# Object Oriented Programming using Java

Lab 1

## Content

- Introduction to the IDE Tool
- Getting started with NetBeans
  - How to install
  - A Quick start Guide
  - Displaying outputs
  - How to run
  - How to debug
- Reading inputs from user
- Hands on

## Introduction to IDE

- In this course we will practice the OOP concept using Java Standard Edition programming language "Java SE".

- So we need to choose a suitable IDE to code in Java.

## INTRODUCTION TO IDE

- An **Integrated Development Environment** is a computer software to help computer programmers develop software.

- **The Leaders:**
  - NetBeans
  - Microsoft Visual Studio
  - Eclipse

## Introduction to IDE- cont.

- **What does an IDE consist of?**
  - Source code Editor.
  - Compiler and/or interpreter.
  - Build- automation tools.

- **Optional Tools:**
  - Debugger.
  - **Various tools to simplify the construction of a GUI**.

## Getting Started with NetBeans

- How to install
- A Quick start Guide
- Displaying outputs

## INSTALLATION

1. **Installing JDK:** in order to install NetBeans you need to first install JDK.

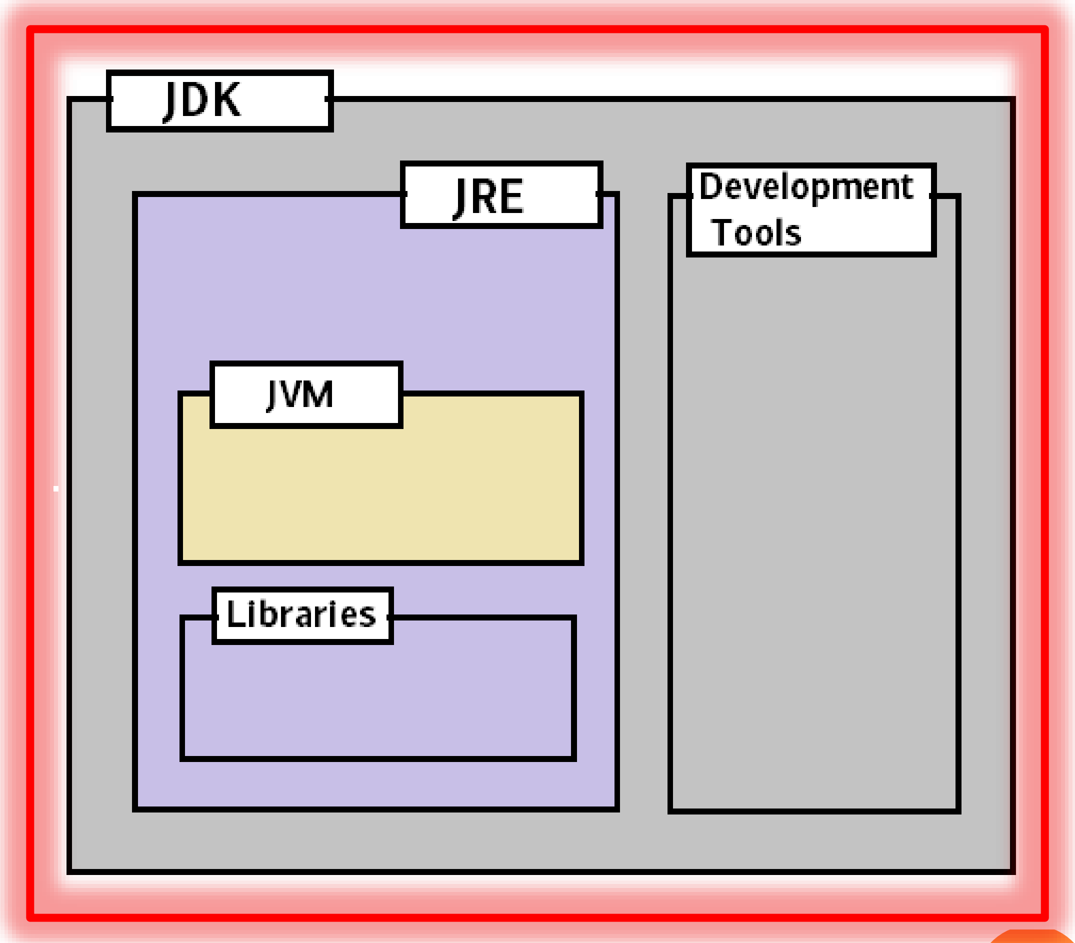
2. **NetBeans installation:** run the installation application source for version 8.0.2.

## Installation

- **<u>What is a JDK ?</u>**

The JDK includes

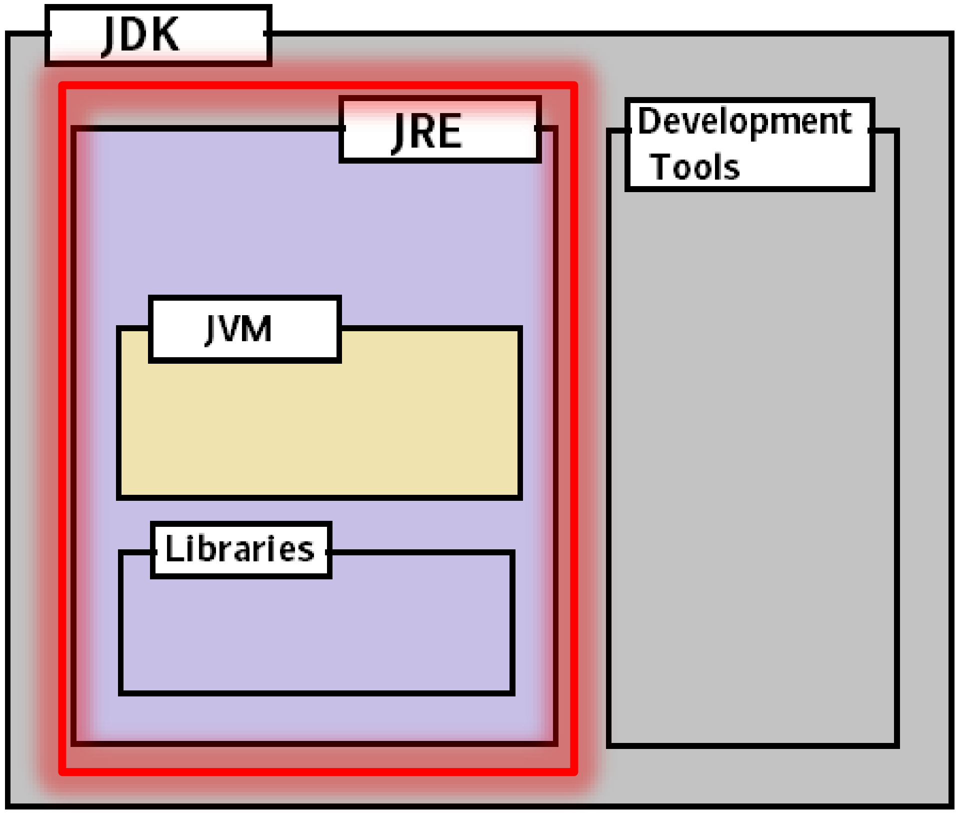
- a set of tools for compiling and running your java code
- "Java Runtime Environment" JRE .



## INSTALLATION

- **<u>What is a JRE ?</u>**

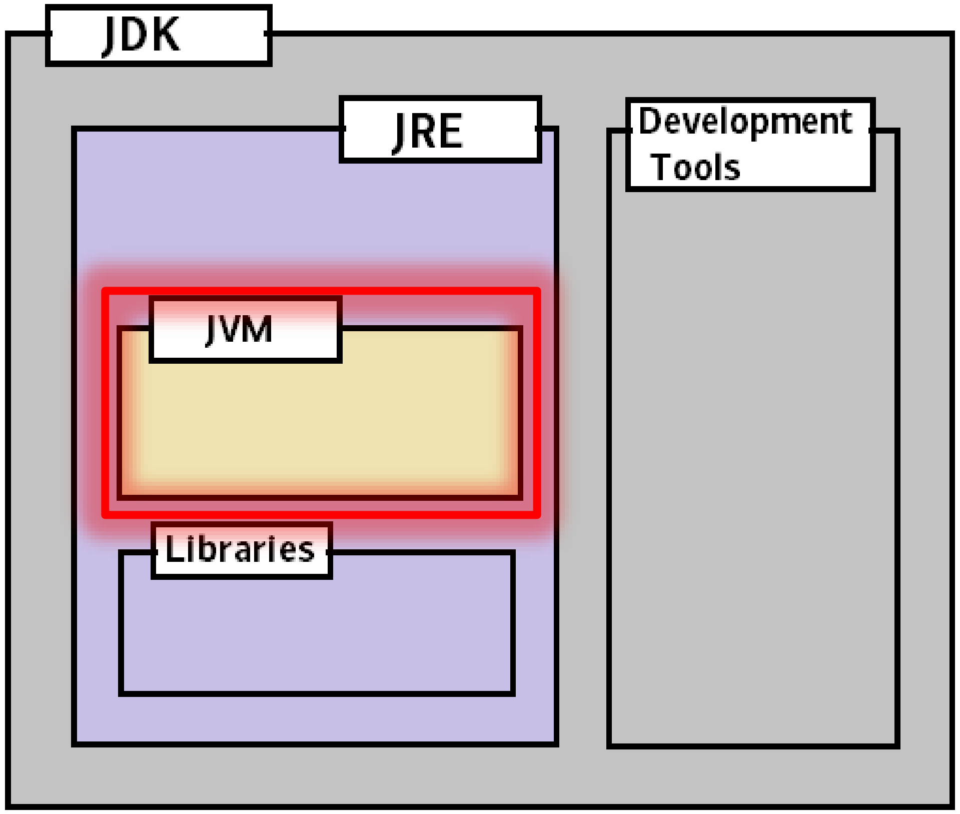
1. includes the JVM
2. code libraries that are necessary for running programs



## INSTALLATION

- **<u>What is a JVM ?</u>**

JVM is the heart of the java language “write once, run anywhere”.

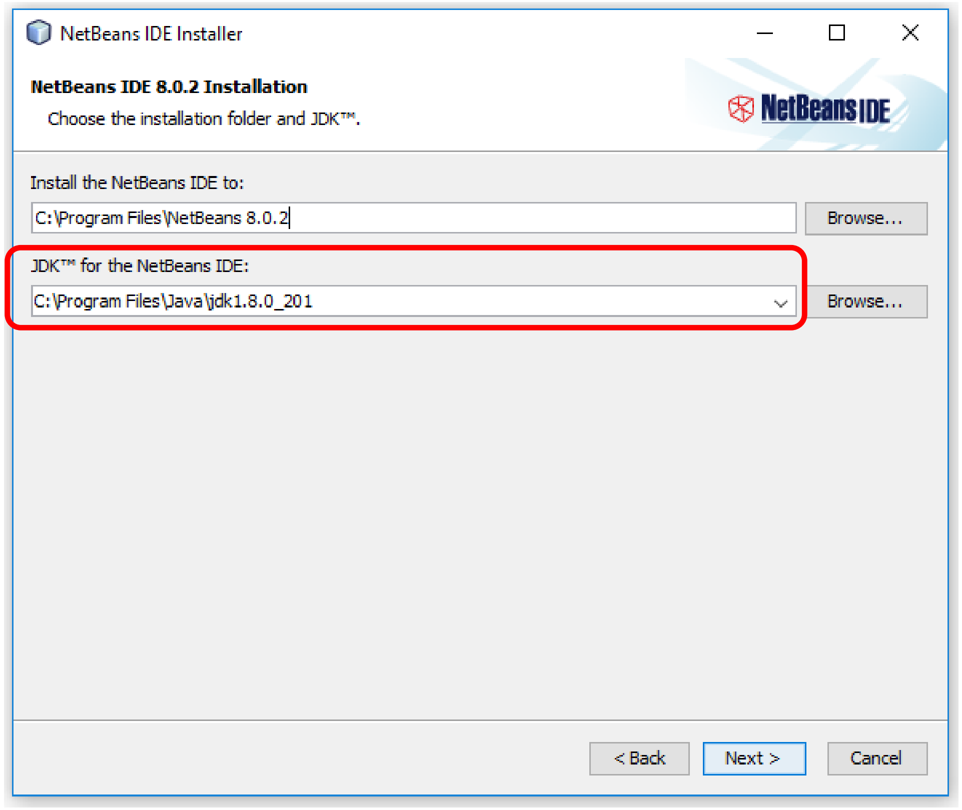


## INSTALLATION

- JVM is the virtual engine and the one which enables byte code support.
- JRE contains JVM and all the other libraries to run Java application. It is enough to run any Java application.
- JDK is a superset which comprises of JVM, JRE, and the tools to develop Java Application. Its primary objective is to provide support for the build and compilation.

## INSTALLATION

- When installing the NetBeans make sure that the path of the JDK is the same path of the JDK you installed.



NetBeans IDE Installer

**NetBeans IDE 8.0.2 Installation**

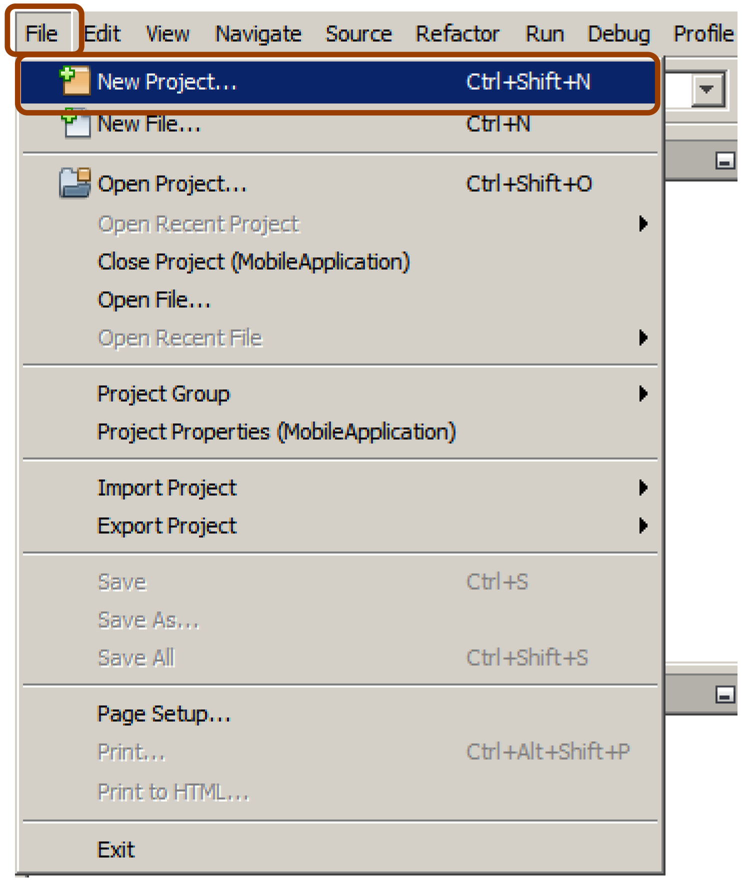
Choose the installation folder and JDK™.

