



OBJECT ORIENTED PROGRAMMING USING JAVA

Lab 1

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

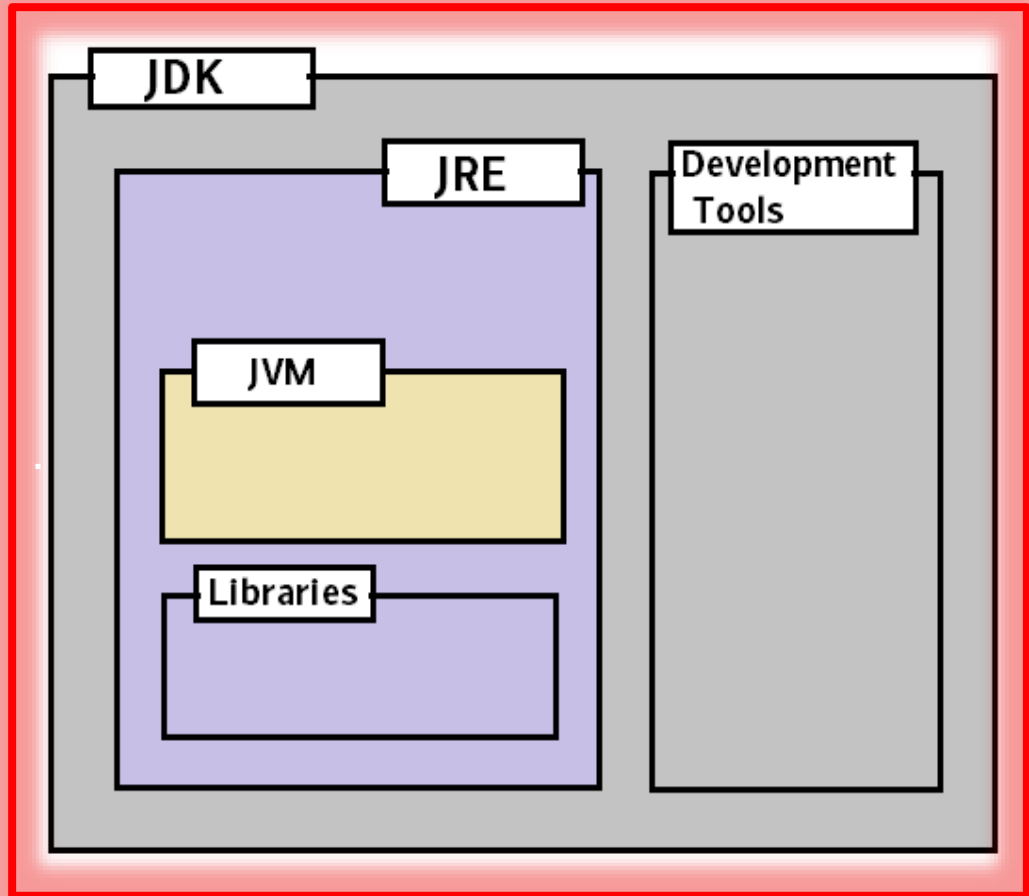
1. **Installing JDK:** in order to install NetBeans you need to first install JDK.
2. **NetBeans installation:** run the installation application source for version 8.0.2.

INSTALLATION

○ What is a JDK ?

The JDK includes

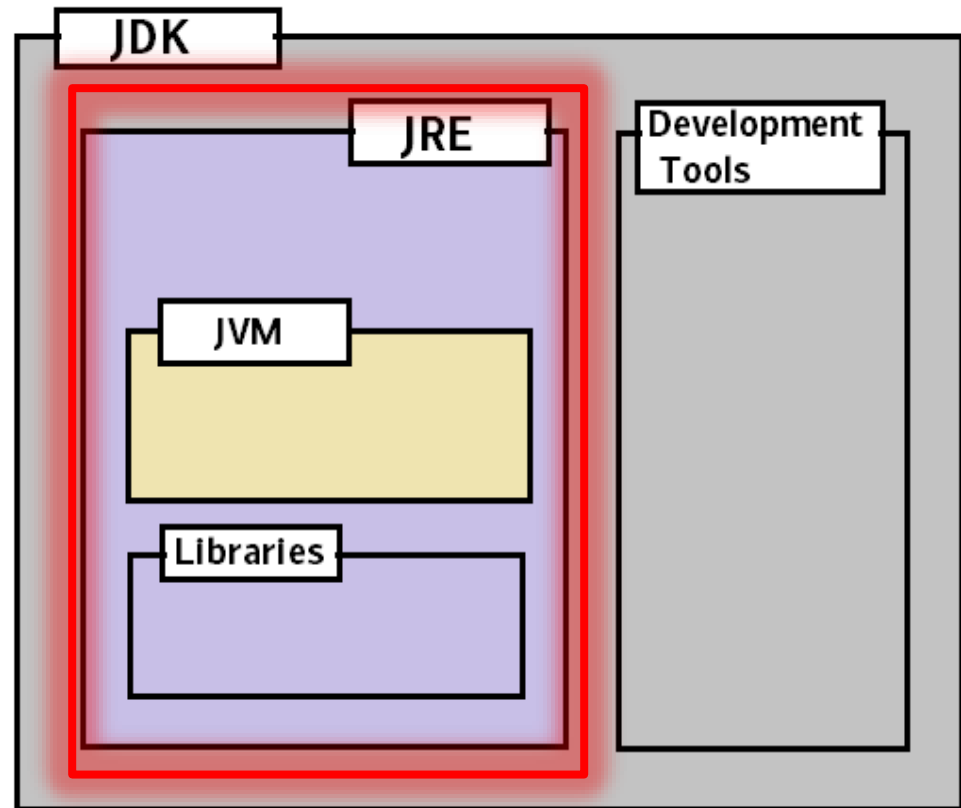
- a set of tools for compiling and running your java code
- “Java Runtime Environment” JRE .



