

Programming 1

Lesson 04 – First View of IDE

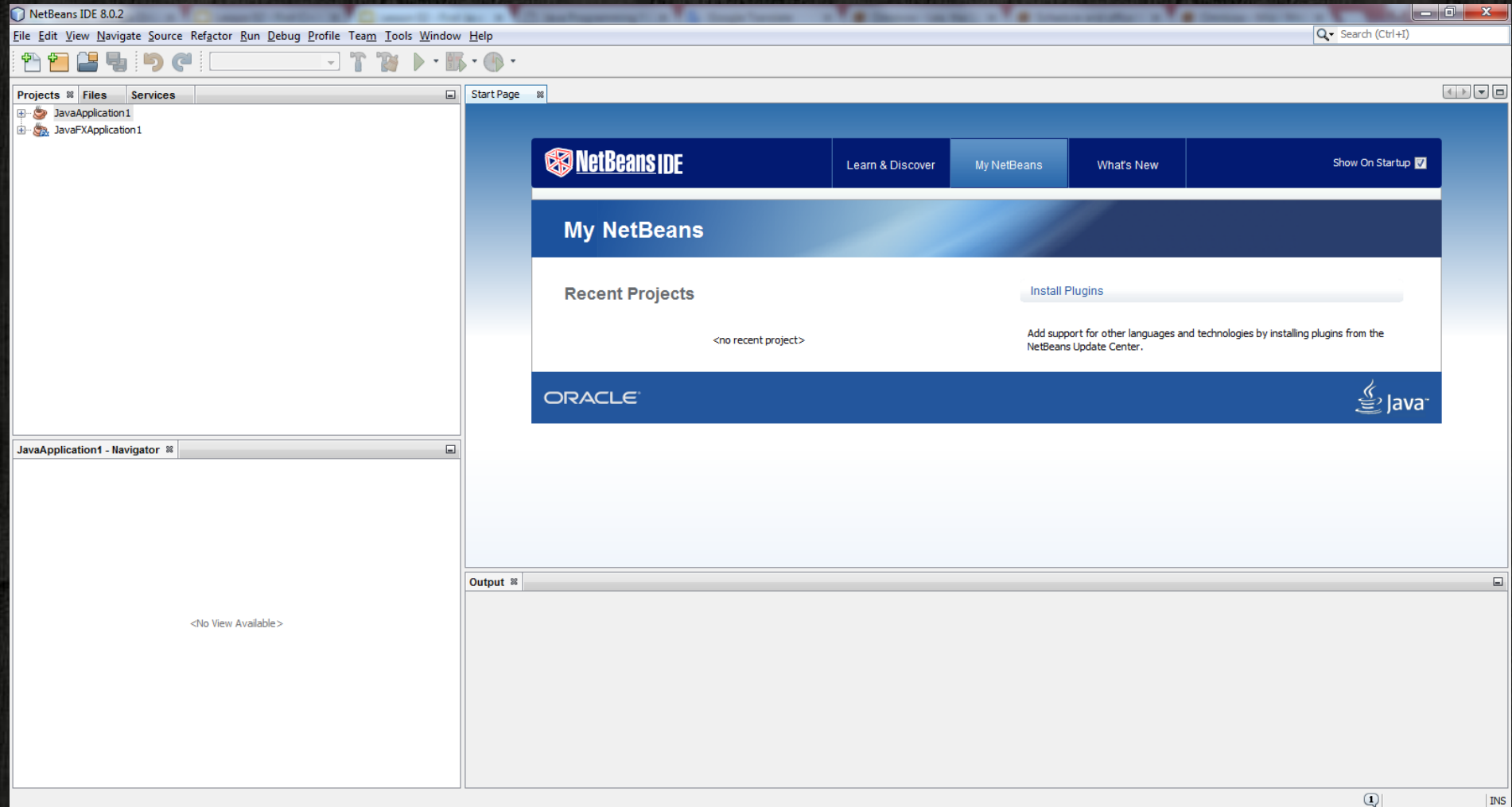


First View of IDE

NetBeans IDE

- ◆ NetBeans is a popular Java IDE, and today we will take a look at it.
- ◆ You can download NetBeans from its official website
- ◆ To open NetBeans, either you can create a shortcut of it on the desktop, or you can search for it in the “Start” menu.
- ◆ Usually it will take a few seconds to start an IDE since it takes more memory than text editor.