Install the NetBeans IDE to: C:\Program Files\NetBeans 8.0.2

JDK™ for the NetBeans IDE: C:\Program Files\Java\jdk1.8.0_201

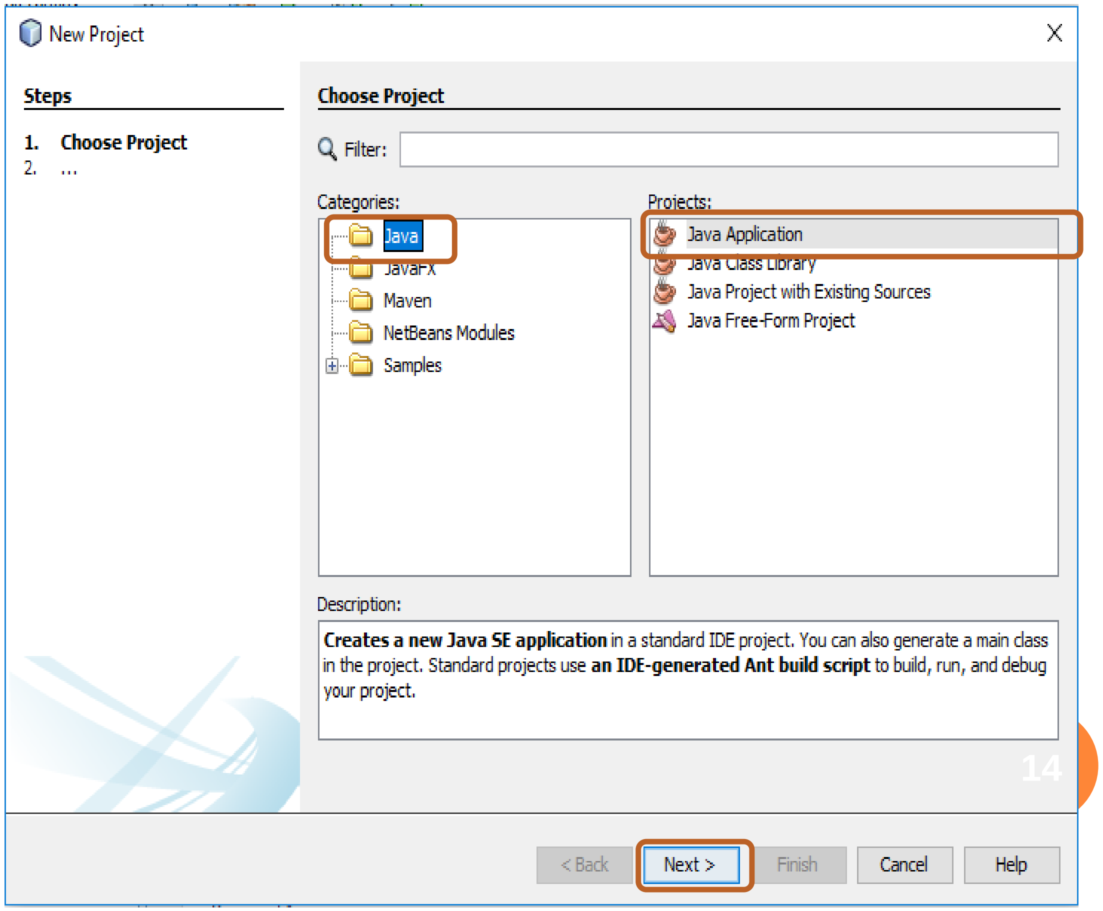
## Quick Start Guide

- Choose File > New Project, as shown in the figure below.



## Quick Start Guide

- In the New Project wizard, expand the Java category and select Java Application as shown in the figure below.
- Then click Next.

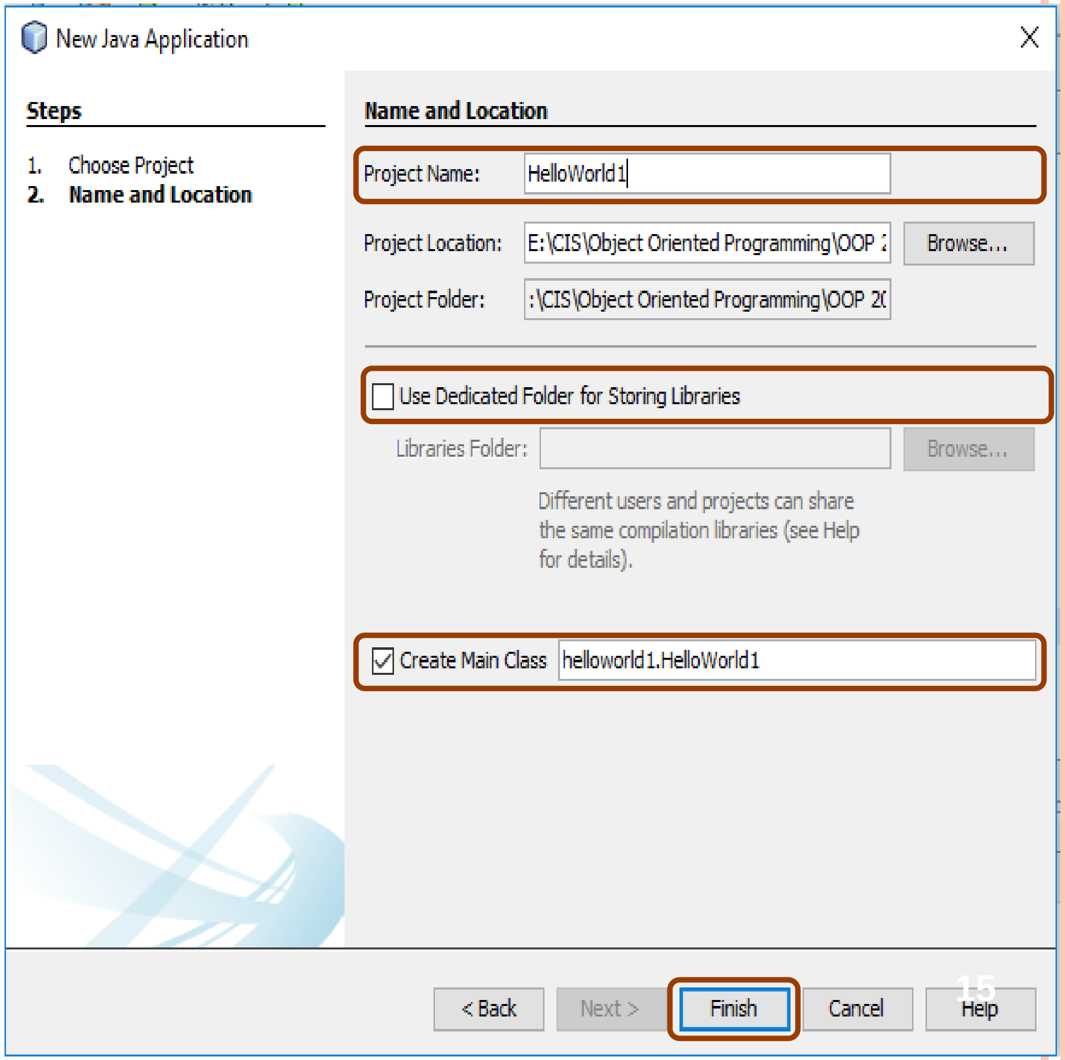


Description:

**Creates a new Java SE application** in a standard IDE project. You can also generate a main class in the project. Standard projects use **an IDE-generated Ant build script** to build, run, and debug your project.

# Quick Start Guide

- In the Project Name field, type HelloWorld.
- Leave the Use Dedicated Folder for Storing Libraries checkbox unselected.
- In the Create Main Class field, type helloworldapp.HelloWorldApp
(or it will be automatically written).


New Java Application

**Steps**

1. Choose Project
2. **Name and Location**

**Name and Location**

Project Name: HelloWorld1

Project Location: E:\CIS\Object Oriented Programming\OOP 2

Project Folder: :\CIS\Object Oriented Programming\OOP 20

[ ] Use Dedicated Folder for Storing Libraries

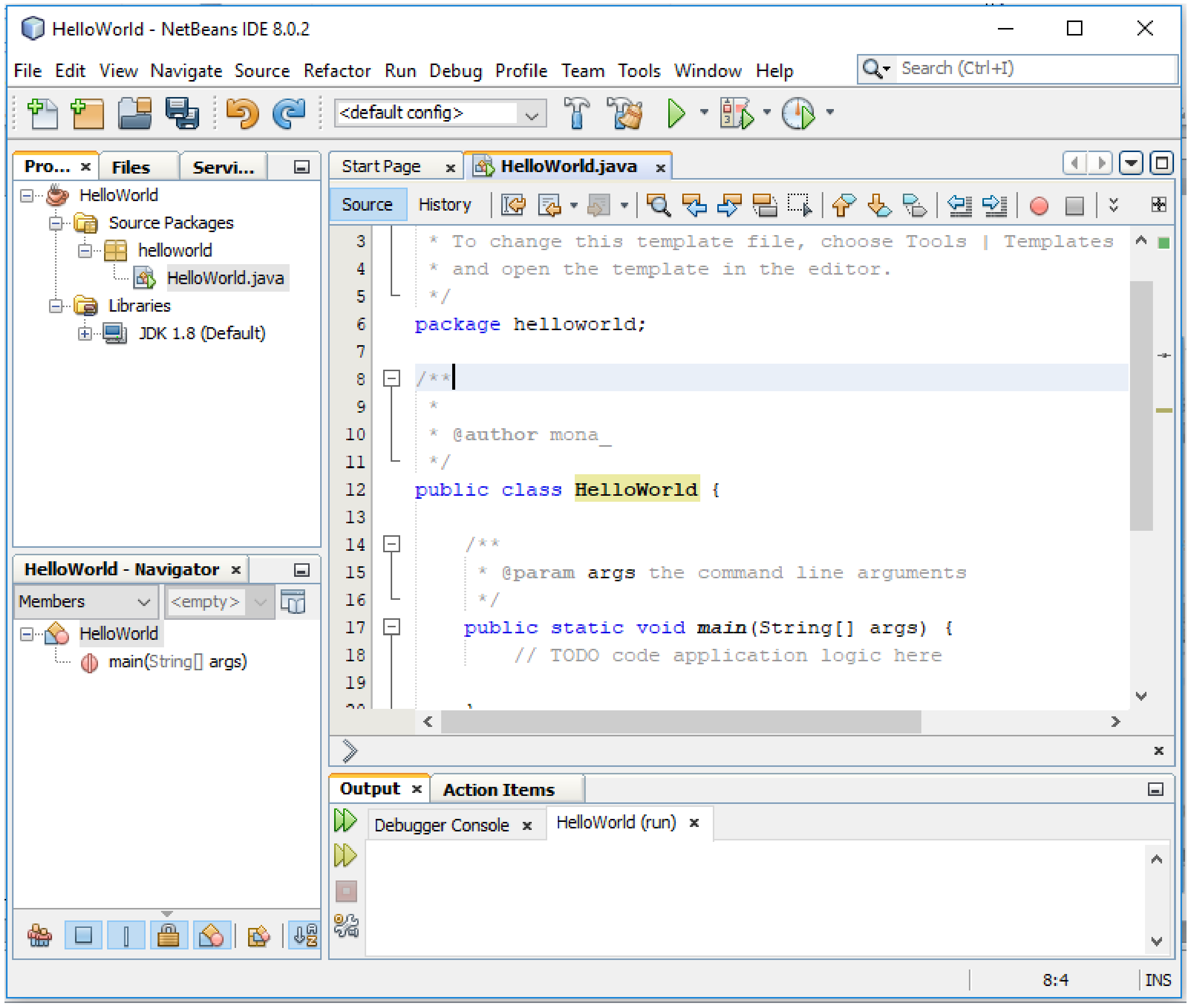
Libraries Folder:

Different users and projects can share the same compilation libraries (see Help for details).

[✓] Create Main Class helloworld1.HelloWorld1

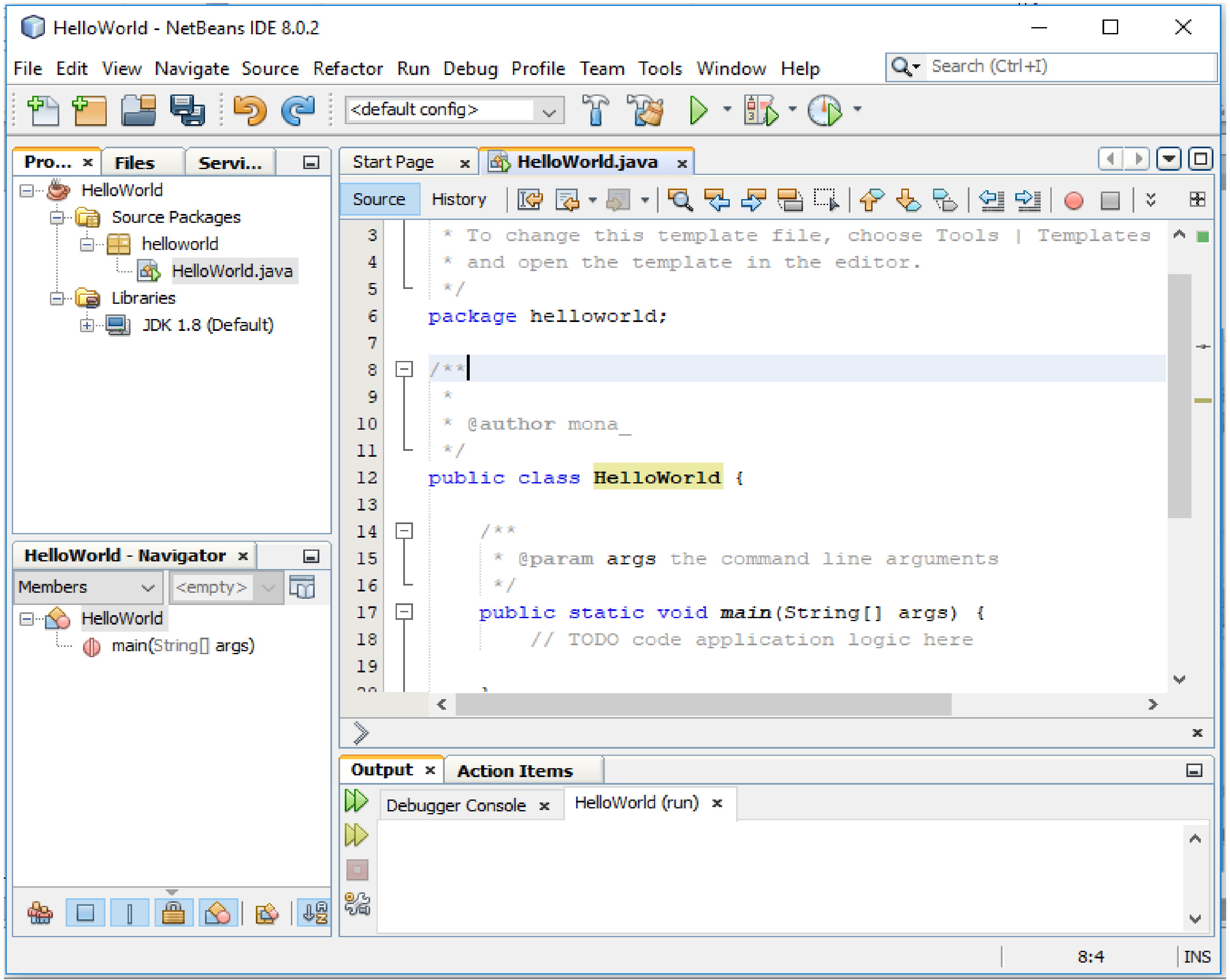
## Quick Start Guide

- The project is created and opened in the IDE.