INSTALLATION

- **What is a JRE ?**

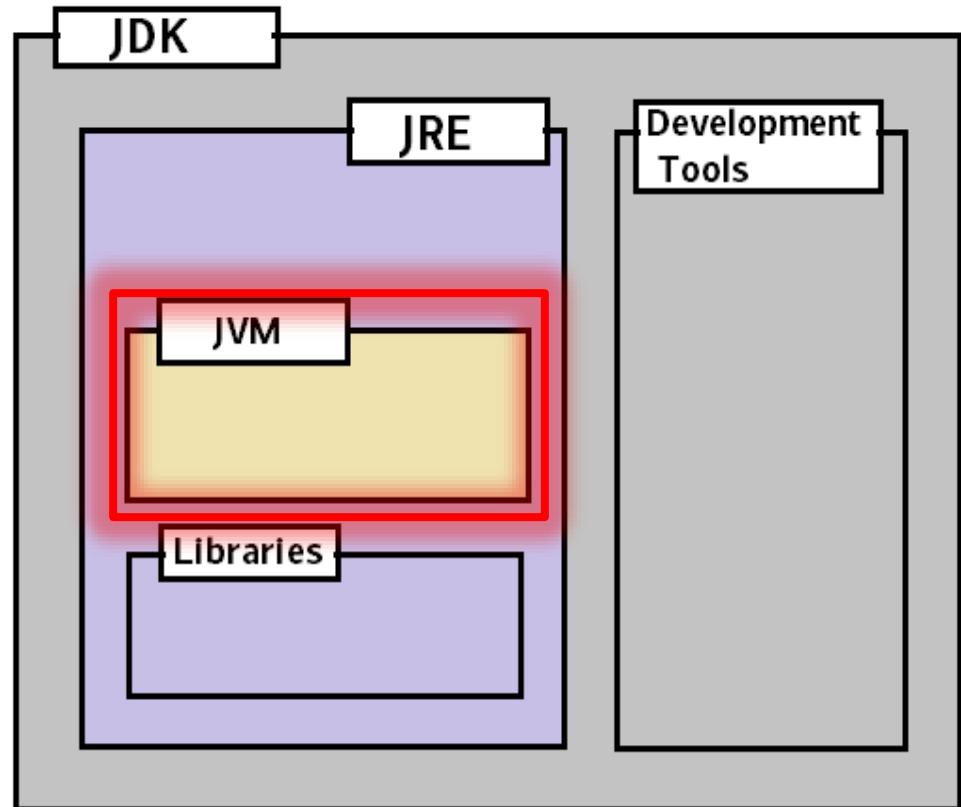
1. includes the JVM
2. code libraries that are necessary for running programs



INSTALLATION

- **What is a JVM ?**

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

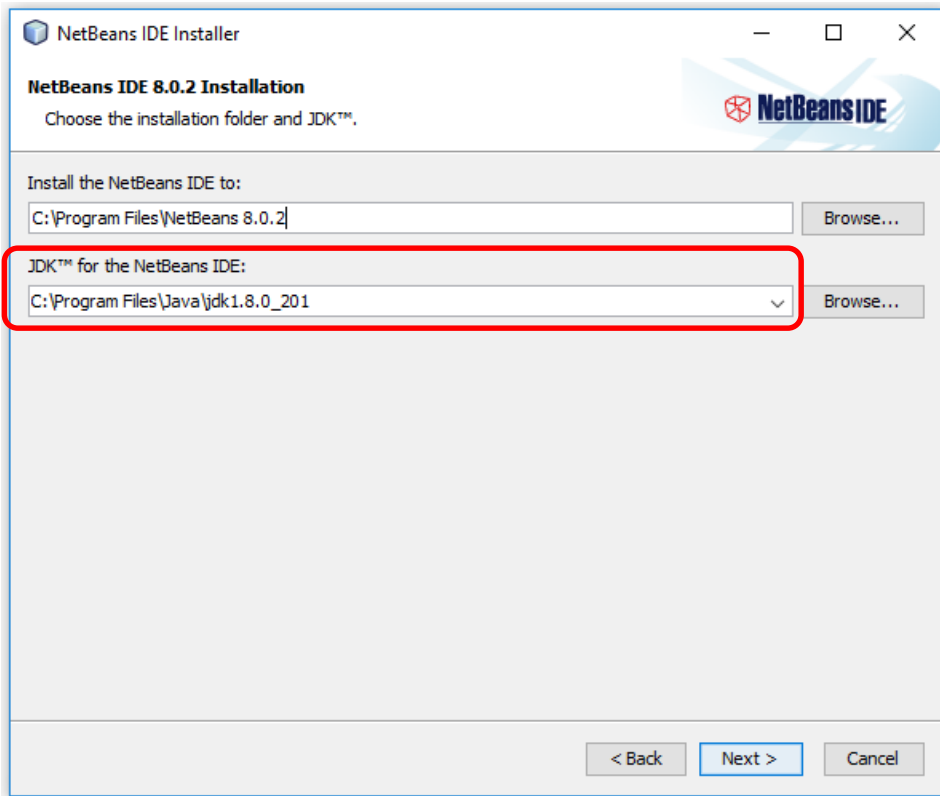


INSTALLATION

- 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.