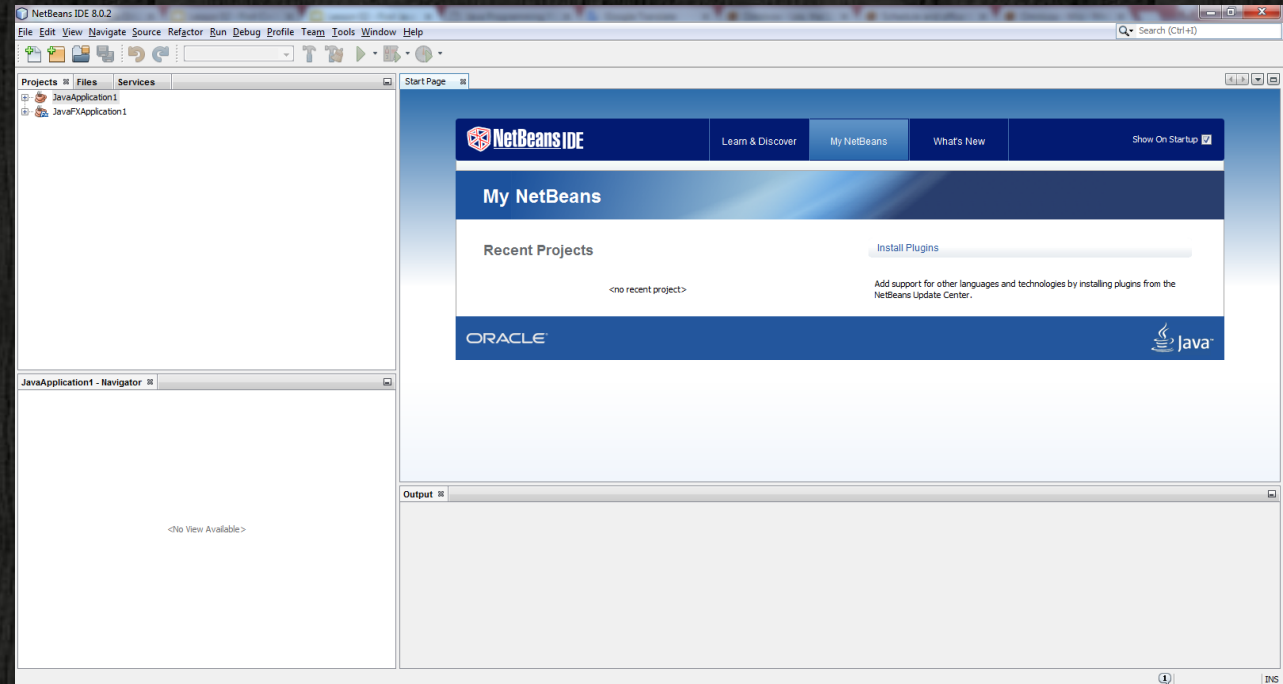
NetBeans Layout



NetBeans IDE

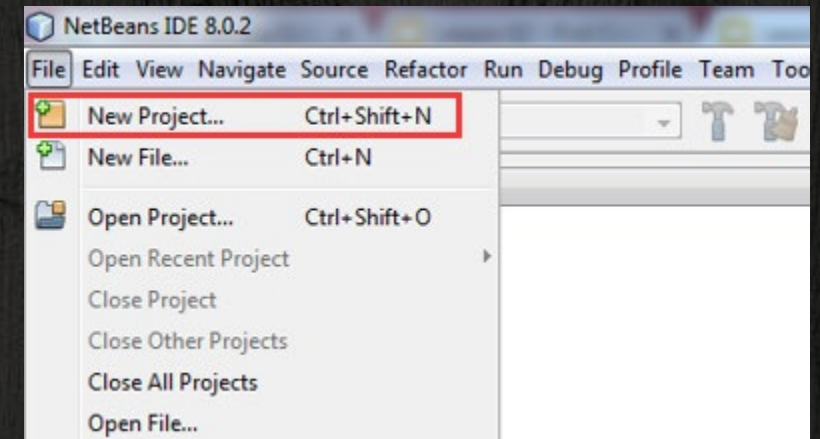
◆ Globally NetBeans IDE can be separated into 4 parts:

- Top left: **Project / File navigator**: You can check all the files related to projects.
- Bottom left: **Member navigator**: You can all the members (data members and methods) of each project.
- Top right: **Text editor**: This is the main window for you to write your Java code.
- Bottom right: **console**: You can communicate (input and output) with the computer through the console.