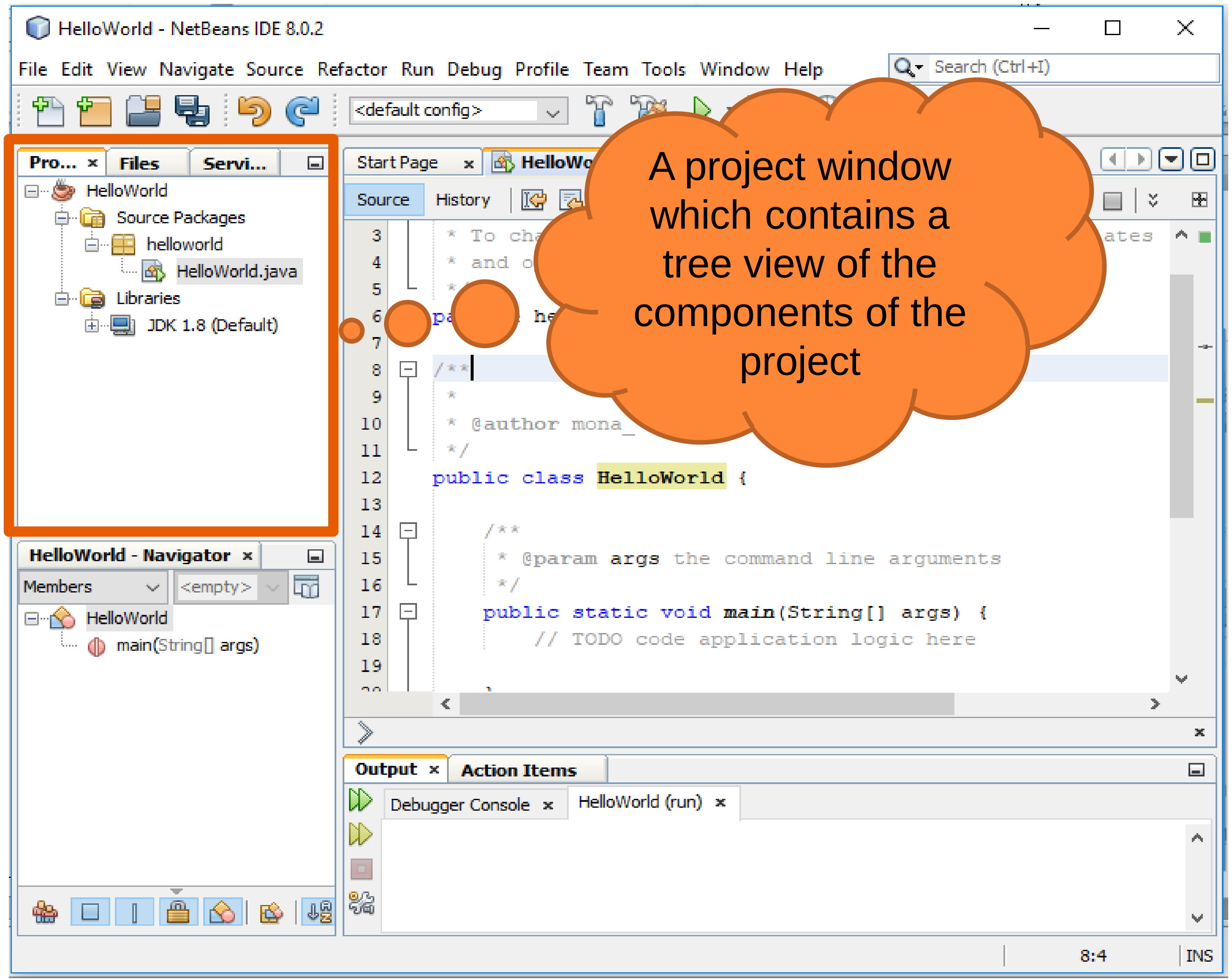
```
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package helloworld;

/**
 *
 * @author mona_
 */
public class HelloWorld {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here

```

## Quick Start Guide



```
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package helloworld;

/**
 *
 * @author mona_
 */
public class HelloWorld {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here

```

# QUICK START GUIDE

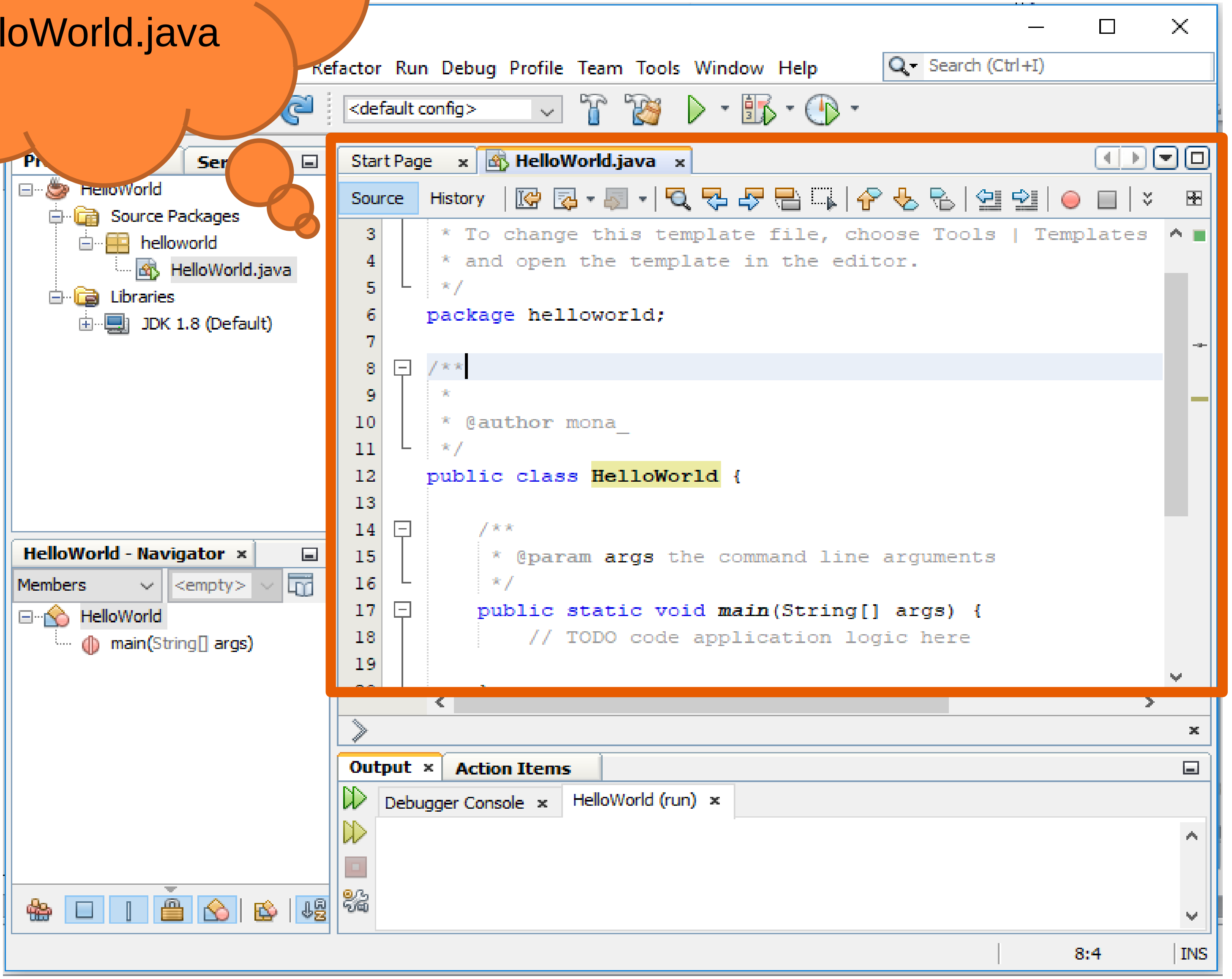


```
 * To cha                                              ates
 * and o
 *
pa          he

/**
 *
 * @author mona_
 */
public class HelloWorld {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
```

A project window which contains a tree view of the components of the project

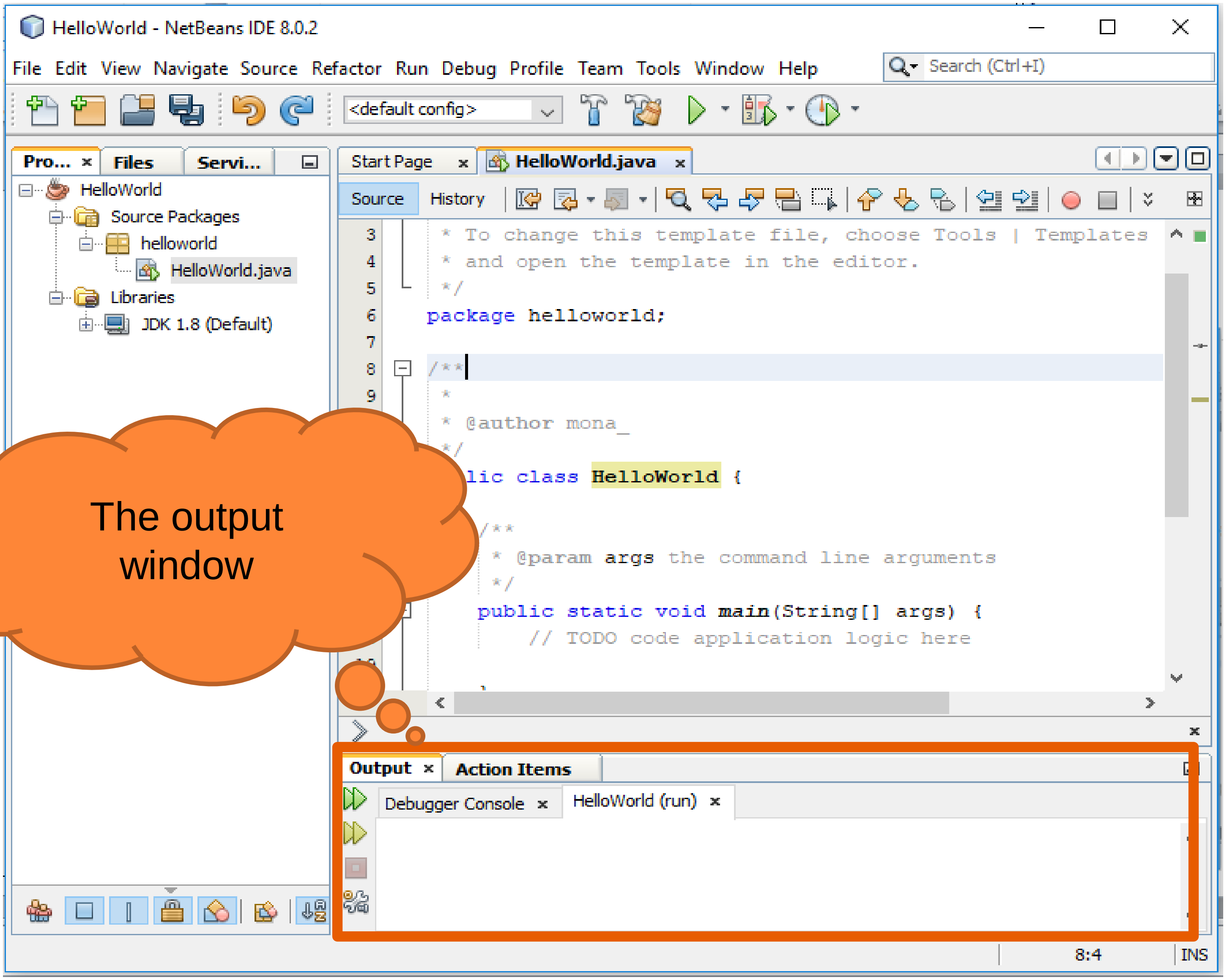
The source editor with class HelloWorld.java

## IDE



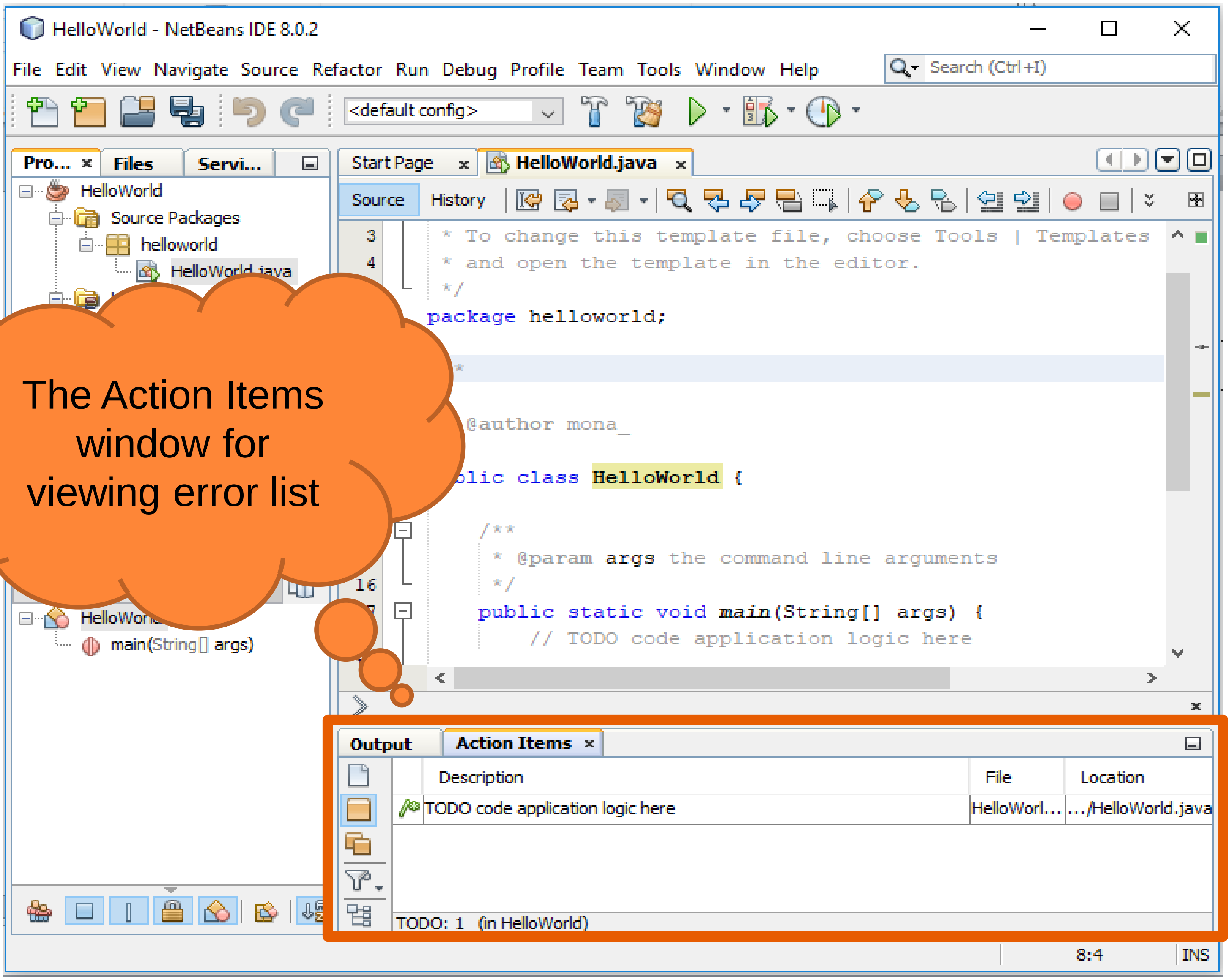
```
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package helloworld;

/**
 *
 * @author mona_
 */
public class HelloWorld {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
```

## Quick Start Guide



```
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package helloworld;

/**
 *
 * @author mona_
 */
...lic class HelloWorld {

  /**
   * @param args the command line arguments
   */
    public static void main(String[] args) {
        // TODO code application logic here
```