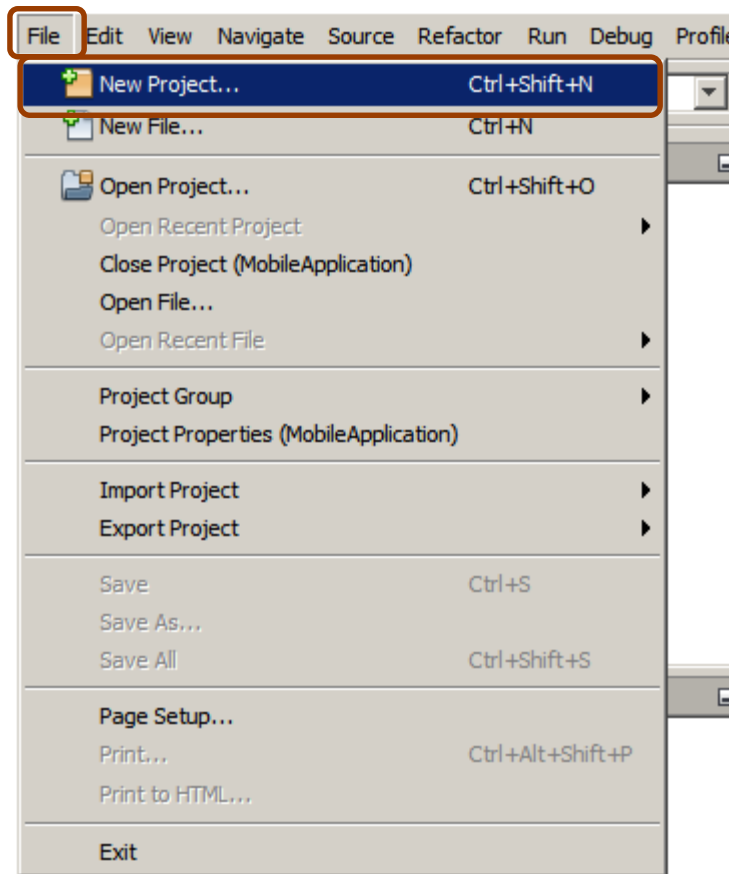
INSTALLATION

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



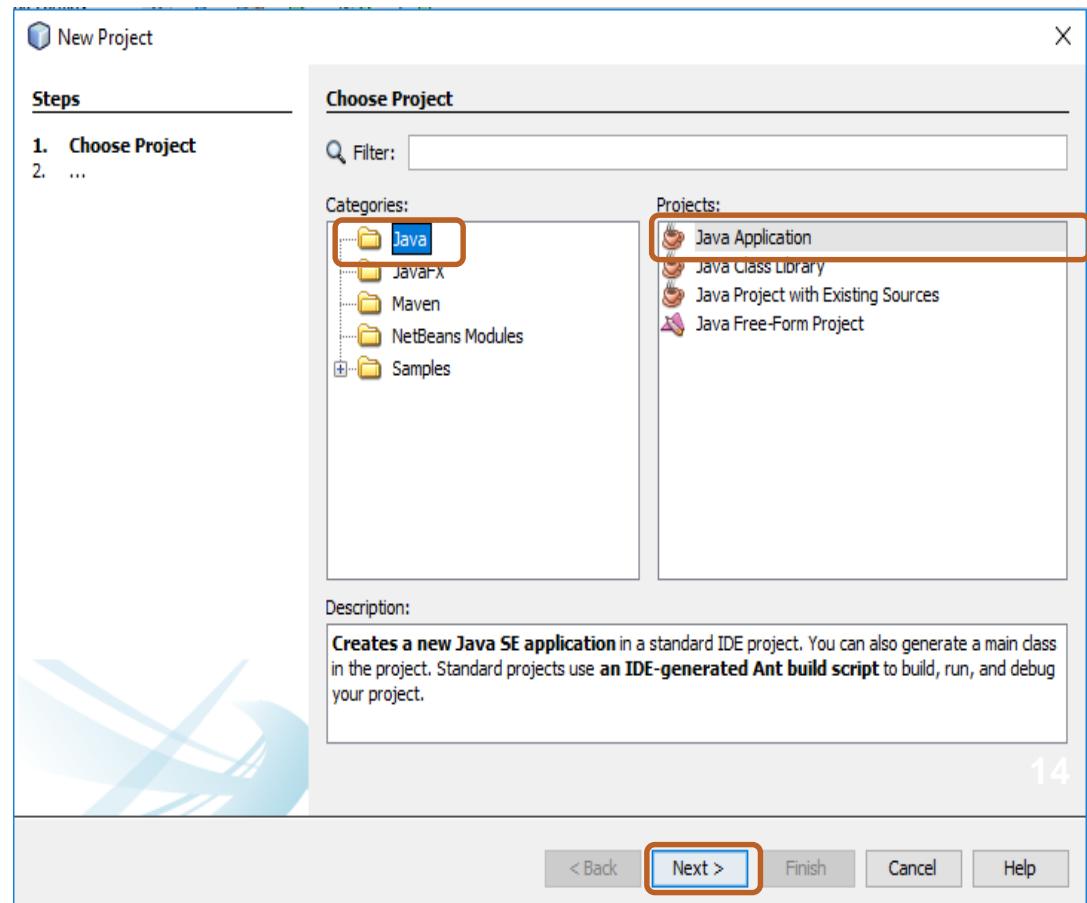
QUICK START GUIDE

- 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.



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.HelloWorld App (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 : Browse...

Project Folder: :\CIS\Object Oriented Programming\OOP 2[

Use Dedicated Folder for Storing Libraries

Libraries Folder: Browse...