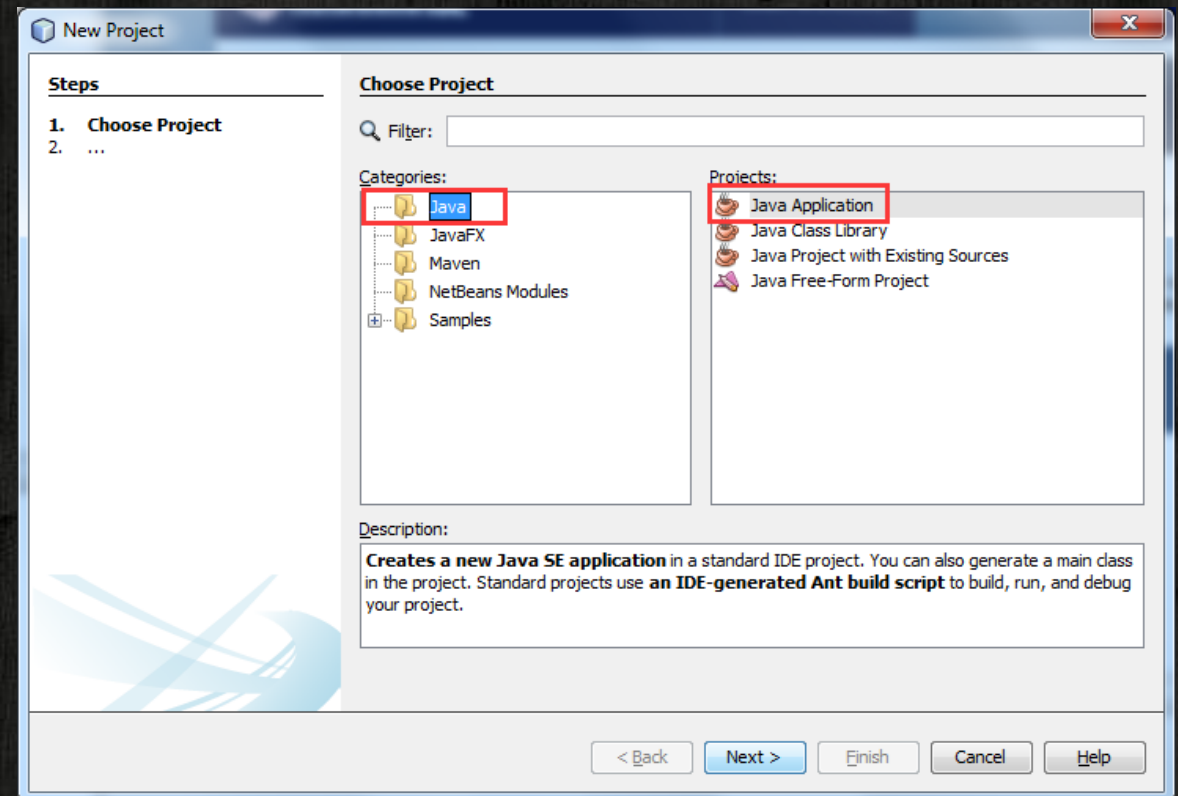
The First Java Project - I

- ◆ For many programming languages, the first task is to create a “**HelloWorld Project**”, which is to ask the computer to print “Hello World” for us.
- ◆ This project will make sure we can ask the computer to print something for us, so we can check the result when we have more complicated task in the future.
- ◆ In Java Programming, for each specific task, we will create a new **project**.
- ◆ **A project may contain more than one java files.**
- ◆ Each project will keep a specific file structure.



The First Java Project - 2

- ◆ Next step, we have to specify what kind of project do we want to create.
- ◆ Here we will create a pure **Java Application**, which belongs to the **Java** Category.



The First Java Project - 3

- ◆ After select the type of the project, we have to give an appropriate **Project name**.
- ◆ The first letter of the project name should be capitalized. The first letter of each word should also be capitalized. (“XxxxXxxx”)
- ◆ If you use your own computer, then store the project wherever you want.
- ◆ If you use the computer in the lab, strongly suggest you to **save the project on the Desktop**, it’s easier to find.
- ◆ Make sure the **last line (“Create Main Class”) is checked**. You don’t need to modify the name after. 8

Name and Location

Project Name: HelloWorld

Project Location: C:\Users\cstuser\Desktop

Project Folder: C:\Users\cstuser\Desktop\HelloWorld

☐ Use Dedicated Folder for Storing Libraries

Libraries Folder:

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

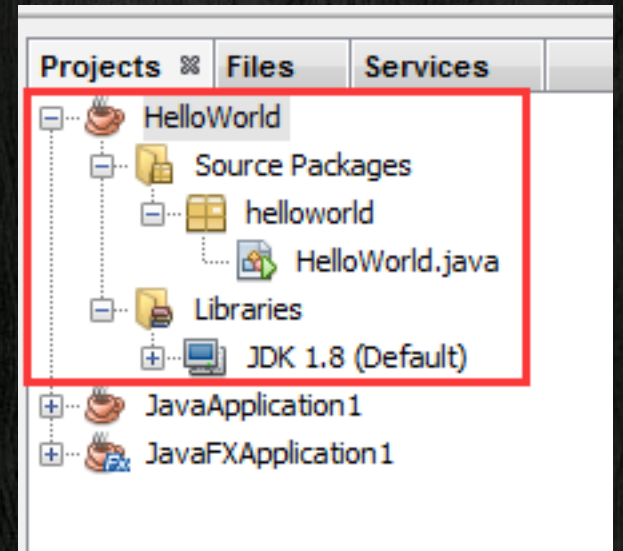
☒ Create Main Class helloworld.HelloWorld

The First Java Project

- ◆ Once you finish all these steps, a Java project is created.
- ◆ There will be a **new folder with the same name as the project** created on the desktop, that is the folder for the project.
- ◆ The folder follows a standard structure.
- ◆ The most important thing inside is the **“src” (stands for source) folder**, you can find all **source files** with extension **“.java”** inside. (These are the files you need to send to me for the assignments and exams.)
- ◆ While all the **compiled bytecode files with extension “.class”** files are located in the **“build” folder**.

The First Java Project – Project Navigator 1

- ◆ Now let's go back to the IDE.
- ◆ The Project navigator is updated.
- ◆ Each **Coffee cup** represents a Java Project.



The First Java Project – Project Navigator 2

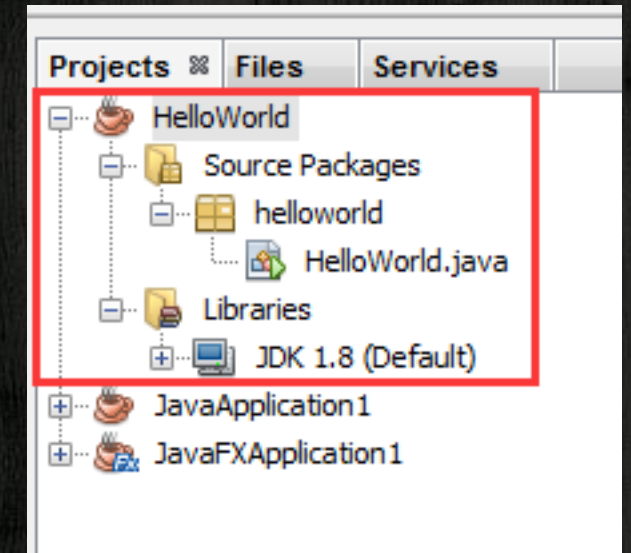
◆ Each project contains two parts:

○ Source Packages

- It contains the **source code**.
- The **box-like icon with name lowercase “helloworld”** is a package, it helps you to organize you code.
- If we have checked the “Create Main Class”, there is a “.java” file with the same name as the Project name here, If the icon of a “.java” file contains a green triangle, that means it contains a “main()” method.