## Quick Start Guide



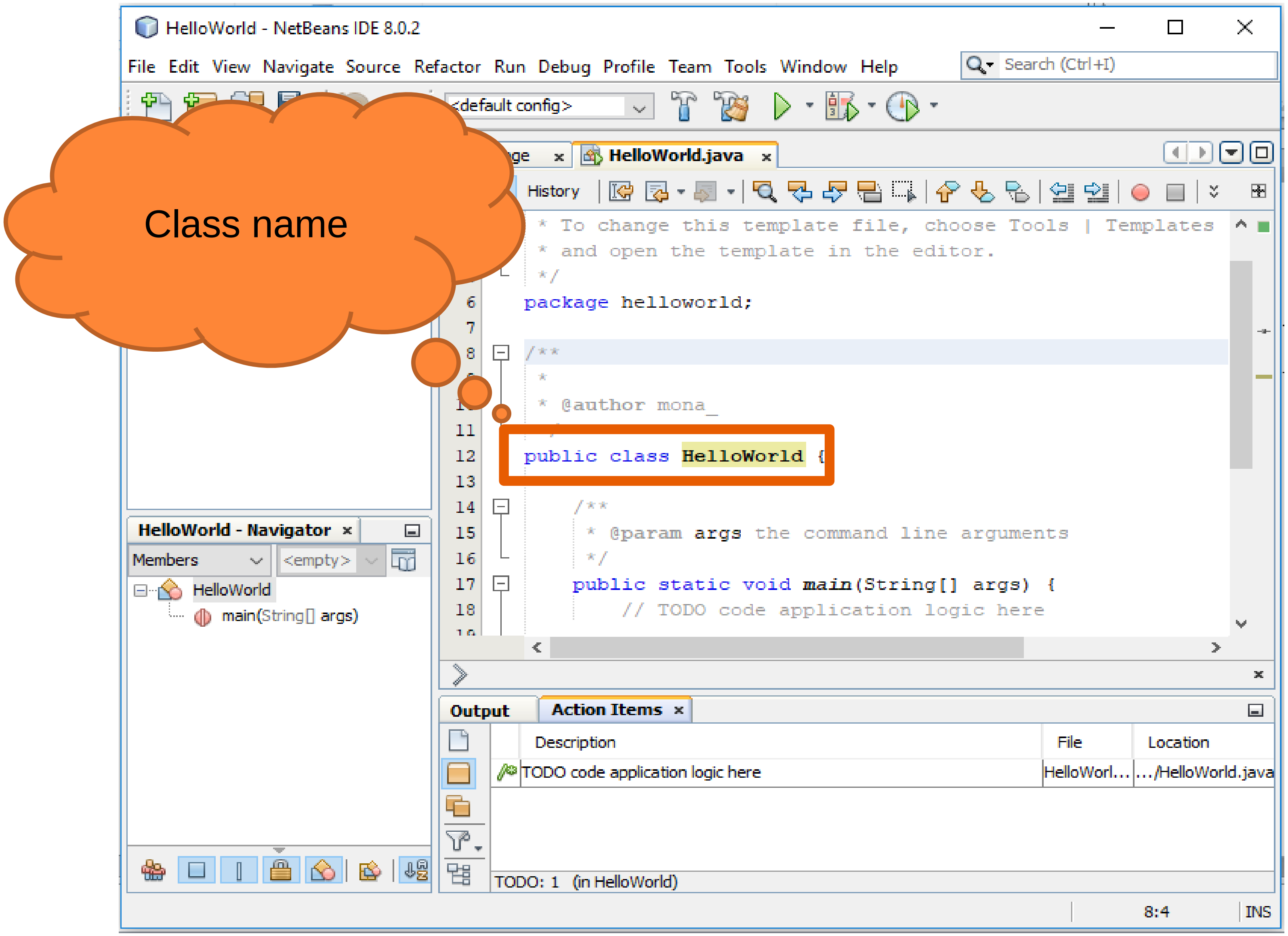
```
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package helloworld;

/*

 @author mona_

blic class HelloWorld {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
```

| | Description | File | Location |
|---|---|---|---|
| | TODO code application logic here | HelloWorl... | .../HelloWorld.java |

TODO: 1 (in HelloWorld)

The Action Items window for viewing error list

# Quick Start Guide



```
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package helloworld;

/**
 *
 * @author mona_
 */
public class HelloWorld {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
```

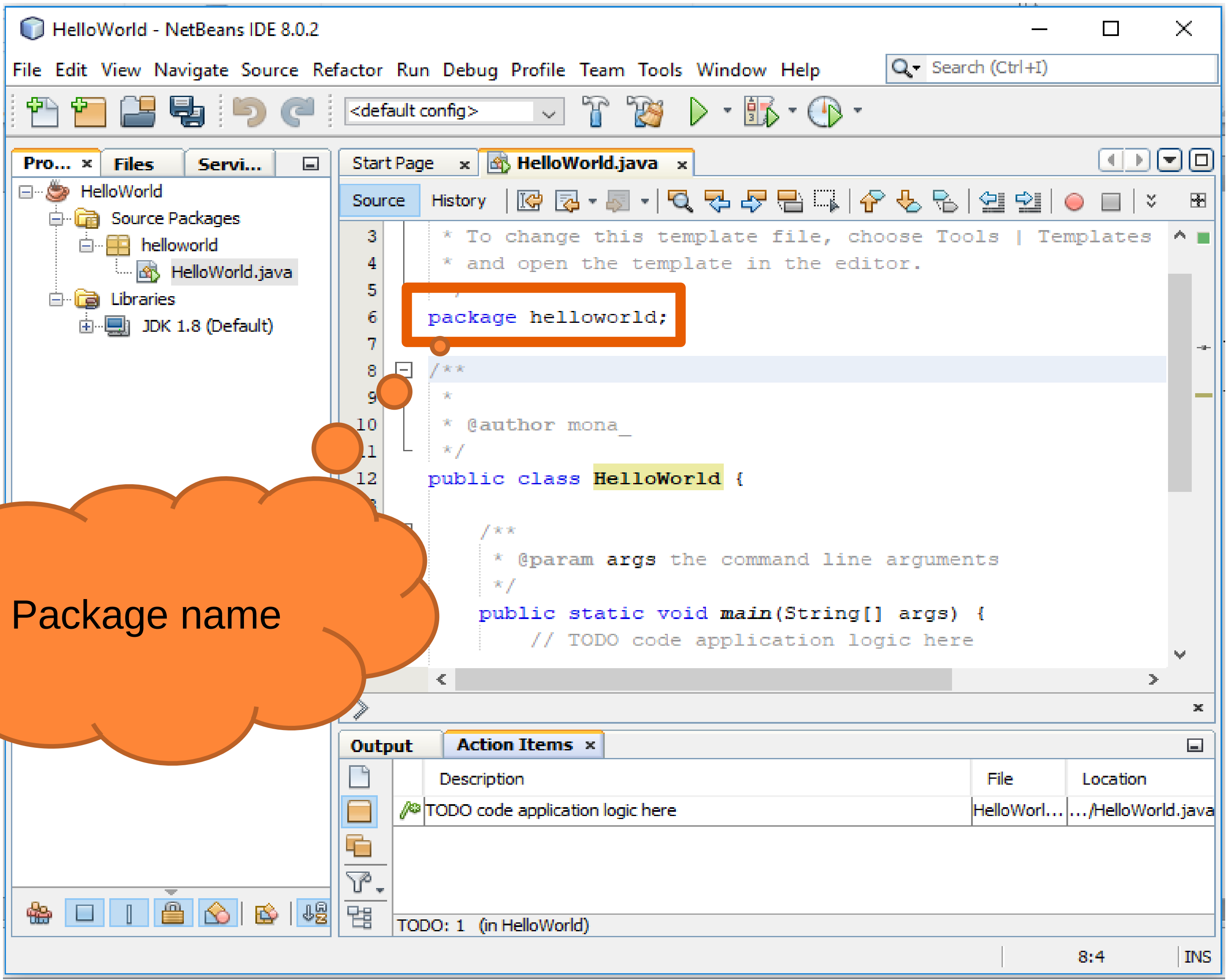
HelloWorld
main(String[] args)

| Description | File | Location |
|---|---|---|
| TODO code application logic here | HelloWorl... | .../HelloWorld.java |

TODO: 1 (in HelloWorld)

8:4 INS

## Quick Start Guide



```
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package helloworld;

/**
 *
 * @author mona_
 */
public class HelloWorld {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
```

| Description | File | Location |
|---|---|---|
| TODO code application logic here | HelloWorl... | .../HelloWorld.java |

TODO: 1 (in HelloWorld)

Package name

## PACKAGES

- A group of related classes.
- The main reason for using packages is to guarantee the uniqueness of class names in the same package.
- To guarantee a unique package name, Sun Microsystems Company recommends that you use your company's Internet domain name (which is known to be unique) written in reverse.
  - For example, asset.com is a domain when written in reverse order, it turns into the package name com.asset.
  - That package can then be further subdivided into subpackages such as com.asset.corejava.
- Packages can be nested.
- Standard Java Packages: java.* , javax.*
  - such as java.lang, java.util, java.net, and so on

## PACKAGES

- A class can use all classes from its own package and all public classes from other packages.

- To access public classes in other packages we use the key word import

```
import java.util.Date;
```

Or we can import all classes in a package

```
import java.util.*;
```

- If the same class “Date” exists in two packages and they are imported in the project , I have to specify which date I want to use

```
import java.util.*;
import java.sql.*;
import java.util.Date;
Date today;
```

Or

```
java.util.Date deadline;
java.sql.Date today;
```

## DISPLAYING OUTPUTS

- To Output the message we use:
  - Print: shows value passed to it.

    System.*out.print (" …");*

    System.*out.print (" Hello");*

  - Println: shows value followed by new Line

    System.out.println(" …");

    System.*out.*println *(" Hello");*

  - Printf: shows value with a certain format

    System.out.printf(….);

## Displaying Outputs: Printf(…)

- *System.out.printf(“%parameter”, value);*
  - Common parameters:

    *'d': decimal integer*
    *'f': decimal notation for float*
    *'c': for a character*
    *'s': for a string.*
    *'b': for a boolean value → "true" or "false"*
    *'o': octal integer*
    *'x': hexadecimal integer*
    *'n': "%n" has the same effect as "\n".*

## Displaying outputs: Printf(...) cont’

- Examples:
  1. *System.out.printf(“%s”, “Hello”);* → ***Hello***
  2. *String str=“hello”;*

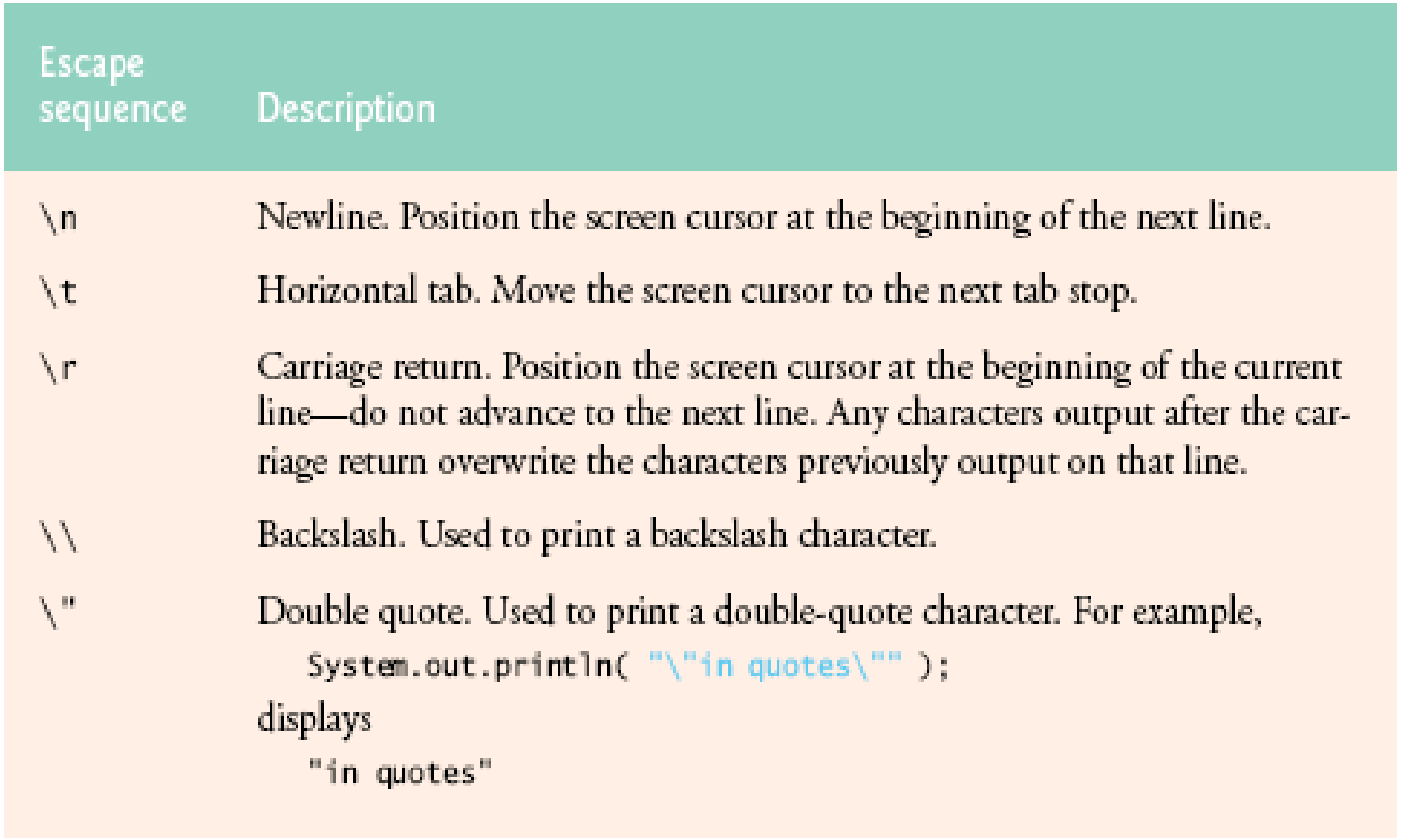
     *System.out.printf(“%s”, st*
  3. *System.out.printf(“%d”*
  4. *int x=10;*

     *System.out.printf(“%s=%*
  5. *int x=1000;*

     *System.out.printf(“ d”, ), → **1,000***
  6. *float y =5.365f;*
     *System.out.printf(“%.1f”, y);* → ***5.4***

.1 means round to the nearest 1 decimal number.
So .5 means round to the nearest 5 decimals

## DISPLAYING OUTPUTS CONT’



| Escape sequence | Description |
|---|---|
| `\n` | Newline. Position the screen cursor at the beginning of the next line. |
| `\t` | Horizontal tab. Move the screen cursor to the next tab stop. |
| `\r` | Carriage return. Position the screen cursor at the beginning of the current line—do not advance to the next line. Any characters output after the carriage return overwrite the characters previously output on that line. |
| `\\` | Backslash. Used to print a backslash character. |
| `\"` | Double quote. Used to print a double-quote character. For example, `System.out.println( "\"in quotes\"" );` displays `"in quotes"` |

Fig. 2.5 | Some common escape sequences.