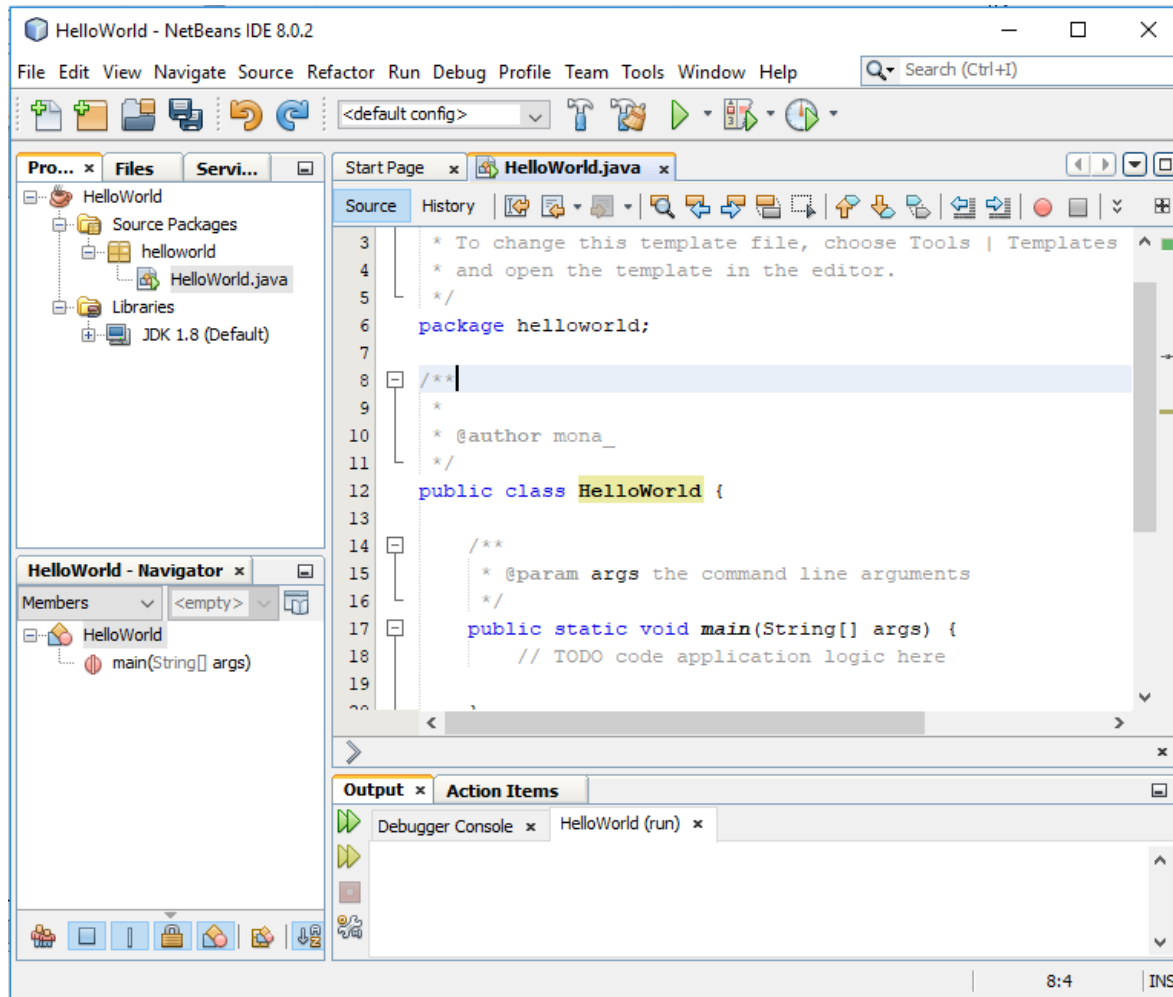
Different users and projects can share the same compilation libraries (see Help for details).

Create Main Class helloworld1.HelloWorld1

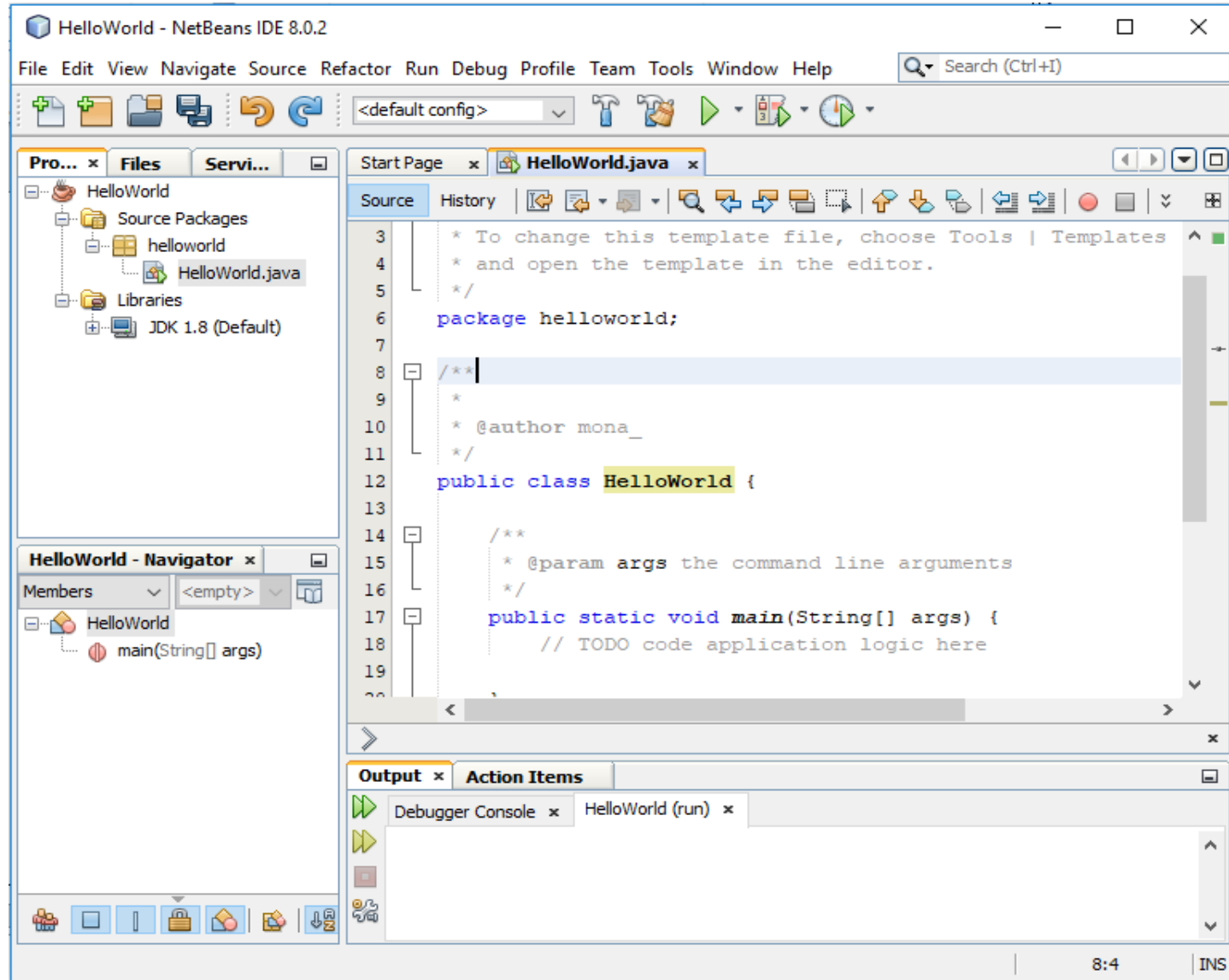
< Back Next > Finish Cancel Help

QUICK START GUIDE

- The project is created and opened in the IDE.



