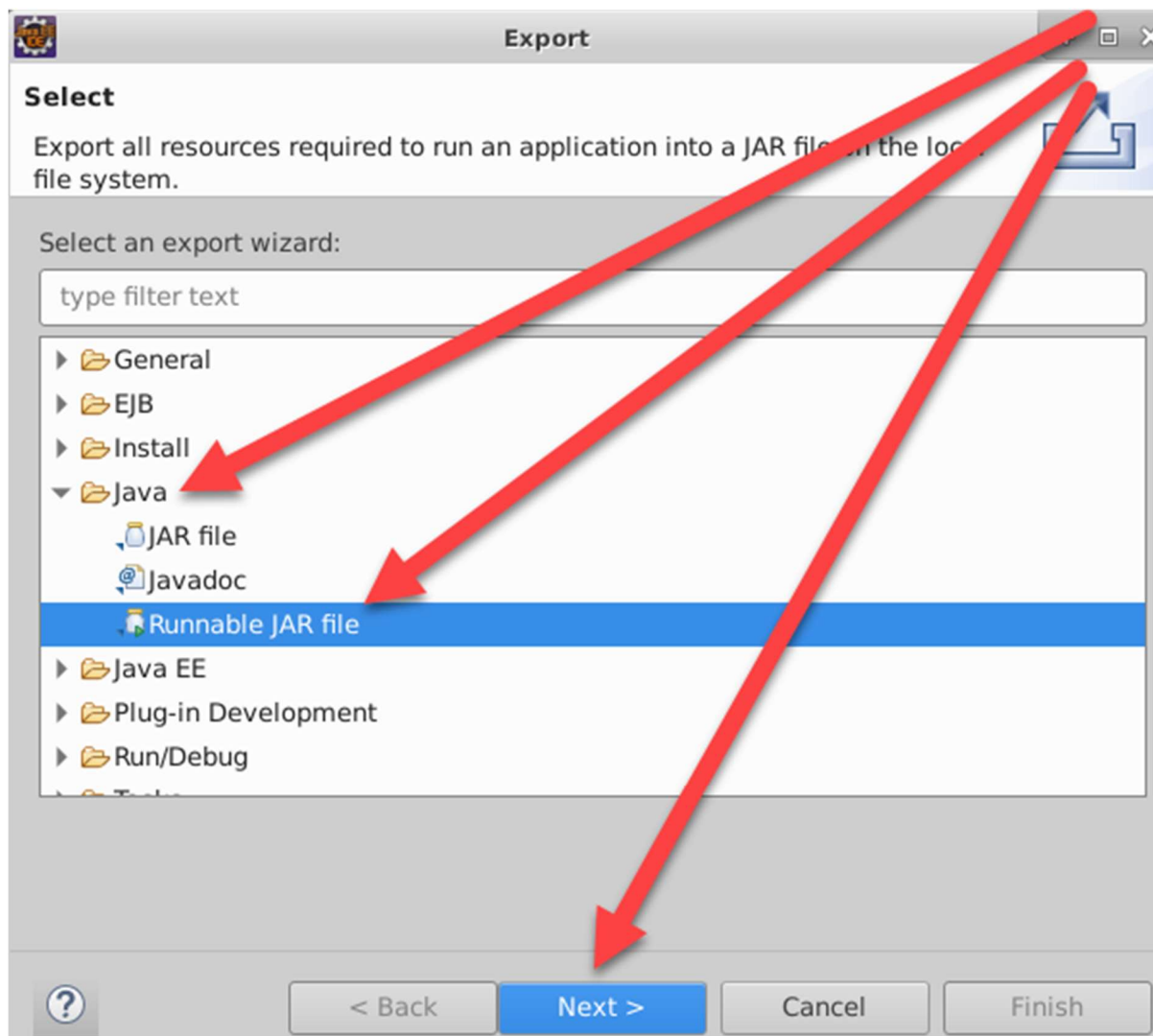
Export a jar file to run from the command line