## Exercise 1: HelloWorld

*int x=1500000;*

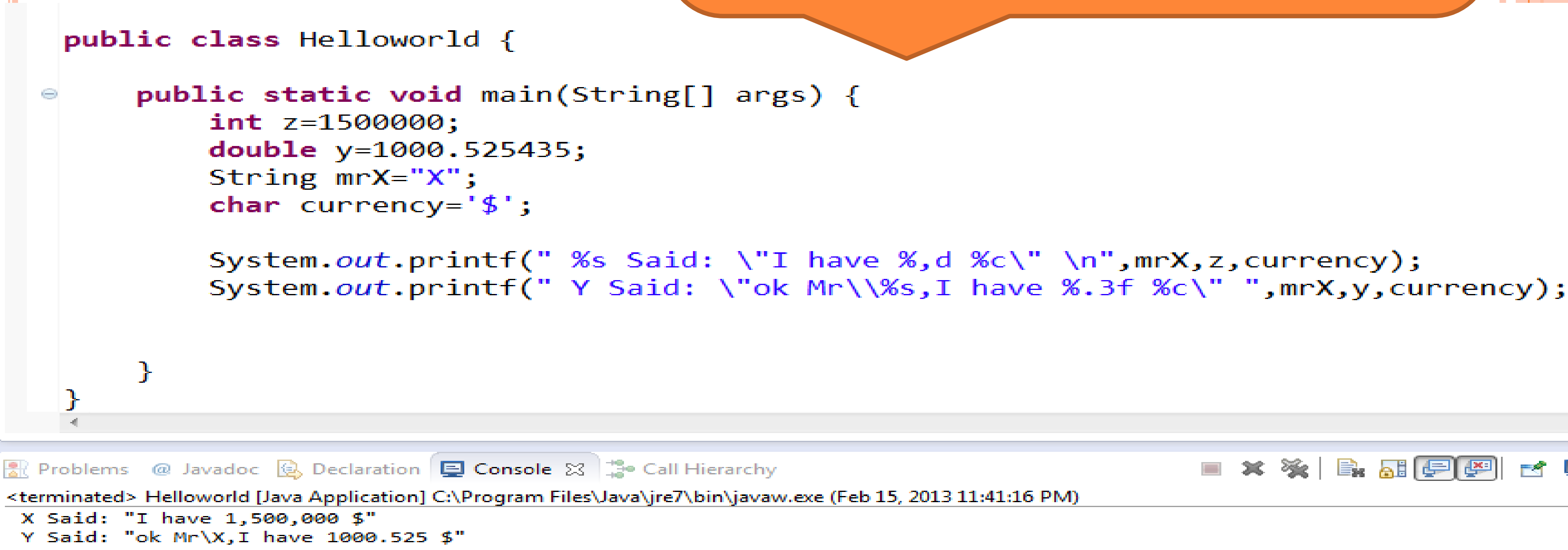
*double y=1000.525435;*

*String mrX="X";*

*char currency='$';*

**X said:"I have 1,500,000$”**

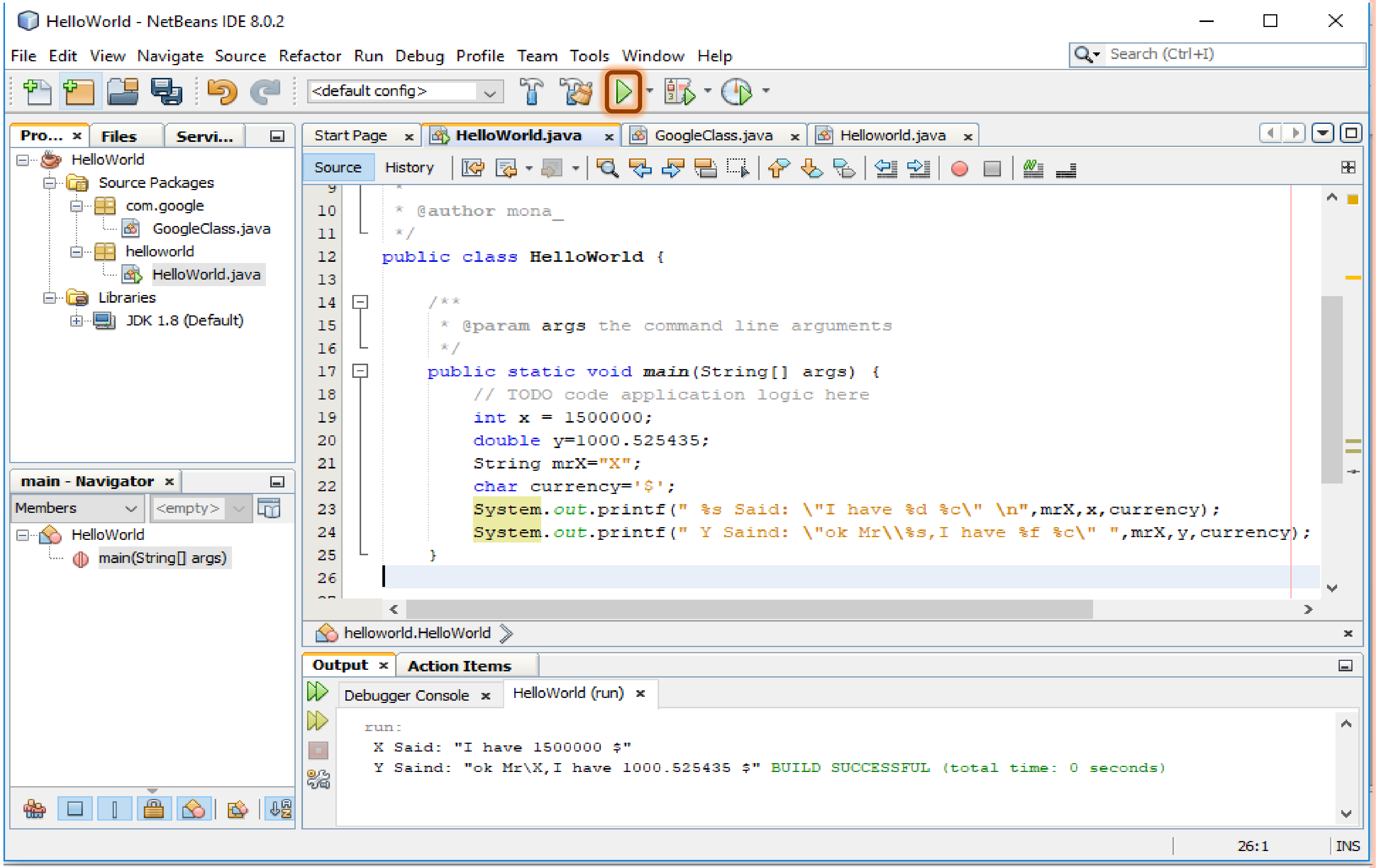
**Y said:"Ok Mr\X, I have 1000.525$"**



```
public class Helloworld {

    public static void main(String[] args) {
        int z=1500000;
        double y=1000.525435;
        String mrX="X";
        char currency='$';

        System.out.printf(" %s Said: \"I have %,d %c\" \n",mrX,z,currency);
        System.out.printf(" Y Said: \"ok Mr\\%s,I have %.3f %c\" ",mrX,y,currency);


    }
}
```

```
<terminated> Helloworld [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (Feb 15, 2013 11:41:16 PM)
 X Said: "I have 1,500,000 $"
 Y Said: "ok Mr\X,I have 1000.525 $"
```

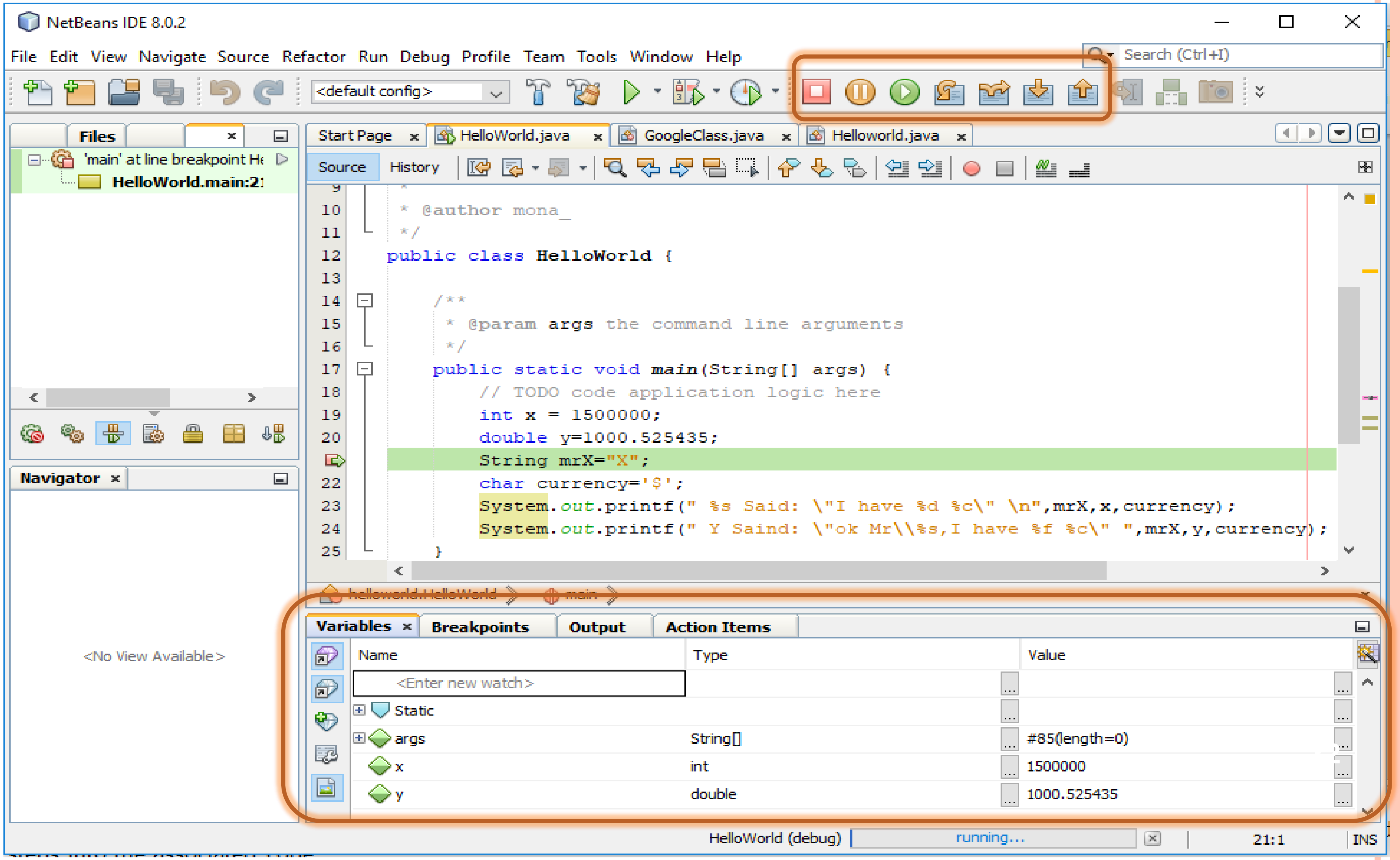
# How To run !!



```
 * @author mona_
 */
public class HelloWorld {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        int x = 1500000;
        double y=1000.525435;
        String mrX="X";
        char currency='$';
        System.out.printf(" %s Said: \"I have %d %c\" \n",mrX,x,currency);
        System.out.printf(" Y Saind: \"ok Mr\\%s,I have %f %c\" ",mrX,y,currency);
    }
```

```
run:
 X Said: "I have 1500000 $"
 Y Saind: "ok Mr\X,I have 1000.525435 $" BUILD SUCCESSFUL (total time: 0 seconds)
```

# How to Debug !!



```
 *
 * @author mona_
 */
public class HelloWorld {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        int x = 1500000;
        double y=1000.525435;
        String mrX="X";
        char currency='$';
        System.out.printf(" %s Said: \"I have %d %c\" \n",mrX,x,currency);
        System.out.printf(" Y Saind: \"ok Mr\\%s,I have %f %c\" ",mrX,y,currency);
    }
```

| Name | Type | Value |
|---|---|---|
| <Enter new watch> | | |
| Static | | |
| args | String[] | #85(length=0) |
| x | int | 1500000 |
| y | double | 1000.525435 |

## Reading Input from User

- To **read** something from the console:
  1. Import the package that has the class **Scanner**.

  (The **import** line is to be written under the name of your package)

  → ***import java.util.Scanner;***

  2. Take an object from the **Scanner** class.

  → ***Scanner** input **=new Scanner(System.in) ;***

  3. Use the **Scanner** suitable method to read the next **input** according to its data type.

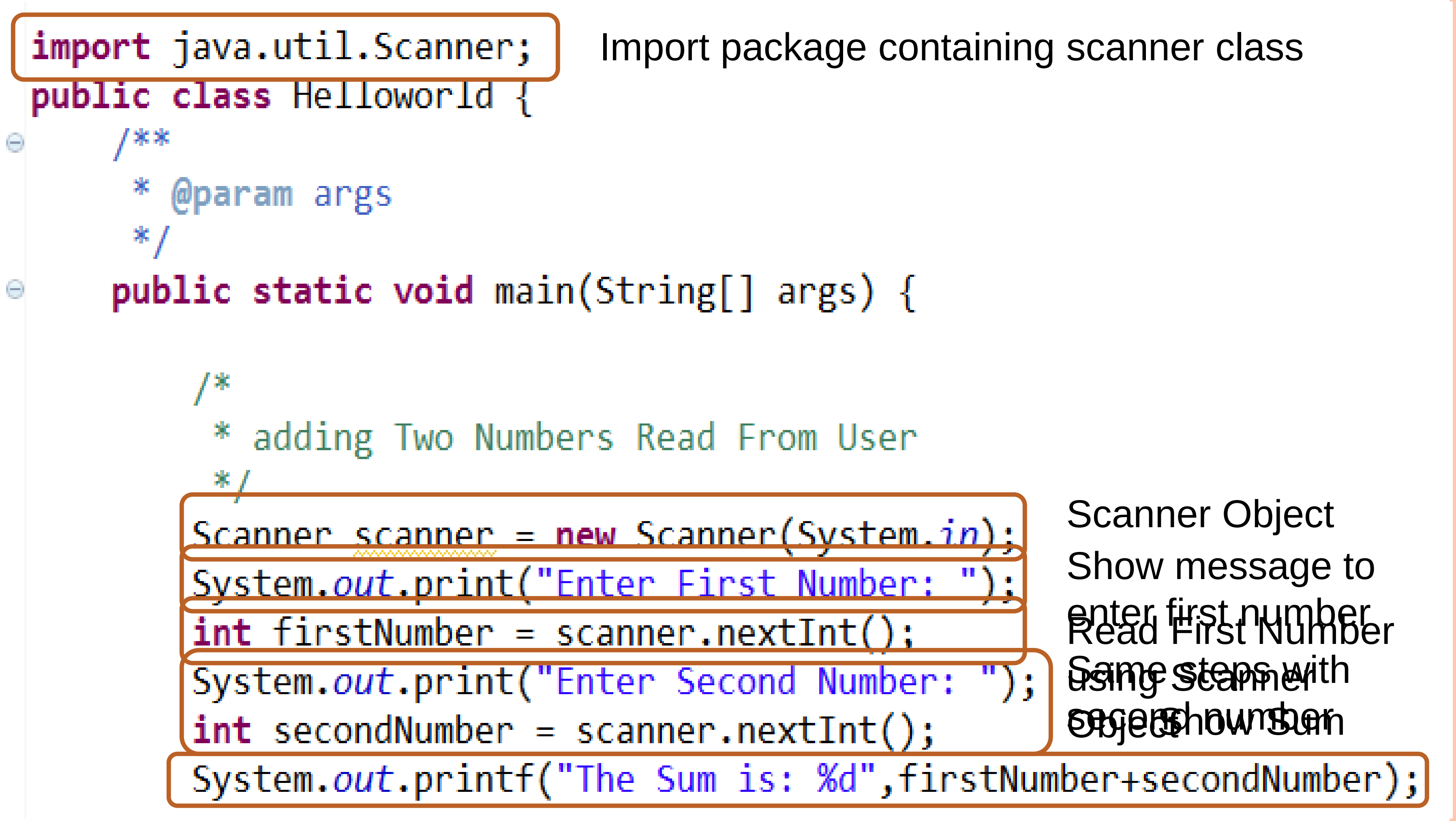
  ***e.g.*** *int x=input.nextInt();*

  *float f= input.nextFloat();*

  *String s= input.next();*

  *char c = input.next().charAt(0);*

## EXERCISE 2: ADDING TWO NUMBERS READ FROM USER



```
import java.util.Scanner;
public class Helloworld {
    /**
     * @param args
     */
    public static void main(String[] args) {

        /*
         * adding Two Numbers Read From User
         */
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter First Number: ");
        int firstNumber = scanner.nextInt();
        System.out.print("Enter Second Number: ");
        int secondNumber = scanner.nextInt();
       System.out.printf("The Sum is: %d",firstNumber+secondNumber);
```