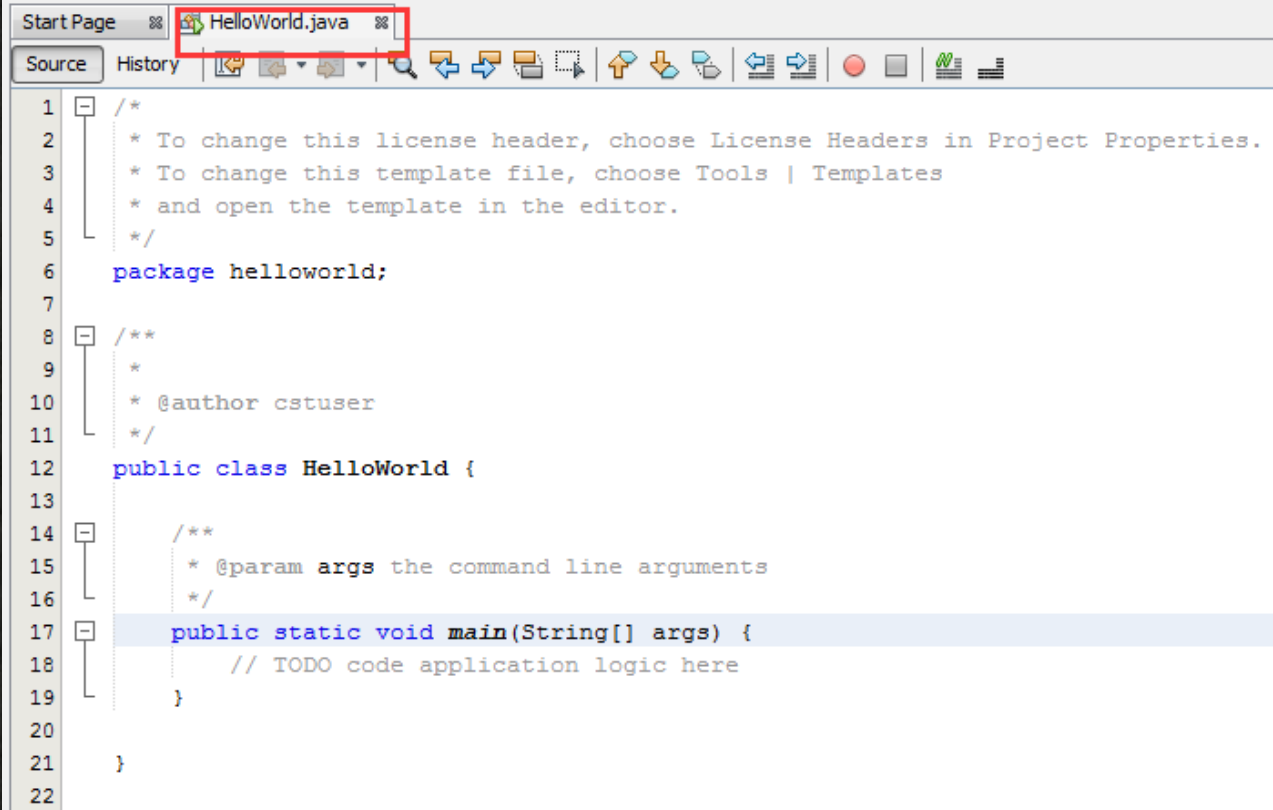
○ Libraries

- It contains a set of **Java build-in packages**.
- You can see in it we have **.jar files**. Each .jar file is a group of .class files. These are the bytecode files. (compiled) (package level)



The First Java Project – Text Editor Window 1

- ◆ Now let's take a look at the text editor part:
- ◆ The “HelloWorld.java” file with some code is automatically opened.




The screenshot shows an IDE window with a tab titled "HelloWorld.java" highlighted with a red rectangle. The editor displays the following Java code:

```
1  /*
2   * To change this license header, choose License Headers in Project Properties.
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 cstuser
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  }
22
```

The First Java Project – Auto-generated Code - Comment

- ◆ The first few lines in gray color are **comments**.
- ◆ Comments are not written for the computer. **It will be ignored when the code is compiled.**
- ◆ Comments are written for humans, to **explain the code, or debug the code.**
- ◆ In NetBeans default theme, comments are displayed in **gray color**.

```
1   /*  
2      * To change this license header, choose License Headers in Project Properties.  
3      * To change this template file, choose Tools | Templates  
4      * and open the template in the editor.  
5  */
```

The First Java Project – Auto-generated Code - Comment

◆ There are two kinds of comments in Java:

- Single line comment
- Multi lines comment

```
// A simple calculation
System.out.print(1 + 1);

int num1 = 5; // an int variable
int num2 = 6; // another int variable
```

◆ Single line comment

- Start from “//”, till the end of the line.
- Usually appear **one line above the code you want to explain**, or **on the right on a statement**.

◆ Multi lines comment

- Start from “/*”, finish at “*/”

```
/* You can put as many words as you want in between.
   Create two int variables and calculate their sum
*/

int num1 = 5;
int num2 = 6;
int result = num1 + num2;
```


The First Java Project – Auto-generated Code - Comment

- ◆ Personally I would suggest the students to use **single-line comment**.
- ◆ First, all multi-lines comment can be written as single-line comment.

```
// You can put as many words as you want in between.  
// Create two int variables and calculate their sum  
int num1 = 5;  
int num2 = 6;  
int result = num1 + num2;
```

- ◆ Most importantly, there is a **short-cut** (**ctrl + /**) to comment and uncomment your code.
- ◆ You can comment / uncomment **multiple lines** at the same time.