QUICK START GUIDE



QUICK START GUIDE

The screenshot displays the NetBeans IDE interface for a project named "HelloWorld". The main window is titled "HelloWorld - NetBeans IDE 8.0.2" and features a menu bar (File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help) and a search bar (Search (Ctrl+I)). The toolbar includes icons for file operations and development actions. The "Files" window on the left shows a tree view of the project structure, including "Source Packages", "helloworld", "HelloWorld.java", and "Libraries" (JDK 1.8 (Default)). The "HelloWorld - Navigator" window at the bottom left shows the "main(String[] args)" method. The central editor window displays the source code for "HelloWorld.java", which includes a class declaration and a main method. The code is as follows:

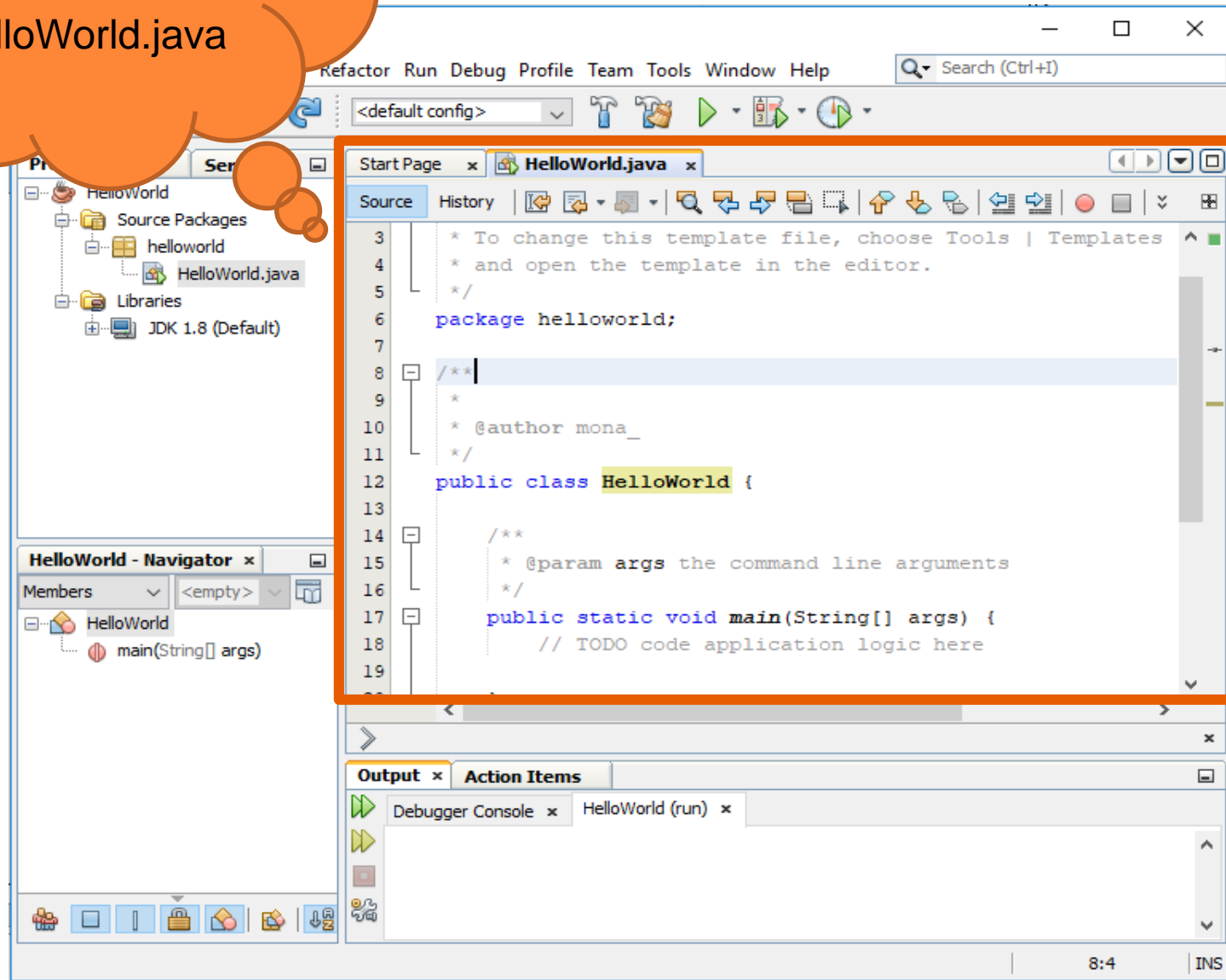
```
3  * To ch
4  * and o
5  *
6  pe he
7
8  /**
9  *
10 * @author mona_
11 */
12 public class HelloWorld {
13
14     /**
15      * @param args the command line arguments
16      */
17     public static void main(String[] args) {
18         // TODO code application logic here
19
20     }
```

An orange callout bubble with a cloud-like shape contains the text: "A project window which contains a tree view of the components of the project".

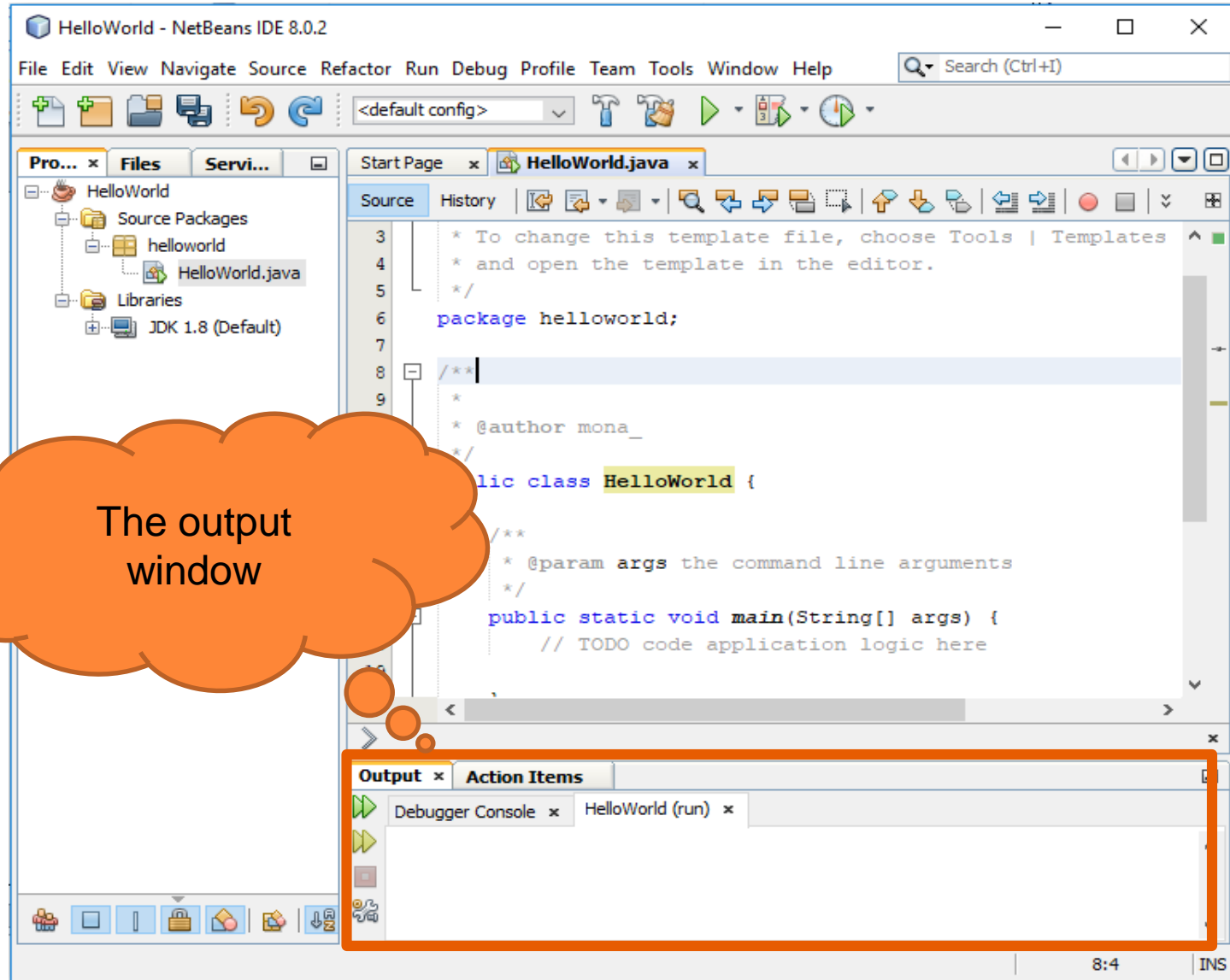
The "Output" window at the bottom right shows "Debugger Console" and "HelloWorld (run)". The status bar at the bottom indicates the time "8:4" and the cursor position "INS".

The source editor
with class
HelloWorld.java

HIDE



QUICK START GUIDE



QUICK START GUIDE

The screenshot shows the NetBeans IDE interface. The main editor displays the source code for `HelloWorld.java`. A TODO comment is visible: `// TODO code application logic here`. The Action Items window is open at the bottom, showing a table with one entry for the TODO comment.

The Action Items window contains the following table:

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

At the bottom of the Action Items window, it indicates: `TODO: 1 (in HelloWorld)`

The Action Items window for viewing error list

QUICK START GUIDE

Class name

```
package helloworld;

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

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

Output

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

TODO: 1 (in HelloWorld)

8:4 | INS

QUICK START GUIDE

NetBeans IDE 8.0.2 window showing the source code of `HelloWorld.java`. The code is as follows:

```
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5
6  package helloworld;
7
8  /**
9   *
10  * @author mona_
11  */
12  public class HelloWorld {
13
14      /**
15       * @param args the command line arguments
16       */
17      public static void main(String[] args) {
18          // TODO code application logic here
19      }
20  }
```

The line `package helloworld;` is highlighted with an orange box. An orange callout bubble points to this line with the text "Package name".

The Output window at the bottom shows the following table:

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

At the bottom of the IDE, it shows "TODO: 1 (in HelloWorld)" and the status bar indicates "8:4 | INS".

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", str);`

3. `System.out.printf("%d",`

4. `int x=10;`

`System.out.printf("%s=",`

5. `int x=1000;`

`System.out.printf("%d", x);` → **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
<code>\n</code>	Newline. Position the screen cursor at the beginning of the next line.
<code>\t</code>	Horizontal tab. Move the screen cursor to the next tab stop.
<code>\r</code>	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.
<code>\\</code>	Backslash. Used to print a backslash character.
<code>\"</code>	Double quote. Used to print a double-quote character. For example, <pre>System.out.println("\"in quotes\"");</pre> 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);  
  
    }  
}
```

Problems @ Javadoc Declaration Console Call Hierarchy
<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 !!

The screenshot displays the NetBeans IDE 8.0.2 interface. The title bar reads "HelloWorld - NetBeans IDE 8.0.2". The menu bar includes "File", "Edit", "View", "Navigate", "Source", "Refactor", "Run", "Debug", "Profile", "Team", "Tools", "Window", and "Help". A search bar on the right contains "Search (Ctrl+I)". The toolbar features various icons, with the Run button (a green play icon) highlighted by a red circle.

The left sidebar shows the "Project Explorer" with a tree view of the "HelloWorld" project. It includes "Source Packages" (com.google, helloworld), "Libraries" (JDK 1.8 (Default)), and "HelloWorld.java". Below it is the "main - Navigator" showing the "main(String[] args)" method.

The central "Source" window displays the code for "HelloWorld.java":

```
9
10  * @author mona_
11  */
12  public class HelloWorld {
13
14      /**
15       * @param args the command line arguments
16       */
17      public static void main(String[] args) {
18          // TODO code application logic here
19          int x = 1500000;
20          double y=1000.525435;
21          String mrX="X";
22          char currency='$';
23          System.out.printf(" %s Said: \"I have %d %c\" \n",mrX,x,currency);
24          System.out.printf(" Y Saind: \"ok Mr\\%s,I have %f %c\" ",mrX,y,currency);
25      }
26
27
```

The bottom "Output" window shows the execution results:

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

The status bar at the bottom right indicates "26:1" and "INS".

HOW TO DEBUG !!

NetBeans IDE 8.0.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+I)

Start Page x HelloWorld.java x GoogleClass.java x HelloWorld.java x

```
9
10 * @author mona_
11 */
12 public class HelloWorld {
13
14     /**
15      * @param args the command line arguments
16      */
17     public static void main(String[] args) {
18         // TODO code application logic here
19         int x = 1500000;
20         double y=1000.525435;
21         String mrX="X";
22         char currency='$';
23         System.out.printf(" %s Said: \"I have %d %c\" \n",mrX,x,currency);
24         System.out.printf(" Y Saind: \"ok Mr\\%s,I have %f %c\" ",mrX,y,currency);
25     }
```

helloworld.HelloWorld → main →

Variables x	Breakpoints	Output	Action Items
Name		Type	Value
<input type="text" value="<Enter new watch>"/>			
Static			
args		String[]	#85(length=0)
x		int	1500000
y		double	1000.525435

HelloWorld (debug) | running... | 21:1 | INS

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 {
```

Import package containing scanner class

```
    /**  
     * @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);
```

Scanner Object

Show message to

enter first number

Read First Number

Same steps with

second number

Object 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.

10 Minutes



SOLUTION

```
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 :

$$\text{Celsius} = ((\text{Fahrenheit} - 32) * 5 / 9)$$

- From Celsius to Fahrenheit :

$$\text{Fahrenheit} = ((\text{Celsius} * 9) / 5) + 32$$

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)

20 Minutes



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'

```
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

The screenshot shows the NetBeans IDE interface with the 'New Java Package' dialog box open. The dialog has a 'Steps' section with two steps: '1. Choose File' and '2. Name and Location'. The 'Name and Location' step is active, showing a tree view of the project structure. The 'com.google' package is selected and highlighted in gray. Below the tree view, the path 'tona\HelloWorld\src\com\google' is visible. The 'Finish' button is highlighted with a red box. An orange callout bubble points to the 'com.google' package with the text: 'It has a gray color as it doesn't contain any classes yet.'

Steps

1. Choose File
2. Name and Location

com.google

tona\HelloWorld\src\com\google

Finish

ADDING CLASSES INTO COM.GOOGLE PACKAGE

The screenshot displays the NetBeans IDE 8.0.2 interface. The main editor window shows the source code for `GoogleClass.java`, which is highlighted with an orange box. The code includes a package declaration and a class definition:

```
1  /*
2  * To change this license header, choose License Headers i
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6  package com.google;
7
8  /**
9   *
10  * @author mona_
11  */
12  public class GoogleClass {
13
14  }
15
```

The left sidebar shows the project structure with `com.google` and `GoogleClass.java` highlighted. The bottom panel shows the Output window with the message: `BUILD SUCCESSFUL (total time: 0 seconds)`. The status bar at the bottom indicates the cursor is at line 15, column 1.

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)

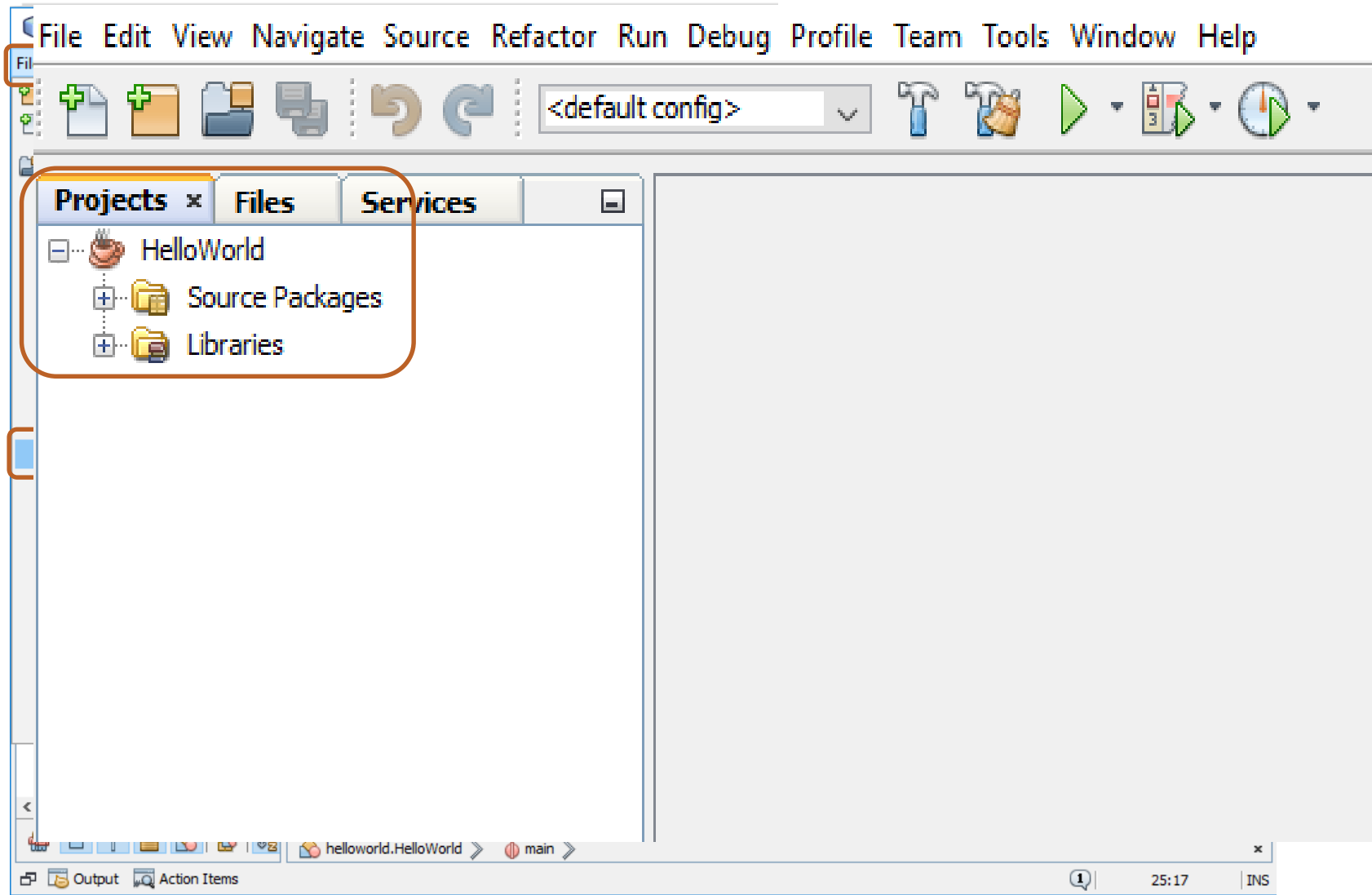
USING PUBLIC CLASS FROM ANOTHER PACKAGE (2)

NetBeans IDE 8.0.2 window showing a project named "HelloWorld". The interface includes a menu bar (File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help), a toolbar, and several panels:

- Files:** Shows the project structure. The "Source Packages" folder contains "com.google" and "helloworld". "GoogleClass.java" is located in "com.google", and "HelloWorld.java" is in "helloworld". Red arrows indicate the relationship between these files.
- Navigator:** Shows the "Members" of the "HelloWorld" class, including the "main(String[] args)" method.
- Source:** Displays the code for "HelloWorld.java". The code includes a package declaration, a class declaration, and a main method. Red boxes highlight the package declaration and the main method.
- Output:** Shows the result of running the program: "BUILD SUCCESSFUL (total time: 0 seconds)".

```
5  L  */
6  package helloworld;
7
8  /**
9   *
10  * @author mona_
11  */
12  public class HelloWorld {
13
14      /**
15       * @param args the command line arguments
16       */
17      public static void main(String[] args) {
18          // TODO code application logic here
19
20      }
21
```

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.

10 Minutes



SOLUTION: USING CLASSES OF DIFFERENT PACKAGES

The screenshot displays an IDE interface with the following components:

- Project Explorer (Left):** Shows a project named "HelloWorld" with a "Source Packages" folder containing "com.google" (with "GoogleClass.java") and "helloworld" (with "HelloWorld.java"). A "Libraries" folder contains "JDK 1.8 (Default)".
- GoogleClass - Navigator (Bottom Left):** Shows the "Members" of the "GoogleClass" class: "ConvertTempToCelsius(float)" and "ConvertTempToFahrenheit(float)".
- Source Editor (Right):** Shows the source code of "GoogleClass.java" with line numbers 1 through 21. The code includes a package declaration, a class declaration, and two methods: "ConvertTempToCelsius" and "ConvertTempToFahrenheit".

```
1  /*
2   * To change this license header, choose License Headers in
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6  package com.google;
7  /**
8   *
9   * @author mona_
10 */
11 public class GoogleClass {
12
13     public float ConvertTempToCelsius(float temp) {
14         return (temp-32)*5/9;
15     }
16
17     public float ConvertTempToFahrenheit(float temp) {
18         return (((temp*9)/5)+32);
19     }
20 }
21
```

SOLUTION: USING CLASSES OF DIFFERENT PACKAGES

The screenshot shows an IDE window with two tabs: `HelloWorld.java` and `GoogleClass.java`. The `HelloWorld.java` tab is active, displaying the following code:

```
5  /*
6  package helloworld;
7  import com.google.GoogleClass;
8  import java.util.Scanner;
9
10 public class HelloWorld {
11     /**
12      * @param args the command line arguments
13      */
14     public static void main(String[] args) {
15         // TODO code application logic here
16         GoogleClass obj=new GoogleClass();
17         Scanner scanner = new Scanner(System.in);
18         System.out.print("Enter Temperature: ");
19         float x = scanner.nextFloat();
20         System.out.print("Convert Temperature to (C for Celsius & F for Fahrenheit)");
21         String tempType = scanner.next().toLowerCase();
22         float convertedTemp;
23         if(tempType.equals("c")){
24             convertedTemp = obj.ConvertTempToCelsius(x);
25             System.out.println(convertedTemp);
26         }
27         else if (tempType.equals("f")){
28             convertedTemp=obj.ConvertTempToFahrenhit(x);
29             System.out.println(convertedTemp);
30     }
```

The IDE's Project Explorer on the left shows the project structure:

- Project: HelloWorld
- Source Packages:
 - com.google
 - GoogleClass.java
 - helloworld
 - HelloWorld.java
- Libraries: JDK 1.8 (Default)

The HelloWorld - Navigator on the bottom left shows the `main(String[] args)` method in the `HelloWorld` class.

QUESTIONS