Click **File** menu -> **Export**



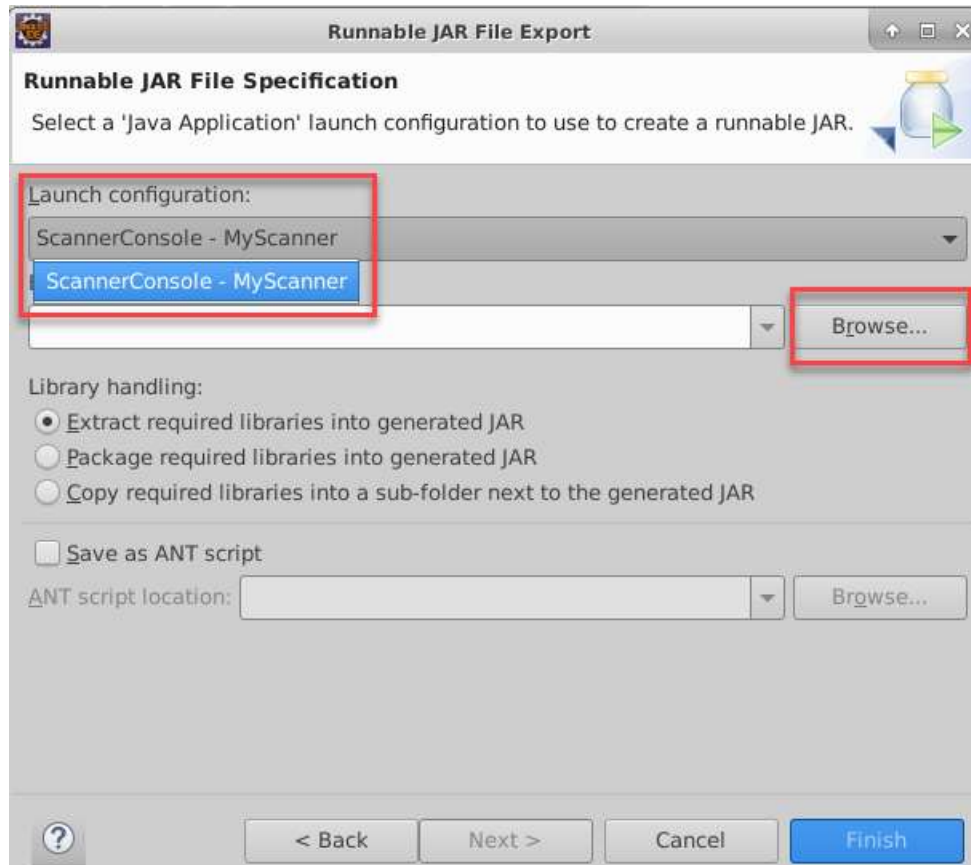
On the **Export** dialog

Click the **Java** folder and select **Runnable Jar file**, and **Next**



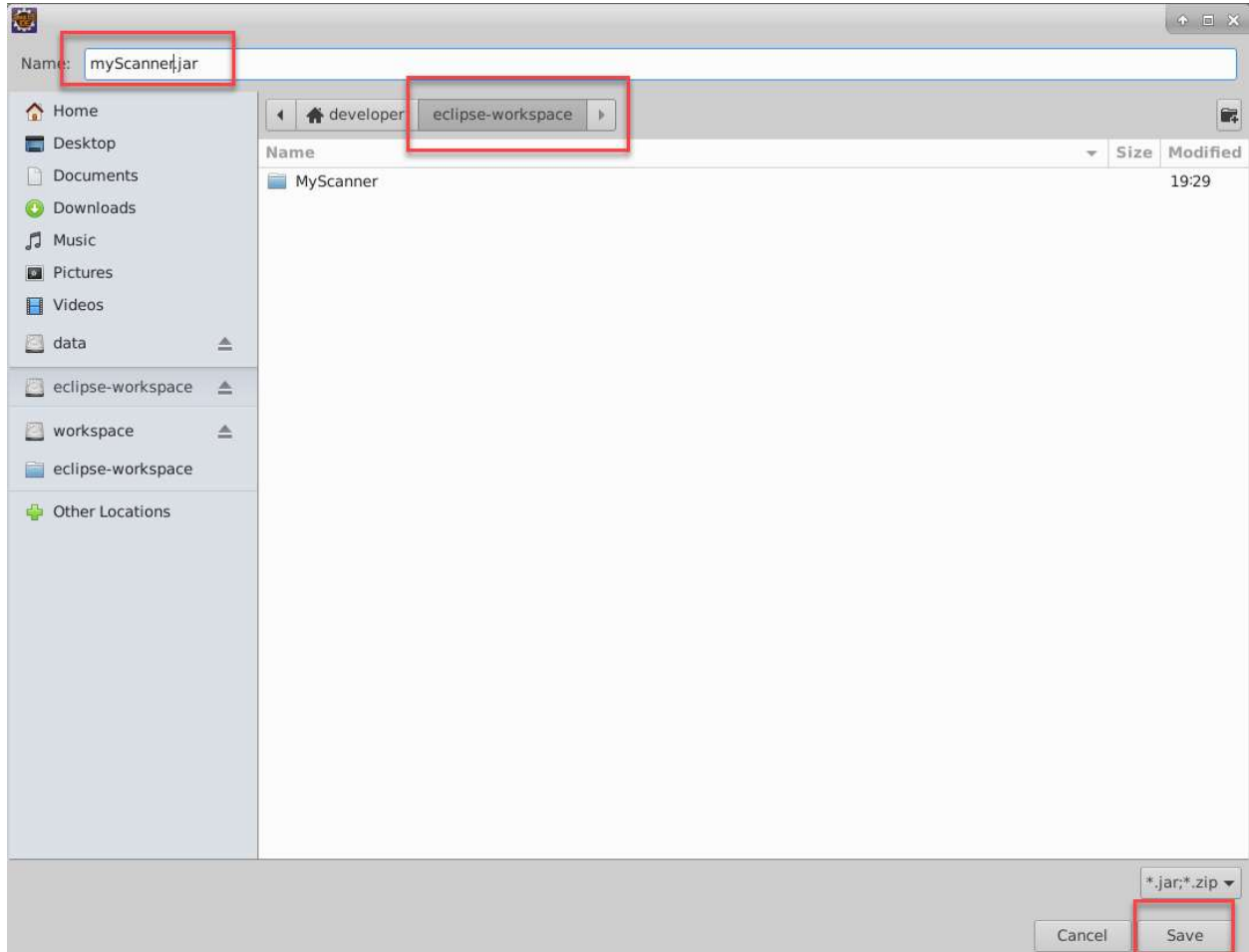
From **Launch configuration** dropdown, select: ScannerConsole – MyScanner, or the available Scanner Launch Configuration