The First Java Project – Auto-generated Code - Comment

- ◆ Why use comment?
- ◆ Firstly, **do not trust your memory**. After a few days you will forget why you wrote a piece of code, no need to mention if you have to review a piece of code you wrote a year ago. Some comments will help you to catch your memory up.
- ◆ Secondly, **team-work is everywhere**. You will need to work with others, which means others need to read your code while you have to read theirs at one point. If you want others hate you less, put comments there in your code to help others to understand you well. **Two things programmers hate most: read other's unclean code, and read other's code with no comment.**

The First Java Project – Auto-generated Code - Comment

- ◆ Why use comment?
- ◆ Third, **let the computer temporally ignore our code**. You might need to try different solutions to solve a problem, and each try may lead you to a wrong result, so you have to do try-and-error to figure the right solution out. Each time when you try a thing that doesn't work, you don't want to delete it directly, cause maybe after a few tries, you will find the first version is OK with some modification. If you delete them at the first place, that means you have to re-write the code again. In this case, we would prefer to comment the code at first, so the computer will ignore it. If we want to have these code later, we can uncomment them back. So we do not need to worry if we loss the code at first.

The First Java Project – Auto-generated Code - Package

- ◆ Each Java package contains **one or more .java files**.
- ◆ **A package is basically a set of classes.**
- ◆ **Each package can be compiled to a .jar (bytecode) file.** (.jar is a group of .class files.)

```
package helloworld;
```

The First Java Project – Auto-generated Code - Documentation

- ◆ After the package there is a part displayed in gray color. It looks like a multi-lines comment, however, it is not.
- ◆ This part is called “**doc comment**”, which is used to comment on a **class level** or a **method level**. It is used to explain what does a whole class or a method do.
- ◆ The JDK javadoc tool uses doc comments when **preparing automatically generated documentation**.
- ◆ Doc comments **start with** “**/****” (two stars) instead of “**/***” (one star).

The First Java Project – Auto-generated Code - Documentation

◆ You can add **tags** in doc comments:

- **@author:** the author of this piece of code
- **@since:** when the code is written, could be a date, or could be a version number
- **@param:** parameters
- **@return:** what is returned from the method
- **@exception:** explain what kind of exception may appear

The First Java Project – Auto-generated Code - Class

```
public class HelloWorld {
```

- ◆ After the first doc comment, there is **a class with a name the same as the name of the .java file.**
- ◆ In Java, **everything is defined in class.** (Object-oriented Programming OOP)
- ◆ **A class name should always start with a capital letter.** (only class, not variable, not method.)
- ◆ **public** (access modifier) means everyone can visit it. (we will see more about it in the future)
- ◆ **A pair of {}** after the class name indicates the range of the body of that class.

The First Java Project – Auto-generated Code – main()

```
public class HelloWorld {
```

- ◆ Inside of the class, we can find a **method** “**main()**”.
- ◆ A **method** can be understood as a **function** in the other high-level languages. Which should do a specific job. But in OOP, people prefer to call it method instead function.
- ◆ **A method is defined to do a specific task.** For example: `max(a, b)` is used to compare two numbers, `a` and `b` and find the max between them. `round(a)` is used to round a number to the closest integer number. `countVowel(a)` is used to count how many vowels are there in a string, etc.

The First Java Project – Auto-generated Code – main()

```
public class HelloWorld {
```

- ◆ A project may contains many classes, while a class may contain many methods.
- ◆ Then here comes a problem: if we have many methods inside of a project, then which one should Java use to start the program?
- ◆ The answer is main(). **Java will always start a project from a main() method.** main() can be considered as the entrance of a project.
- ◆ Once a “.java” file contains a “main()” method, then you can see the small green triangle in its icon.
- ◆ **main() should be the place for you to “test” your code once you have finished designing your code.**

The First Java Project – Auto-generated Code – Keywords

- ◆ In NetBeans default theme, **all Java keywords are displayed in blue color.**
- ◆ **A key word is something pre-reserved in Java**, and you can only use it in the way Java defined. You have no right to modify its usage

- ◆ So far we have seen 6 keywords:

- **package**: indicates which package this file belongs to.
- **public**: (one kind of access modifier) means everyone can visit
- **class**: define a class
- **static**: You can use it without declaring an object. (We will see it in the future)
- **void**: no return value.

```
1  /*
2  * To change this license header, choose License Headers in Project Properties.
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 cstuser
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  }
22
```

The First Java Project – Auto-generated Code – Keywords

- ◆ It is OK if you don't understand all these keywords. Anyway this is the first day of learning Java.
- ◆ But you should remember **main() method always have “public static void” in front.**
- ◆ If you want to create a main() method by yourselves. You can type “p” and wait for 0.1 second, then you will see a **auto-completion** menu will pop-up, the first one is **main()**. You can press “Enter” or “Tab” to select it. So basically press two keys: **p, and enter or tab.** (You have to first check the **auto-completion** first).
- ◆ To turn auto-completion on:
 - **Tools -> Options -> Editors -> Code Completion -> Language: Java -> check “Auto Popup on Typing Any**

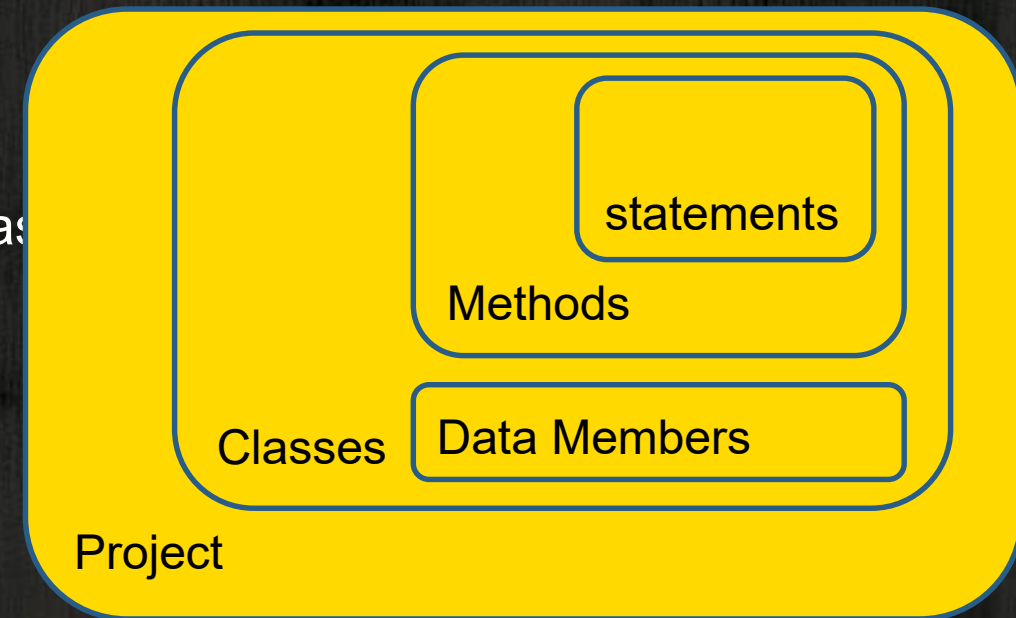
The First Java Project – Auto-generated Code – Method

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

- ◆ If we zoom-in to look at a method (e.g.: main()), we will see a method has
 - **a name**: to identify a method. the first letter of a method name is lower-case (e.g.: main, max, round, countVowel).
 - **a pair of ()**: parameters of a method, you can understand it as the input of a method.
 - **a pair of {}**: indicate the range of the body of the method.
- ◆ In main(), there is only one one-line comment, which tells us to write the code inside. Since we know we need to write our code there, **we can directly delete it.**

The First Java Project – Statement

- ◆ In Java, the smallest unit is a **statement**.
- ◆ A statement can only do one tiny thing. For example, declare a variable, or calculate one math operation.
- ◆ A statement end with “;”.
- ◆ In general you can understand Java structure as



The First Java Project – println

- ◆ We will learn our first Java statement today:

```
System.out.println([parameter]);
```

- ◆ We can simply call it as “print L N” or “print line”, which is used to print something in the console.
- ◆ Since there is a pair of (), so this statement is also a method, which is pre-defined by JDK. Here we are not defining it, but only use it.
- ◆ If we want to print something, we have to tell Java what do we want to print, which is the parameter (input) of a method, which is given between ().