Import package containing scanner class

Scanner Object

Show message to enter first number

Read First Number using Scanner Object

Same steps with second number

Show Sum

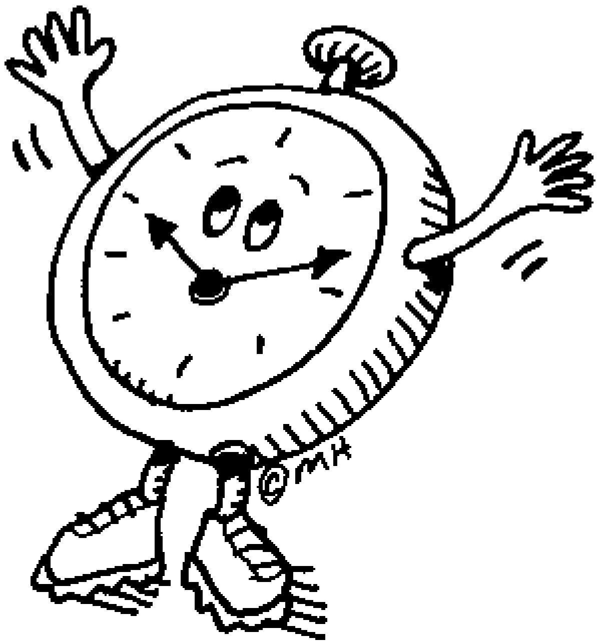
## Hands on #1: Quadratic Equation

- Consider the following quadratic equation:

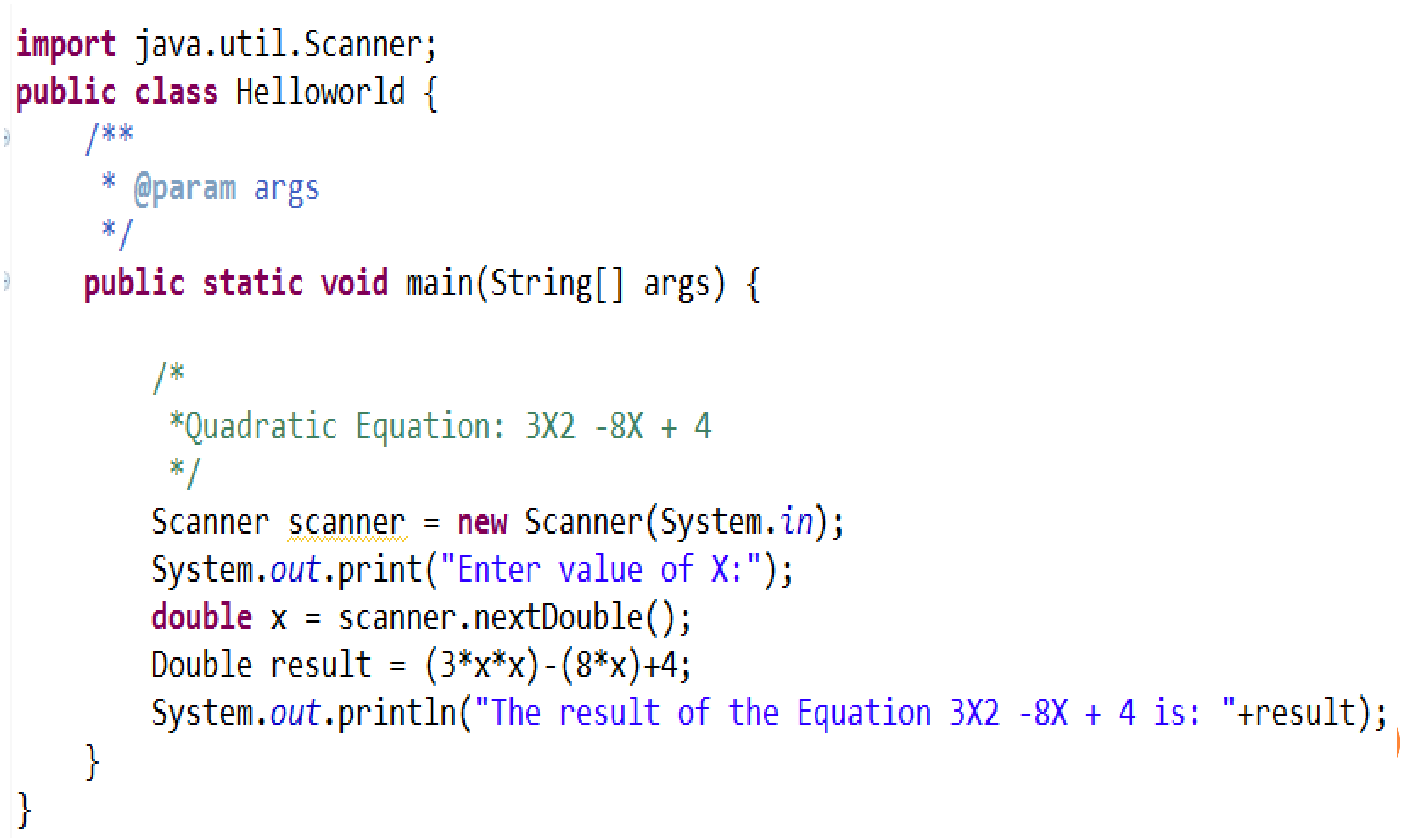
  **$3X^2 - 8X + 4$**

- Write a program that reads X from user and shows result.
  - Try the following values
    - X=2 the result will be zero.
    - X=200 the result will be 118404.
    - X=1 the result will be -1.

**<u>10 Minutes</u>**



## SOLUTION



```java
import java.util.Scanner;
public class Helloworld {
    /**
     * @param args
     */
    public static void main(String[] args) {

        /*
         *Quadratic Equation: 3X2 -8X + 4
         */
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter value of X:");
        double x = scanner.nextDouble();
        Double result = (3*x*x)-(8*x)+4;
        System.out.println("The result of the Equation 3X2 -8X + 4 is: "+result);
    }
}
```

## Hands on #2: Temperature Converter

- Write a program to convert temperature from Fahrenheit to Celsius and vice versa.

- From Fahrenheit to Celsius :

  Celsius = ( ( Fahrenheit - 32 ) * 5 / 9 )

- From Celsius to Fahrenheit :

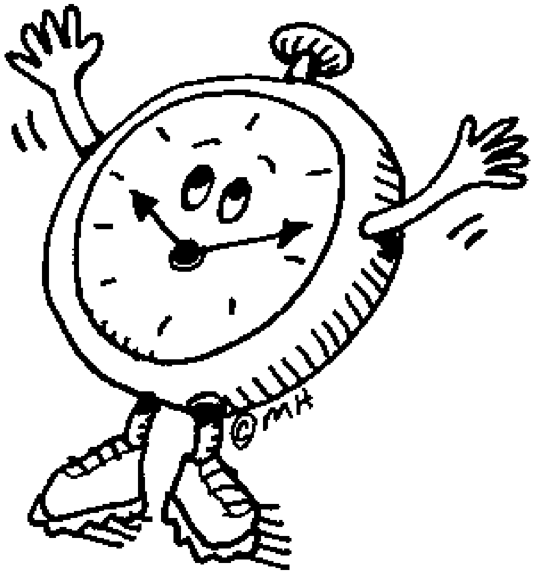
  Fahrenheit = ( ( Celsius * 9 ) / 5 ) + 32

## Hands on #2: Temperature Converter

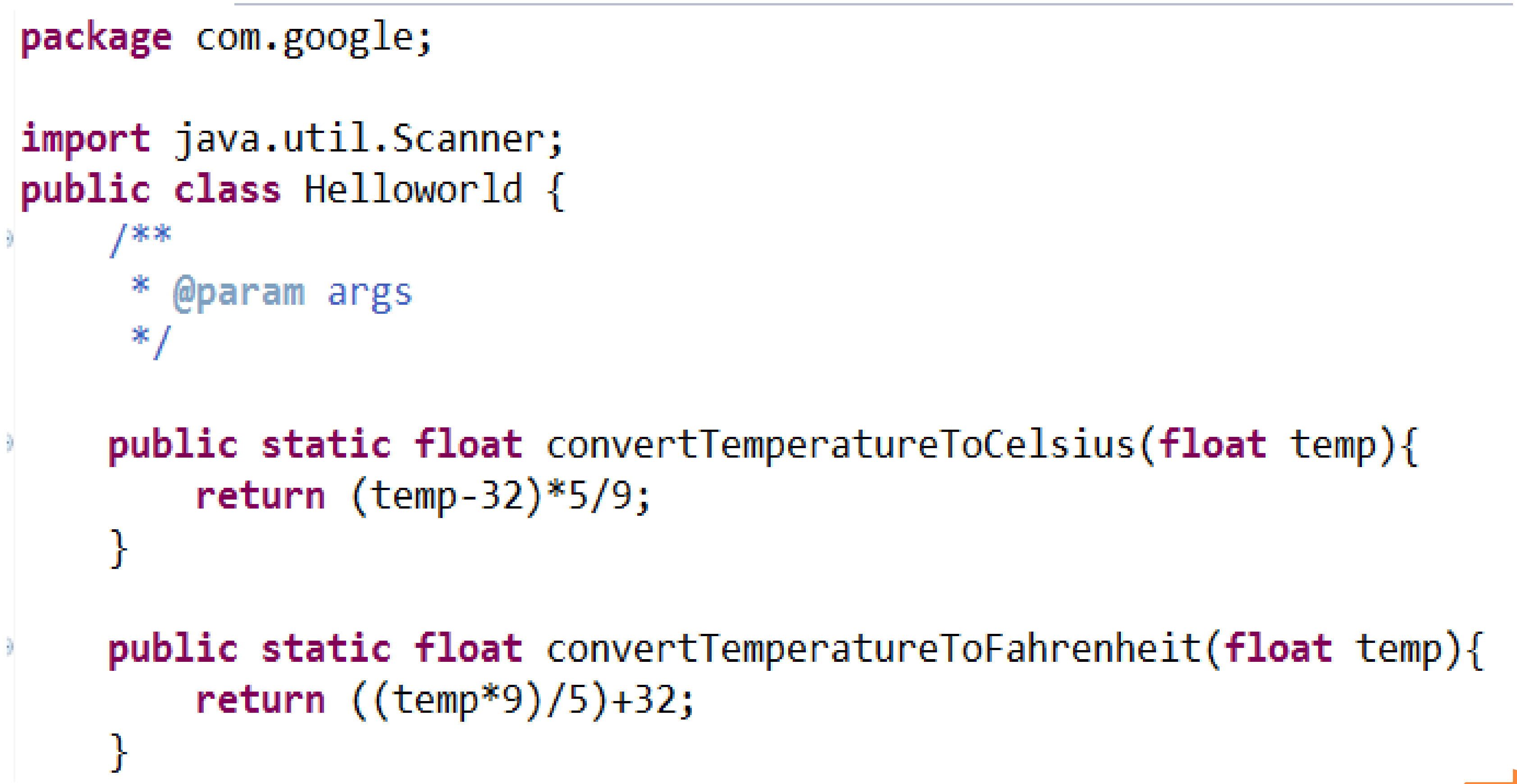
- Implement two methods (functions) for conversion.
- Read Temperature and type to convert to from user.
- Display converted temperature .

- Test:
  - Enter (26) and convert it to Fahrenheit which will be (78.8)

**<u>20 Minutes</u>**

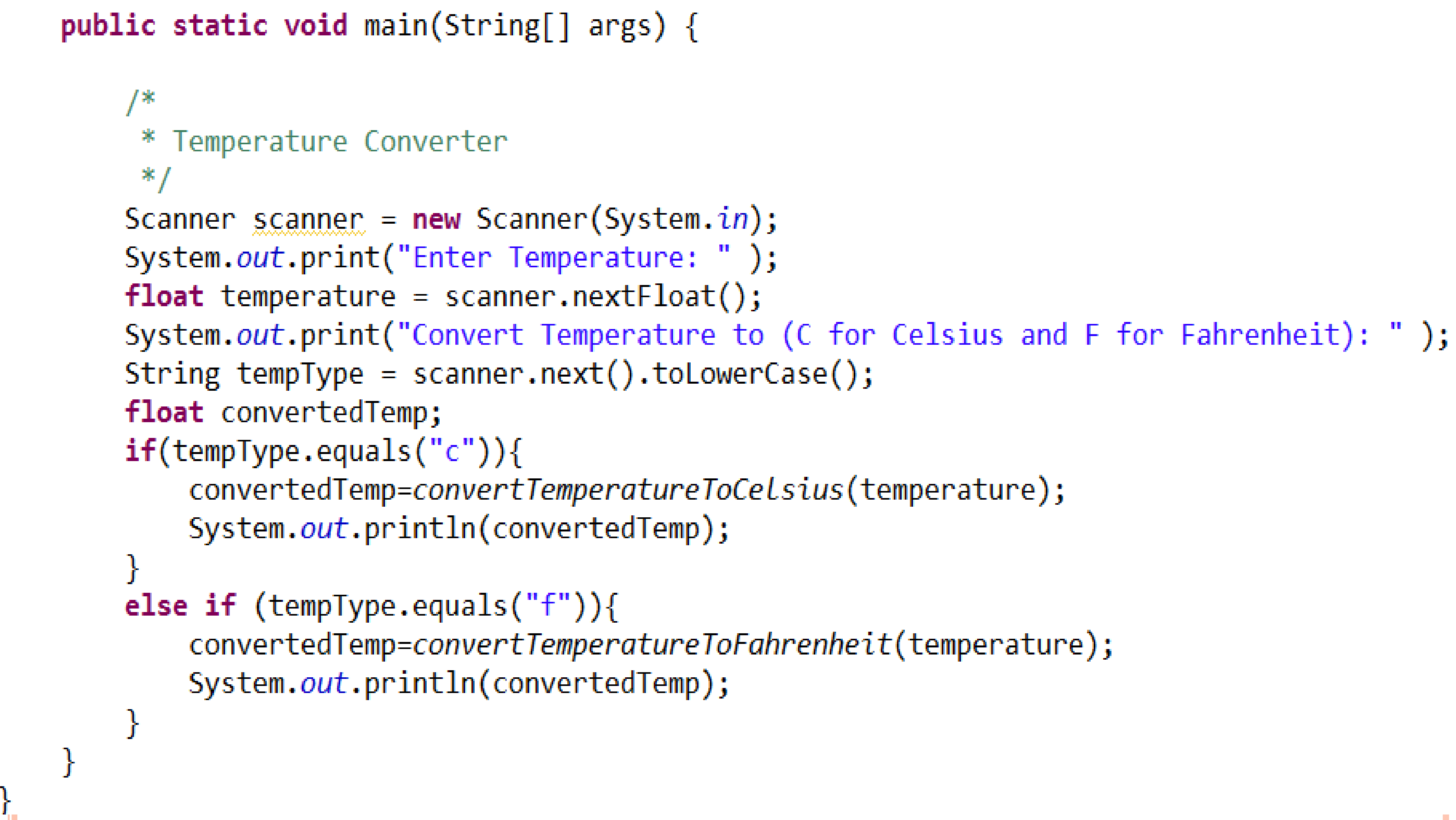


## SOLUTION



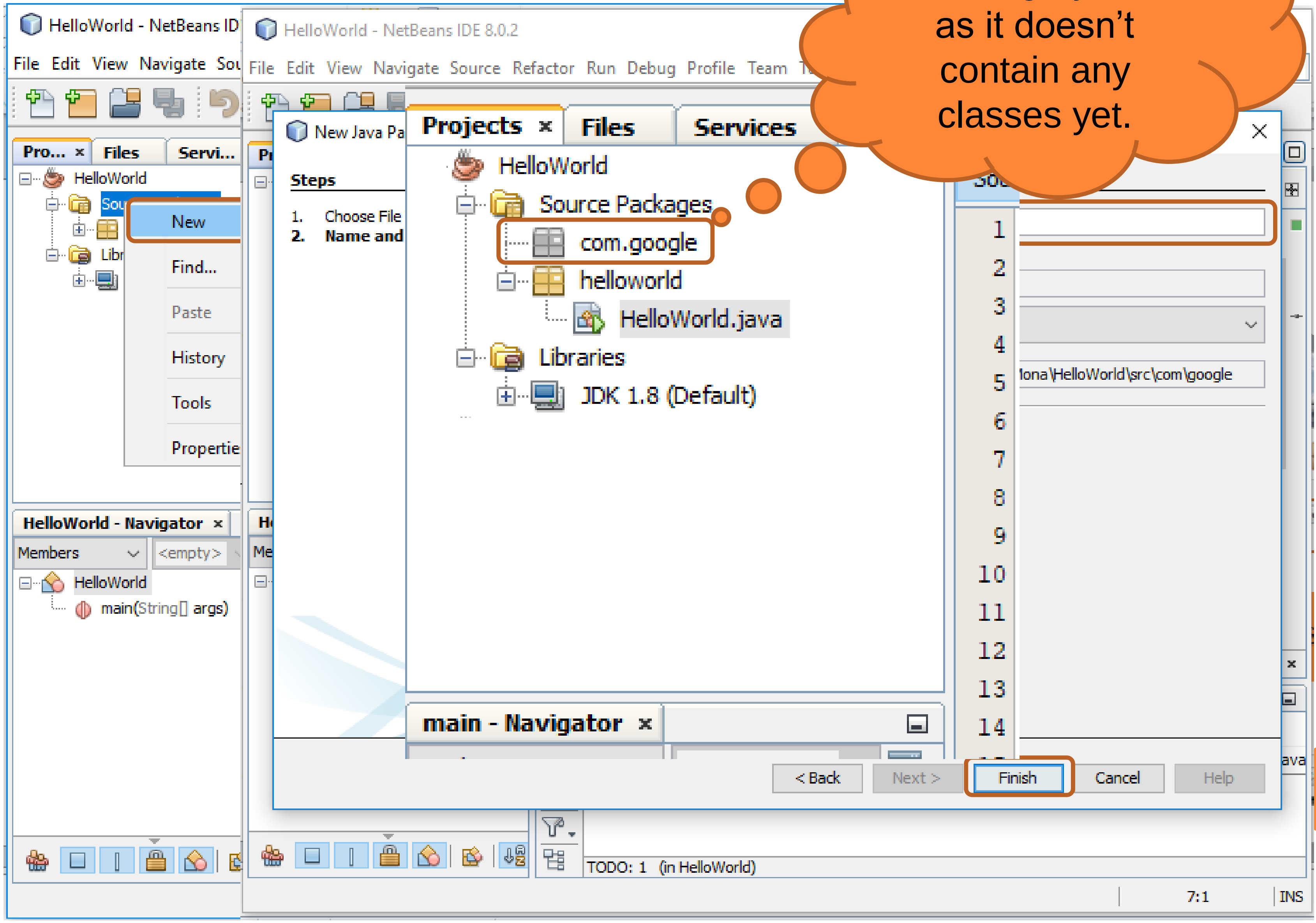
```
package com.google;

import java.util.Scanner;
public class Helloworld {
    /**
     * @param args
     */

    public static float convertTemperatureToCelsius(float temp){
        return (temp-32)*5/9;
    }

    public static float convertTemperatureToFahrenheit(float temp){
        return ((temp*9)/5)+32;
    }
```

## SOLUTION CONT’



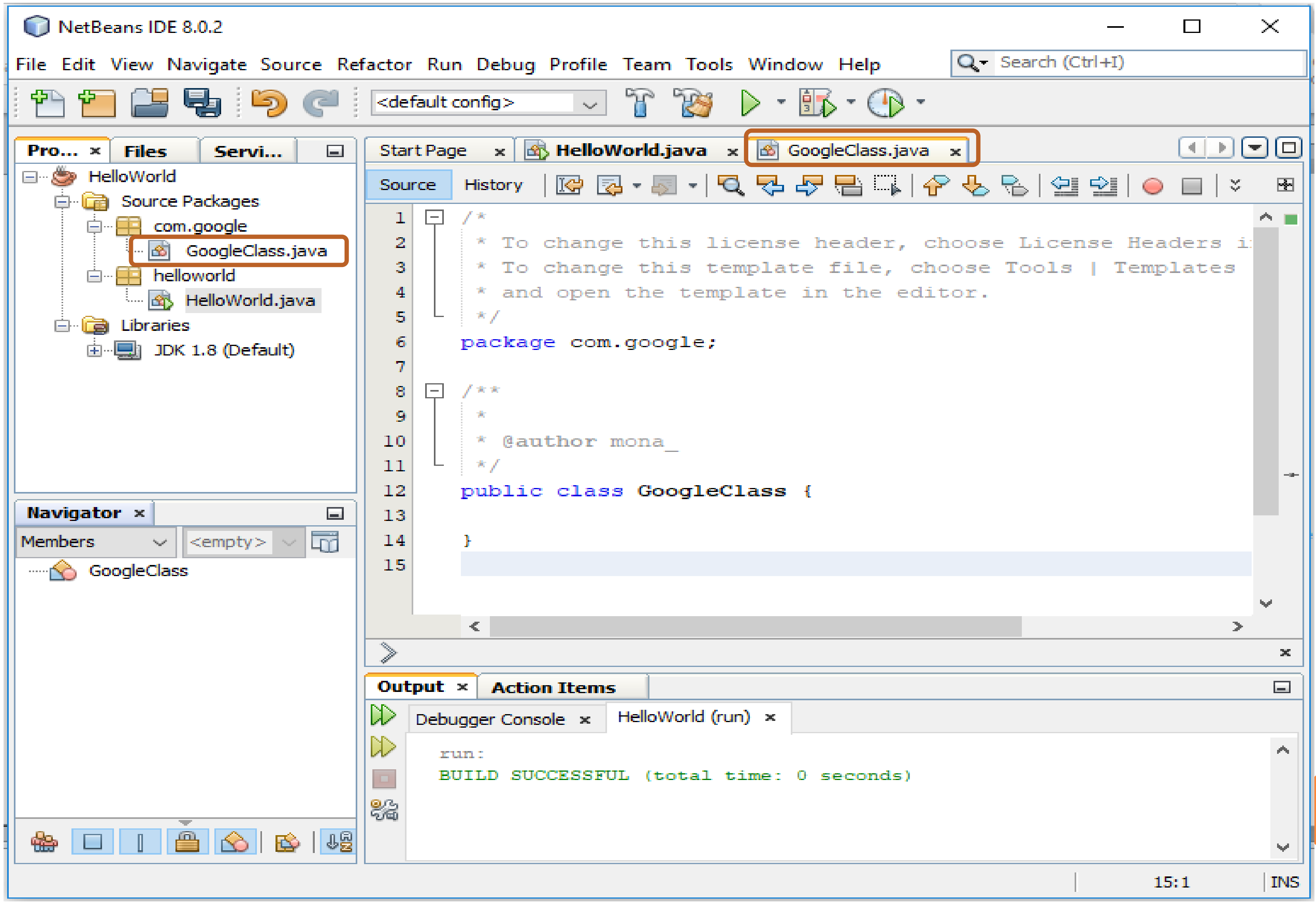
```java
    public static void main(String[] args) {

        /*
         * Temperature Converter
         */
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter Temperature: " );
        float temperature = scanner.nextFloat();
        System.out.print("Convert Temperature to (C for Celsius and F for Fahrenheit): " );
        String tempType = scanner.next().toLowerCase();
        float convertedTemp;
        if(tempType.equals("c")){
            convertedTemp=convertTemperatureToCelsius(temperature);
            System.out.println(convertedTemp);
        }
        else if (tempType.equals("f")){
            convertedTemp=convertTemperatureToFahrenheit(temperature);
            System.out.println(convertedTemp);
        }
    }
}
```

## Creating Packages



It has a gray color as it doesn't contain any classes yet.

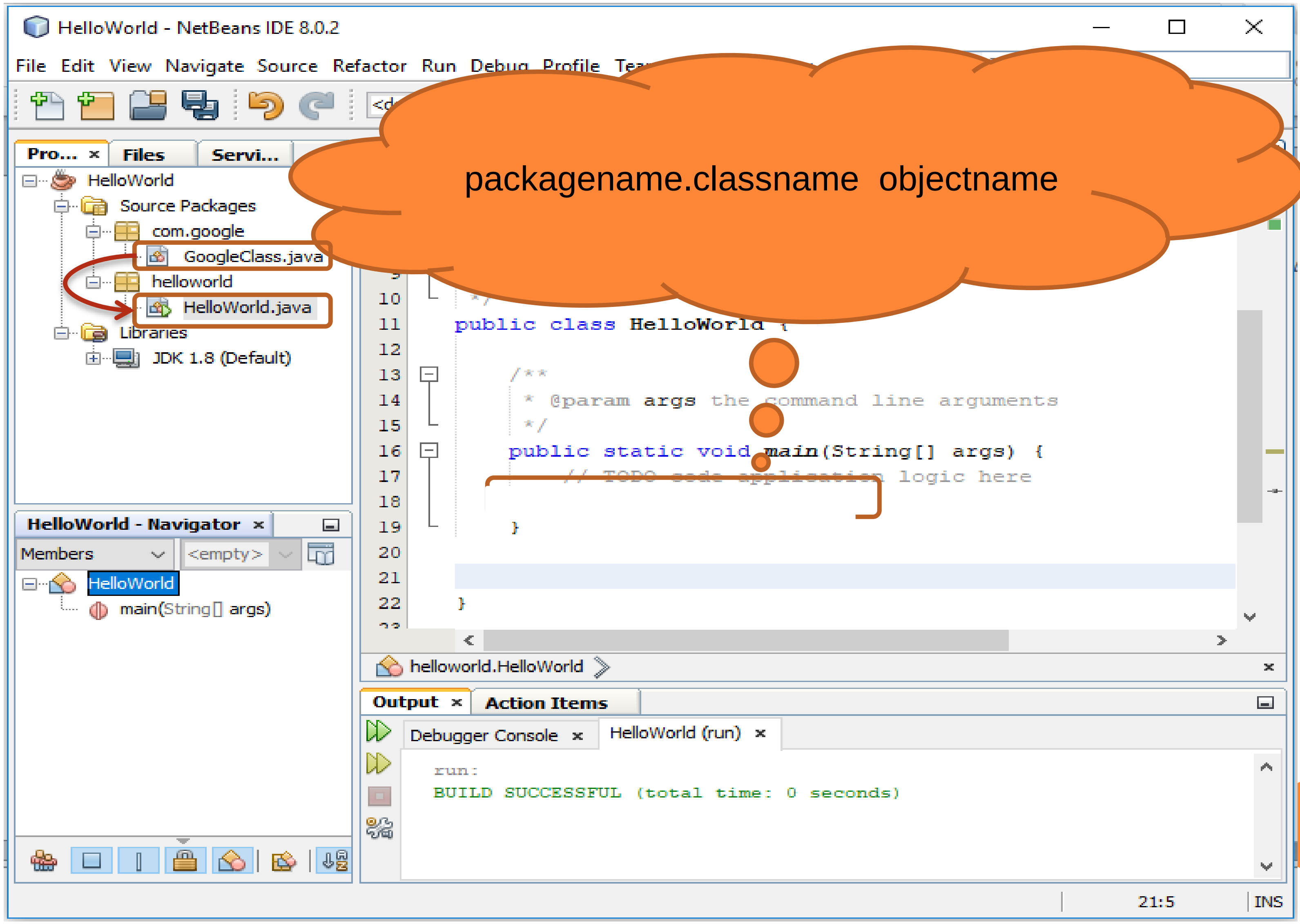
# ADDING CLASSES INTO COM.GOOGLE PACKAGE



```
/*
 * To change this license header, choose License Headers i
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package com.google;

/**
 *
 * @author mona_
 */
public class GoogleClass {

}

```

```
run:
BUILD SUCCESSFUL (total time: 0 seconds)
```

# Using public class from another package (1)

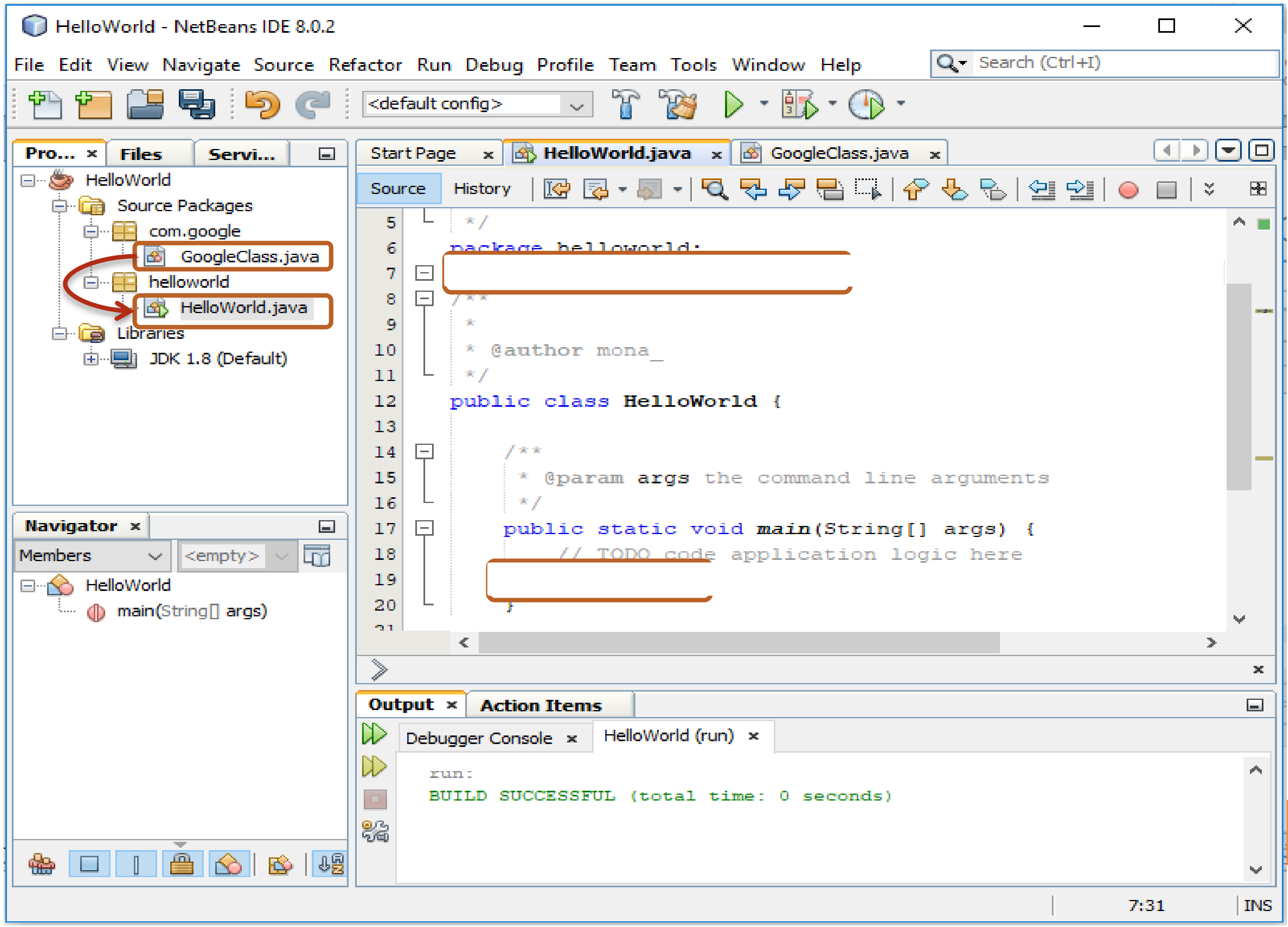


packagename.classname objectname

```
    */
public class HelloWorld {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here

    }


}
```

```
run:
BUILD SUCCESSFUL (total time: 0 seconds)
```

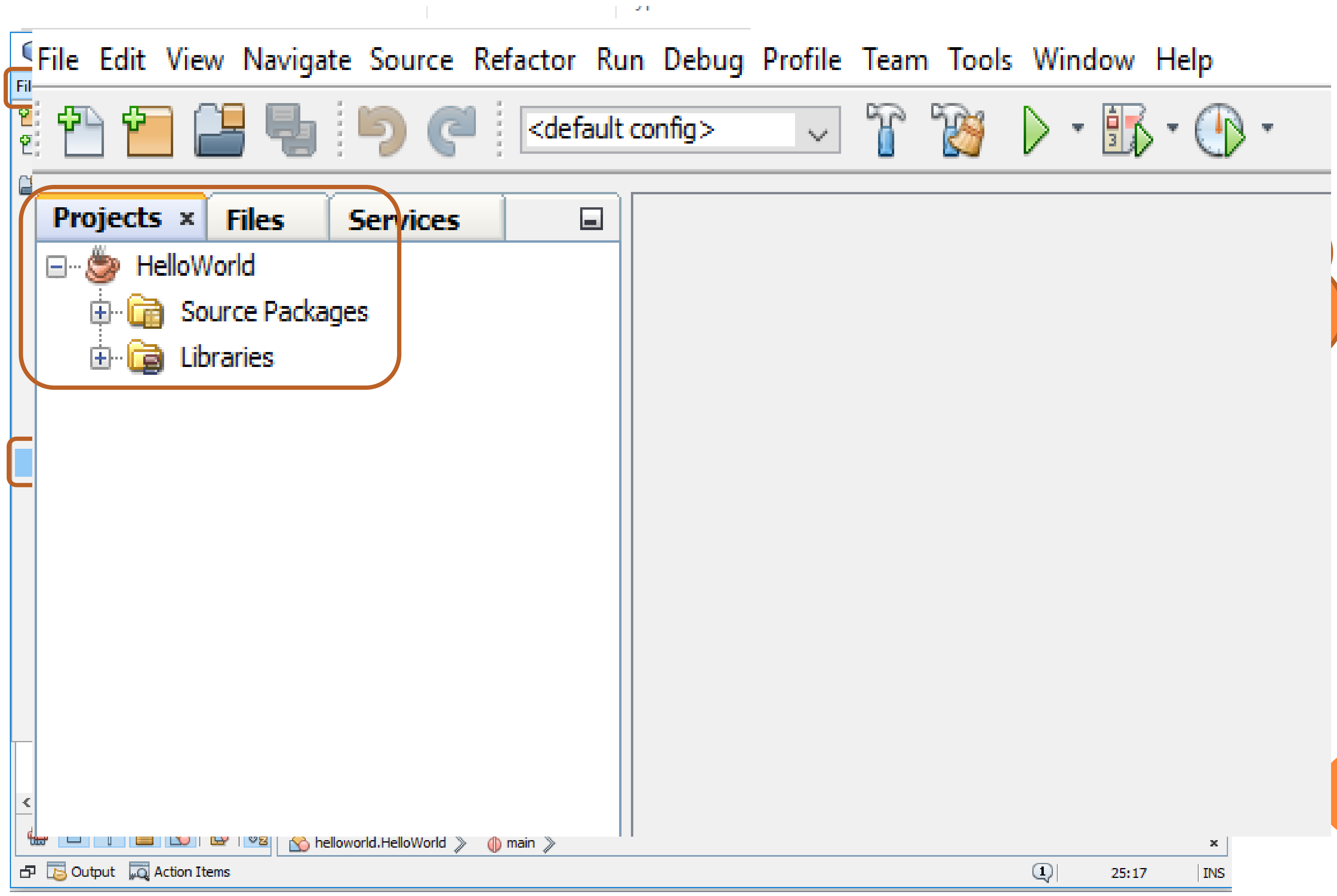
## Using public class from another package (2)



```
 */
package helloworld;

/**
 *
 * @author mona_
 */
public class HelloWorld {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here

    }
```

```
run:
BUILD SUCCESSFUL (total time: 0 seconds)
```

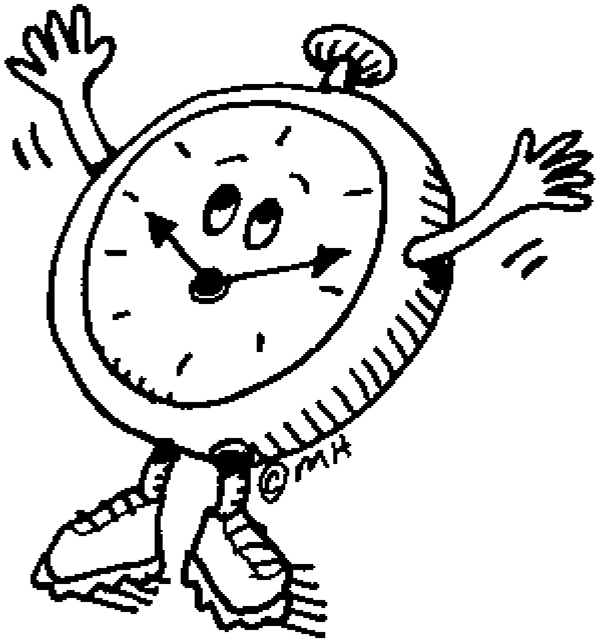
# Importing projects into NetBeans



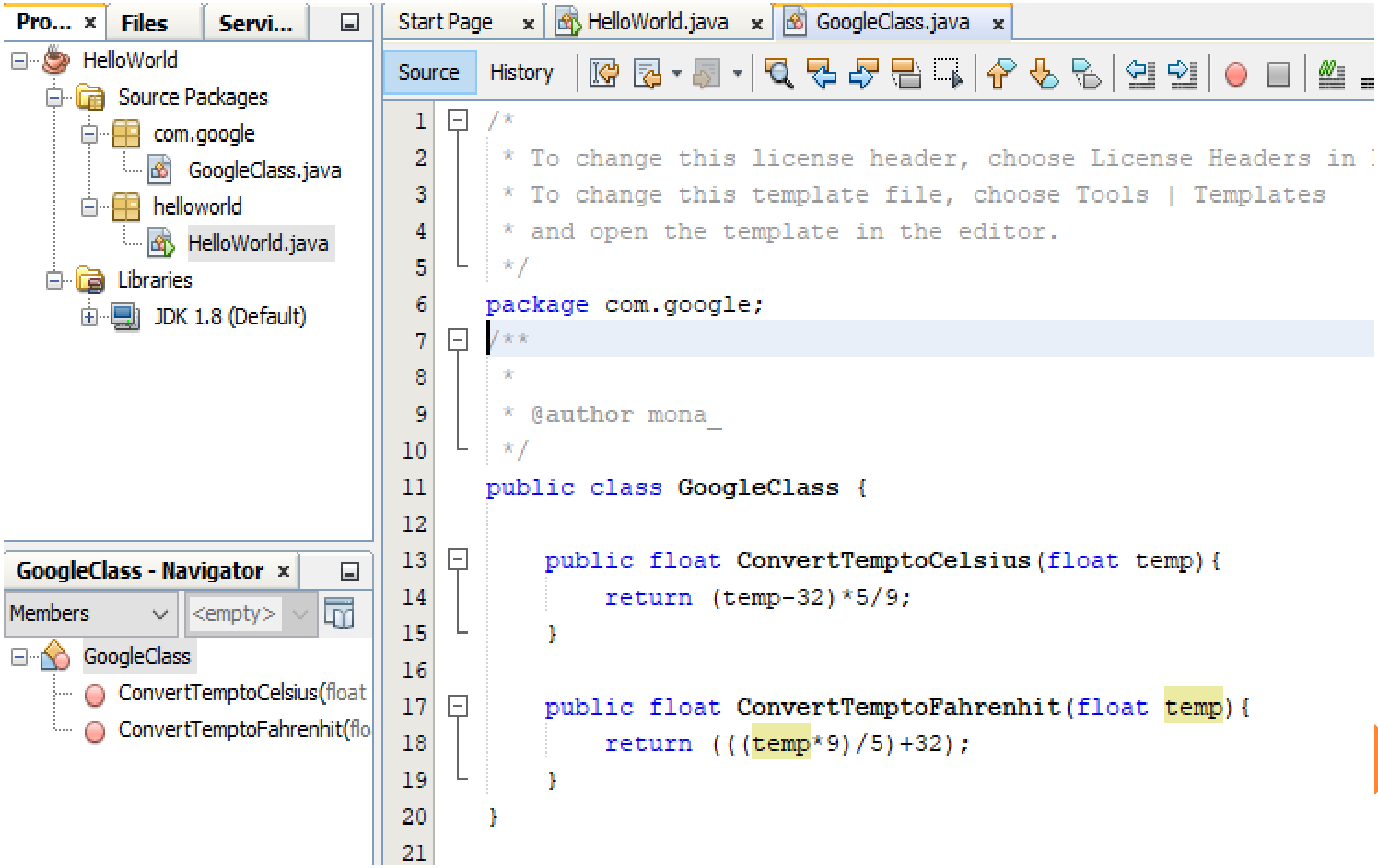
## Hands on #3: Temperature Converter (2)

- Now, try to update your solution of the last problem and use different classes in different packages.

**<u>10 Minutes</u>**

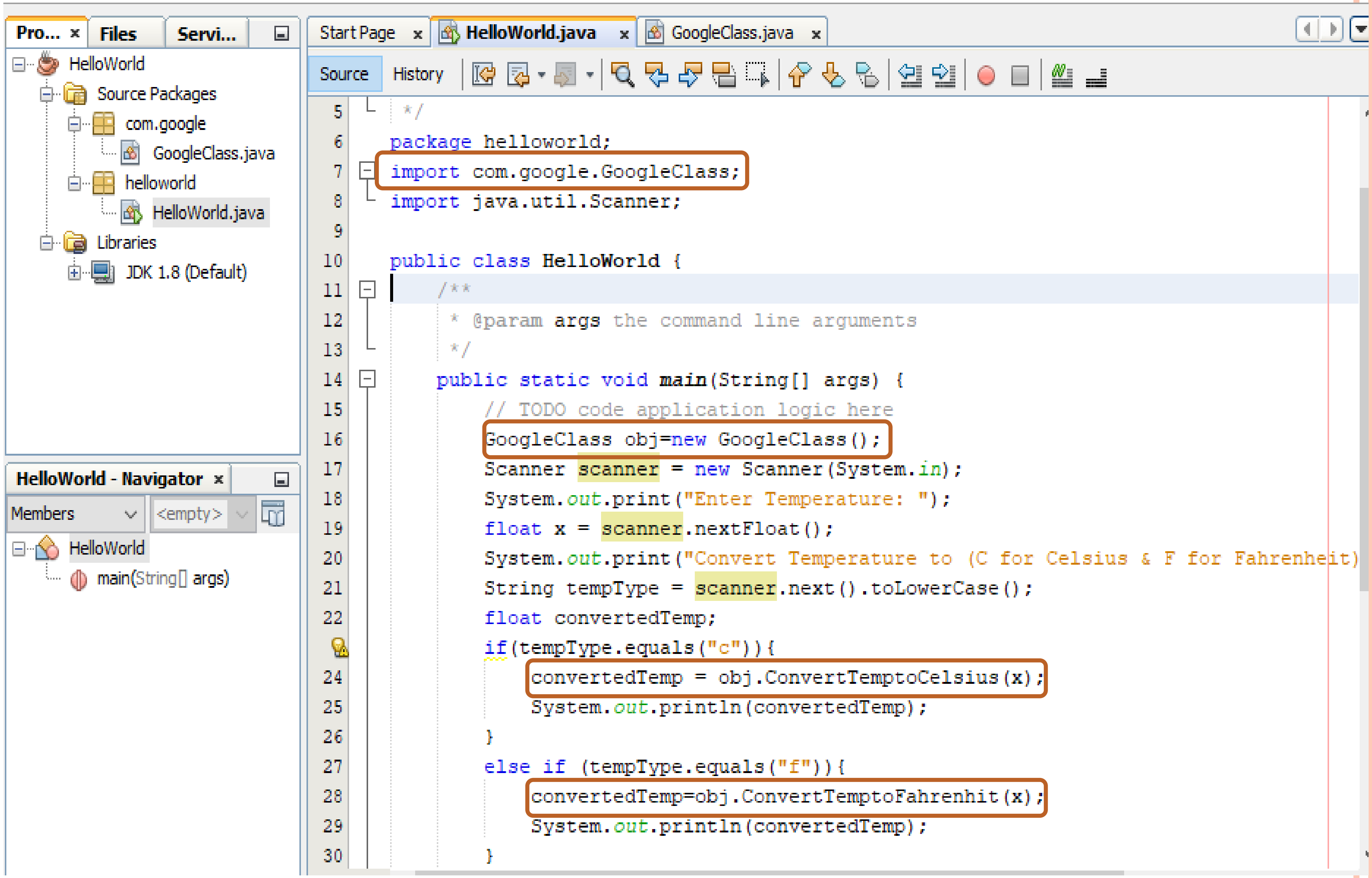


## Solution: Using classes of different packages



```java
/*
 * To change this license header, choose License Headers in
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package com.google;
/**
 *
 * @author mona_
 */
public class GoogleClass {

    public float ConvertTemptoCelsius(float temp){
        return (temp-32)*5/9;
    }

    public float ConvertTemptoFahrenhit(float temp){
        return (((temp*9)/5)+32);
    }
}
```

# SOLUTION: USING CLASSES OF DIFFERENT PACKAGES



```java
     */
package helloworld;
import com.google.GoogleClass;
import java.util.Scanner;

public class HelloWorld {
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        GoogleClass obj=new GoogleClass();
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter Temperature: ");
        float x = scanner.nextFloat();
        System.out.print("Convert Temperature to (C for Celsius & F for Fahrenheit)
        String tempType = scanner.next().toLowerCase();
        float convertedTemp;
        if(tempType.equals("c")){
            convertedTemp = obj.ConvertTemptoCelsius(x);
            System.out.println(convertedTemp);
        }
        else if (tempType.equals("f")){
            convertedTemp=obj.ConvertTemptoFahrenhit(x);
            System.out.println(convertedTemp);
        }
```

## QUESTIONS