Click **Browse** next to the **Export destination** field, to select the directory for the jar file.

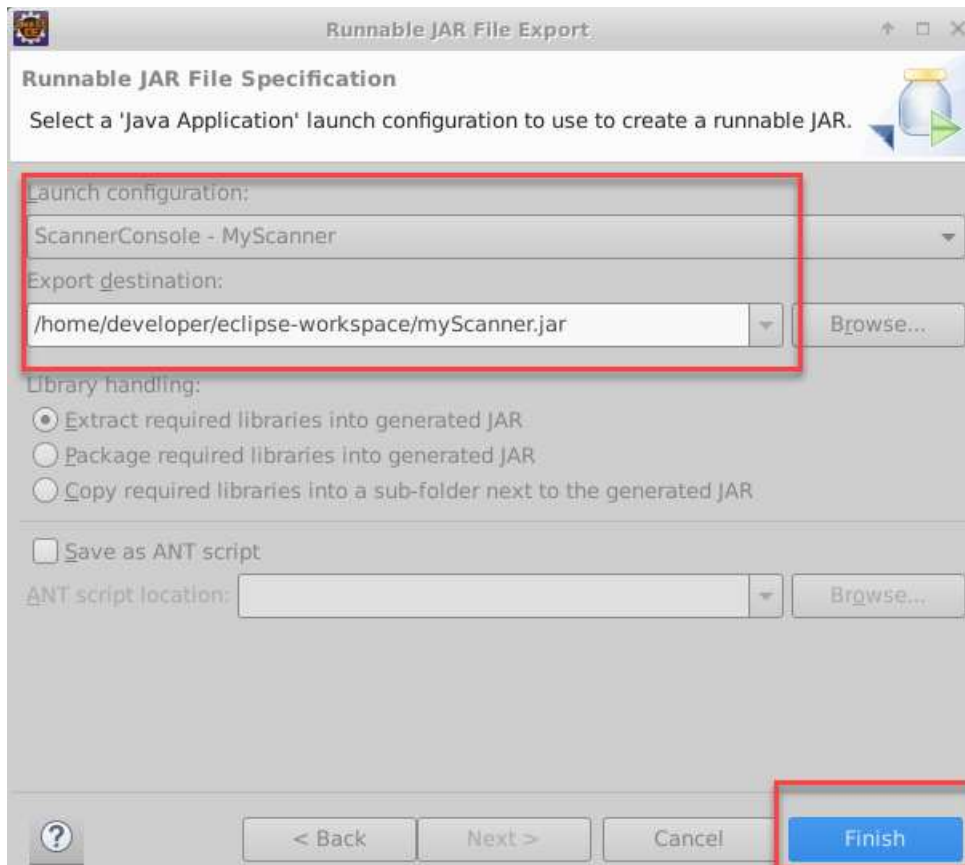


Name the jar file: myScanner.jar

Click **eclipse-workspace** and **Save**.

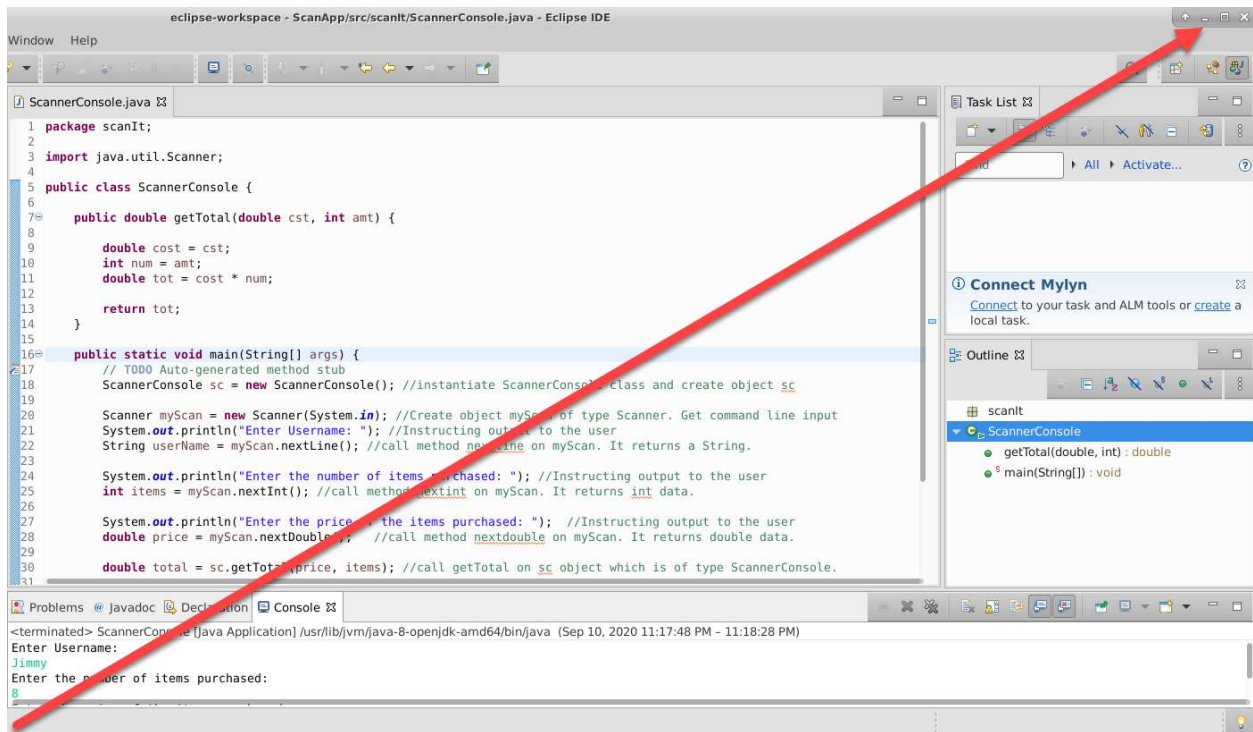


Make note of the **Export destination** directory and click **Finish**

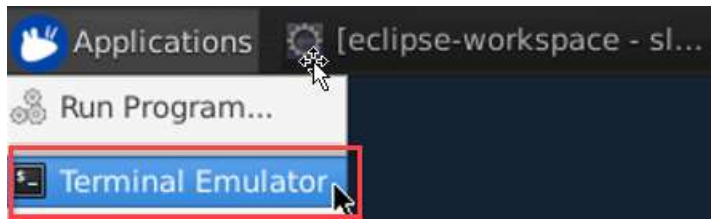


Run the jar from the command line.

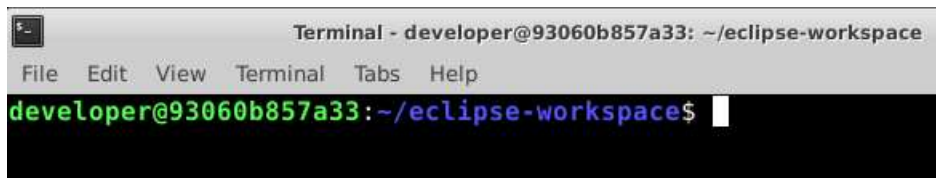
Minimize Eclipse. Click the minimize button in the upper right hand corner. This brings you back to the Desktop.



On the Desktop, click the Applications button, and **Terminal Emulator**.

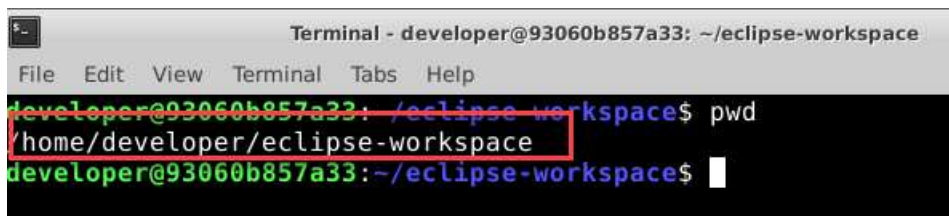


A terminal will open.

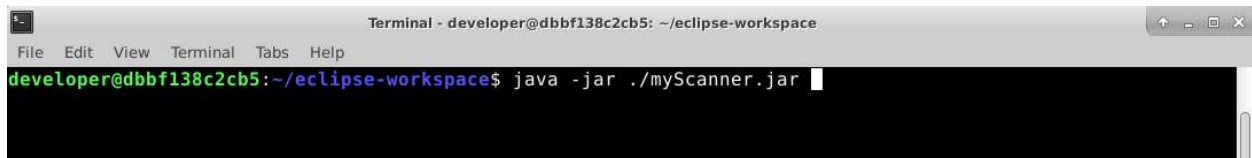


In the terminal, run command: **pwd**

Make sure the present working directory (**pwd**) is: **/home/developer/eclipse-workspace**

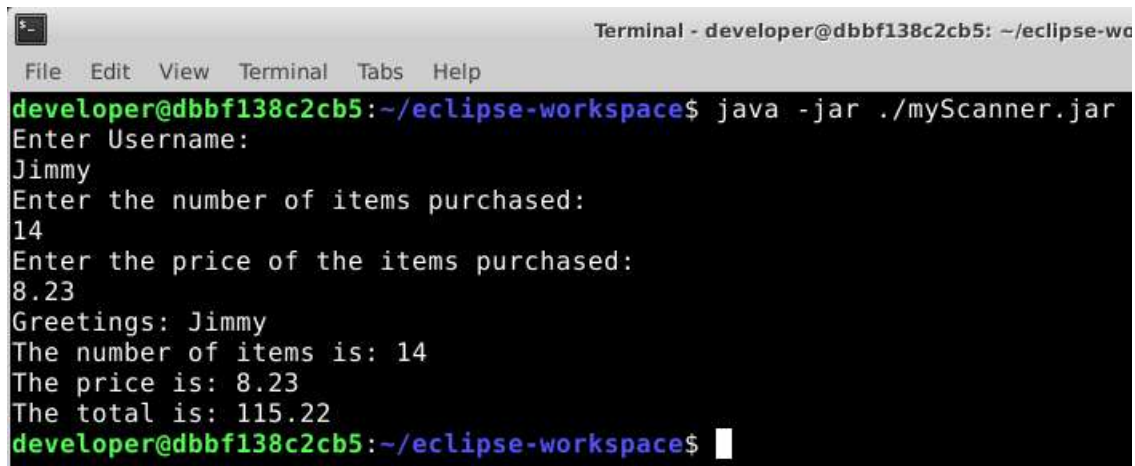


To run the jar file enter the command: `java -jar ./myScanner.jar`

A screenshot of a terminal window titled "Terminal - developer@dbbf138c2cb5: ~/eclipse-workspace". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The command prompt shows the user entering the command `java -jar ./myScanner.jar`.

```
Terminal - developer@dbbf138c2cb5: ~/eclipse-workspace
File Edit View Terminal Tabs Help
developer@dbbf138c2cb5:~/eclipse-workspace$ java -jar ./myScanner.jar
```

Enter a name, a number of items, and a price. Observe the result

A screenshot of a terminal window titled "Terminal - developer@dbbf138c2cb5: ~/eclipse-workspace". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The command prompt shows the user entering the command `java -jar ./myScanner.jar`. The program then prompts for a username, the number of items purchased, and the price of the items purchased. The user enters "Jimmy", "14", and "8.23" respectively. The program then outputs the greetings, the number of items, the price, and the total.

```
Terminal - developer@dbbf138c2cb5: ~/eclipse-workspace
File Edit View Terminal Tabs Help
developer@dbbf138c2cb5:~/eclipse-workspace$ java -jar ./myScanner.jar
Enter Username:
Jimmy
Enter the number of items purchased:
14
Enter the price of the items purchased:
8.23
Greetings: Jimmy
The number of items is: 14
The price is: 8.23
The total is: 115.22
developer@dbbf138c2cb5:~/eclipse-workspace$
```

End of Lab