The First Java Project – println

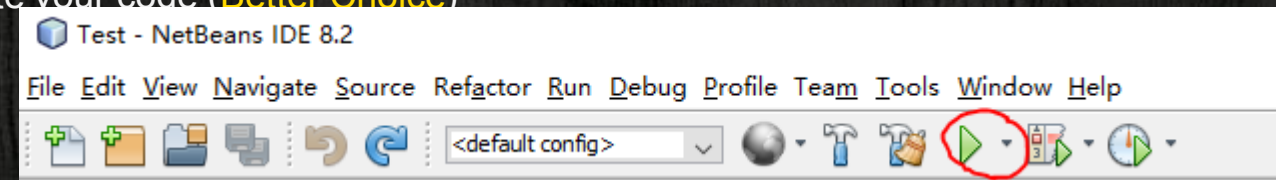
- ◆ We want to print a string “Hello World”.
- ◆ In Java, a string is defined between a pair of “”.

```
System.out.println("Hello World");
```

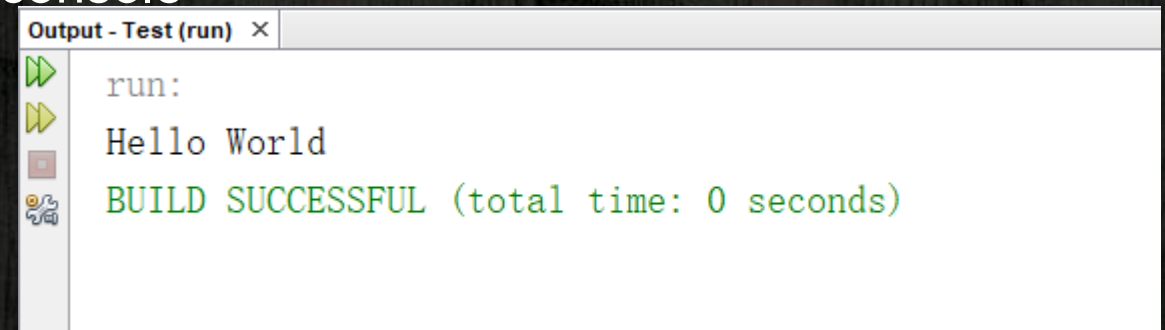
- ◆ This is method with a long name, fortunately there is no need for us to type the full name of it. There is a **short cut** for it. Type “**sout**” and then type “**tab**” or “**enter**”. NetBeans will generate the method for you.

The First Java Project – Execute your Code

- ◆ Once you add this statement in the main(), we can execute our code.
 - You can click the green triangle in the menu bar to execute your code, this will automatically (1) **save your code**, (2) **compile your code**, (3) **link your code**, (4) **execute your code**, and (5) **display the result in the console**.
 - You can also use the short cut (**F6**) to execute your code (**Better Choice**)



- ◆ And the result will be displayed in the console



The First Java Project – print, println, printf

- ◆ We can modify this line of code a little bit:
- ◆ After the second “.”, you can use `print()`, `println()`, and `printf()` (`printf()` is more complicated but much more powerful than the other two. We will learn it very soon).
- ◆ `print`: print the sentence **without a line breaker in the end**.
- ◆ `println`: print the sentence **with a line breaker in the end**.
- ◆ `printf`: more freedom to **format the output** (e.g.: alignment, decimal accuracy, zero-padding, etc.).

The First Java Project – Indent

- ◆ It is obligated to keep your code clean.
- ◆ Always indent your code if it belongs to another bigger container.
- ◆ For example:
 - the main() is part of the HelloWorld class, so the entire main() method should be indented by one level (one tab) in the class.
 - the println() statement is part of the main() method, so this statement should be indented by one level in the main() method.

The First Java Project – Other Notices

- ◆ In Java, **EVERYTHING** is inside of a class.
- ◆ Make sure do not write anything after the closing “}” of a class.
- ◆ Use auto-completion **AS MUCH AS POSSIBLE**.



Questions?

Hands on

Task 1

Print the information as below:

```
run:
Attribute      Value

Name:         Yi Wang
Age:          33
Nationality:   China
Department:    Computer Science
Position:      Lecturer
BUILD SUCCESSFUL (total time: 1 second)
```


Hands on

Task 2

Print the question and fill the answer of the equation:

```
run:
1234 (D) =      (H)
1011 0101 1100 (B) =      (H)
AB (H) =      (D)
795 (D) =      (B)
1101 0110 1010 (B) =      (D)
C7 (H) =      (B)
BUILD SUCCESSFUL (total time: 2 seconds)
```

Hands on

Task 3

Print the shapes displayed as below:

