Introduction to Java for C++ Programmers
JAC444

Week 01
Introduction to Java Programming and IntelliJ

Instructor

Hossein Pourmodheji

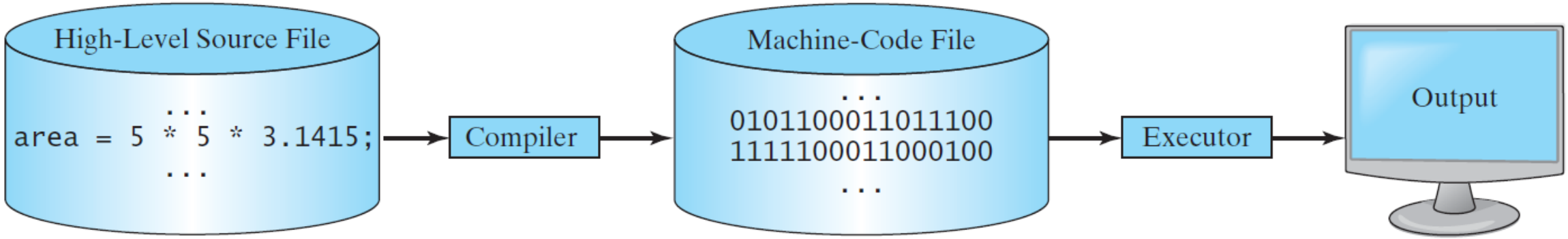
hossein.pourmodheji@senecacollege.ca

Programming Languages

- Computers do not understand human languages, so programs must be written in a language a computer can use.
- Programmers write instructions in various programming languages, some directly understandable by computers and others requiring intermediate translation steps.
- Three general language types:
  - Machine Language
  - Assembly Language
  - High-Level Language

Compiler

- A compiler translates the entire source code into a machine-code **file**, and the machine-code file is then executed.



History of Java

- Sun Microsystems funded an internal corporate research project, which resulted in a C++-based language named Java.
- Originally called *Oak*, Java was designed for use in embedded chips in consumer electronic appliances.
- The web exploded in popularity, Sun saw the potential of using Java to add dynamic content to web pages.
- In 1995, Java was redesigned for developing Web applications.

Java Language Specification and API

- The **Java language specification** is a technical definition of the Java programming language's syntax and semantics.
  - https://docs.oracle.com/javase/specs/

- Java programs consist of pieces called classes.

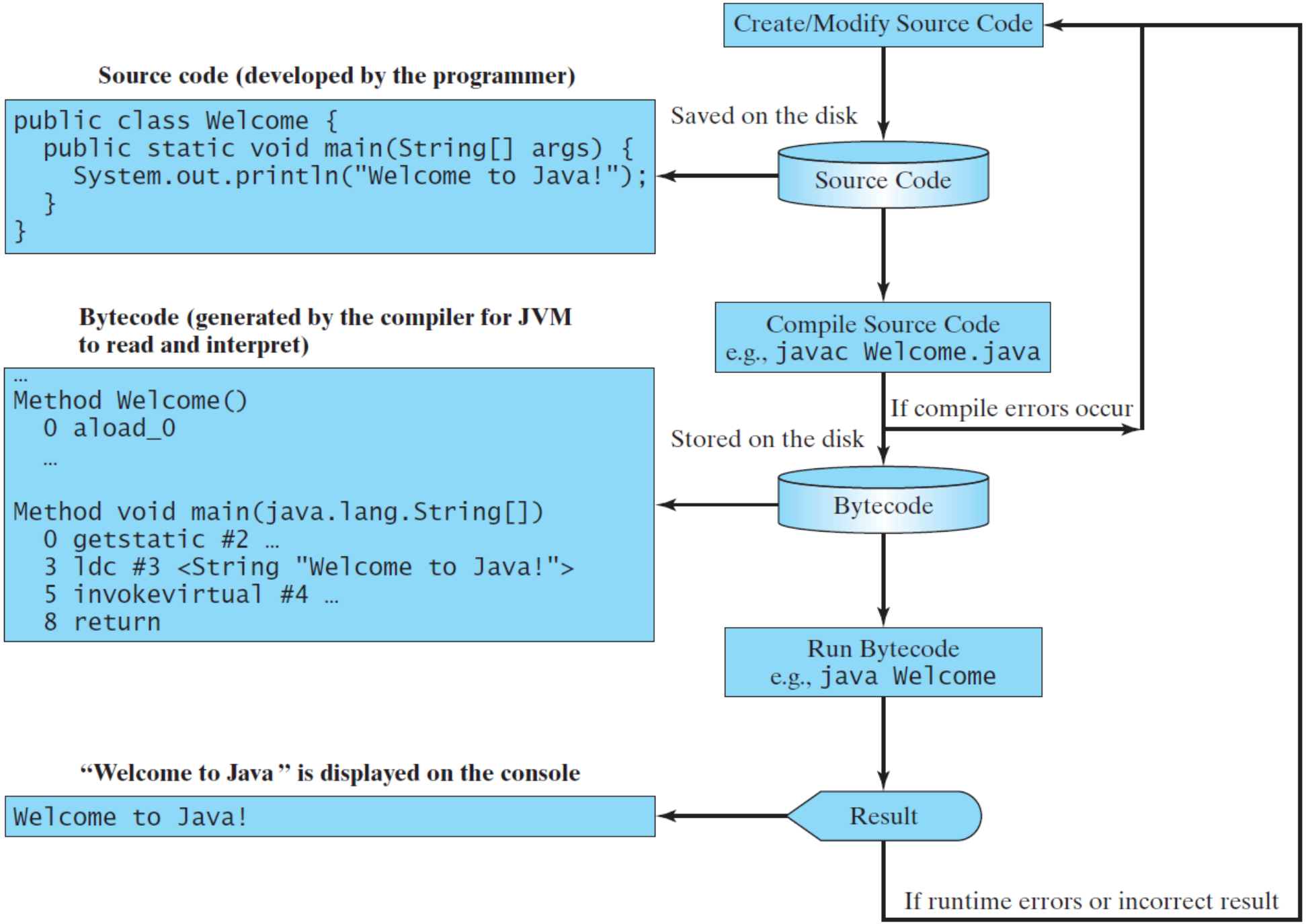
- **Application Program Interface (API)**, also known as **library**, contains predefined classes and interfaces for developing Java programs.
  - https://docs.oracle.com/javase/8/docs/api/

Java **JDK**

- **Java Development Toolkit (JDK)** consists of a set of separate programs, each invoked from a command line, for developing and testing Java programs.

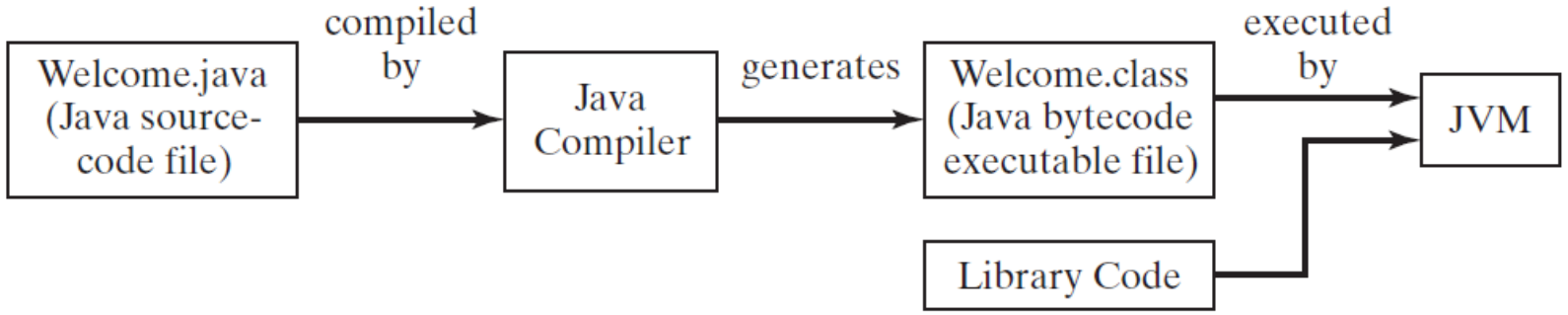
- Oracle releases each version of Java SE with a JDK.

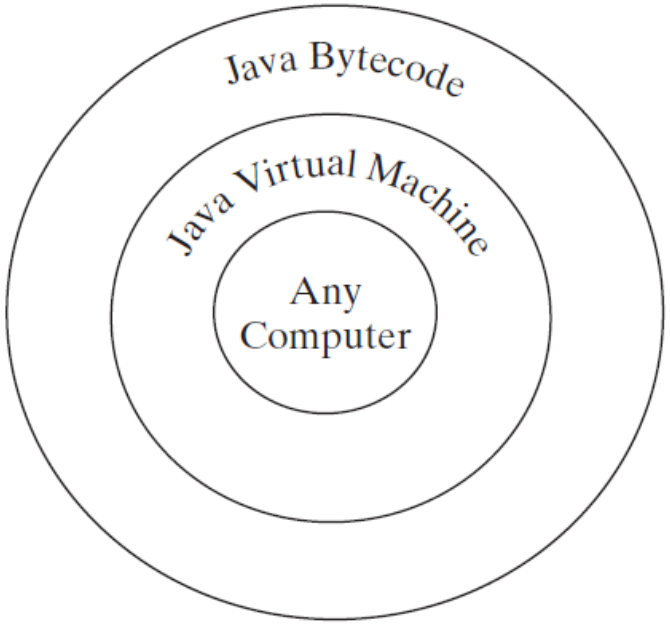
- The Java SE **JDK 8**
  - https://www.oracle.com/java/technologies/downloads/


Create/Modify Source Code
Source code (developed by the programmer)
public class Welcome {
public static void main(String[] args) {
System.out.println("Welcome to Java!");
}
}
Saved on the disk
Source Code
Compile Source Code
e.g., javac Welcome.java
If compile errors occur
Bytecode (generated by the compiler for JVM to read and interpret)
...
Method Welcome()
0 aload_0
...
Method void main(java.lang.String[])
0 getstatic #2 ...
3 ldc #3 <String "Welcome to Java!">
5 invokevirtual #4 ...
8 return
Stored on the disk
Bytecode
Run Bytecode
e.g., java Welcome
"Welcome to Java" is displayed on the console
Welcome to Java!
Result
If runtime errors or incorrect result

Java Virtual Machine (JVM)

- The **bytecode** is similar to machine instructions but is architecture neutral and can run on any platform that has a **Java Virtual Machine (JVM)**.





Editor

- To create or edit a Java file, you can use editor program (normally known simply as an editor).
- The source file must end with the **.java** extension.
- It must have the same exact name as the class name.
- Linux editors: vi, emacs, nano, pico and jEdit.
- Windows editors: Notepad++, EditPlus (www.editplus.com), TextPad (www.textpad.com) and jEdit (www.jedit.org).

Compiling

- Use the command javac (the Java compiler) to compile a program. For example, to compile a program called `HelloWorld.java`, you'd type

```
javac HelloWorld.java
```

- If the program compiles, the compiler produces a .class file called `HelloWorld.class` that contains the compiled version of the program.

Executing

- Bytecodes are platform independent
  - They do not depend on a particular hardware platform.
- Bytecodes are portable
  - The same bytecodes can execute on any platform containing a JVM that understands the version of Java in which the bytecodes were compiled.
- The JVM is invoked by the java command. For example, to execute a Java application called `HelloWorld`, you'd type the command

```
java HelloWorld
```

Java **IDE**

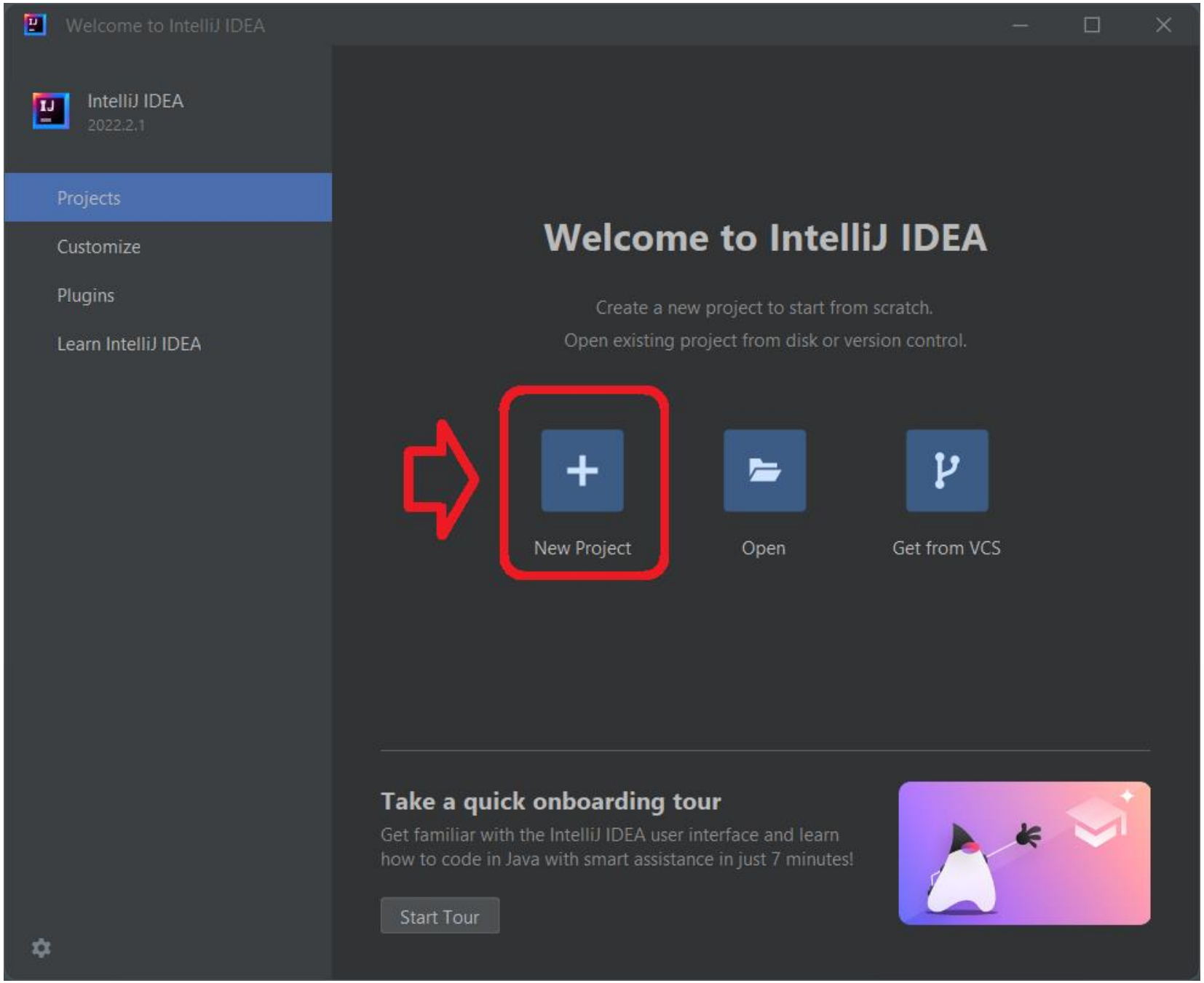
- Java development tool—software that provides an integrated development environment (IDE) for developing Java programs quickly.
  - **IntelliJ IDEA**
  - Eclipse
  - NetBeans

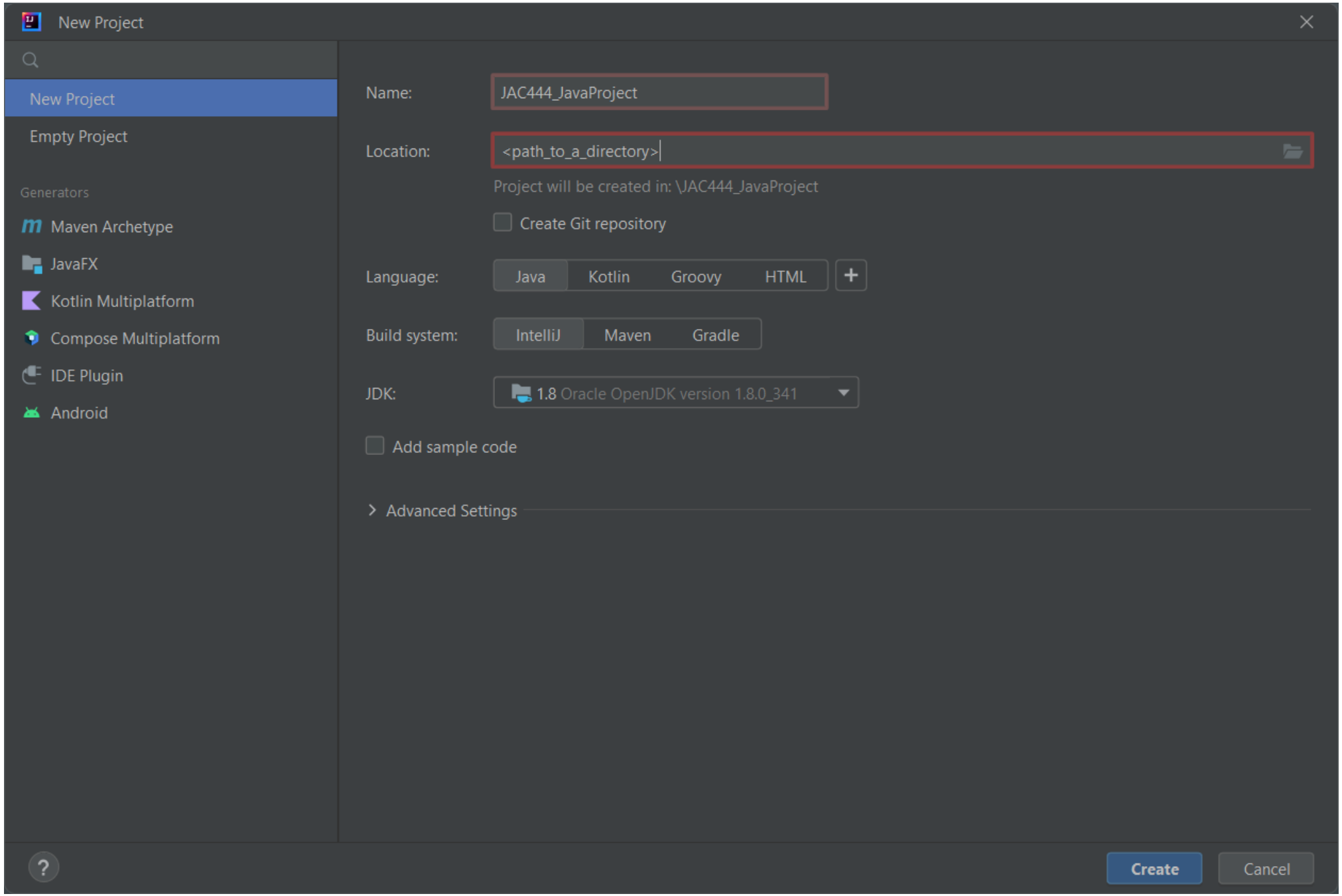
- Editing, compiling, building, debugging, and online help are integrated in one graphical user interface.

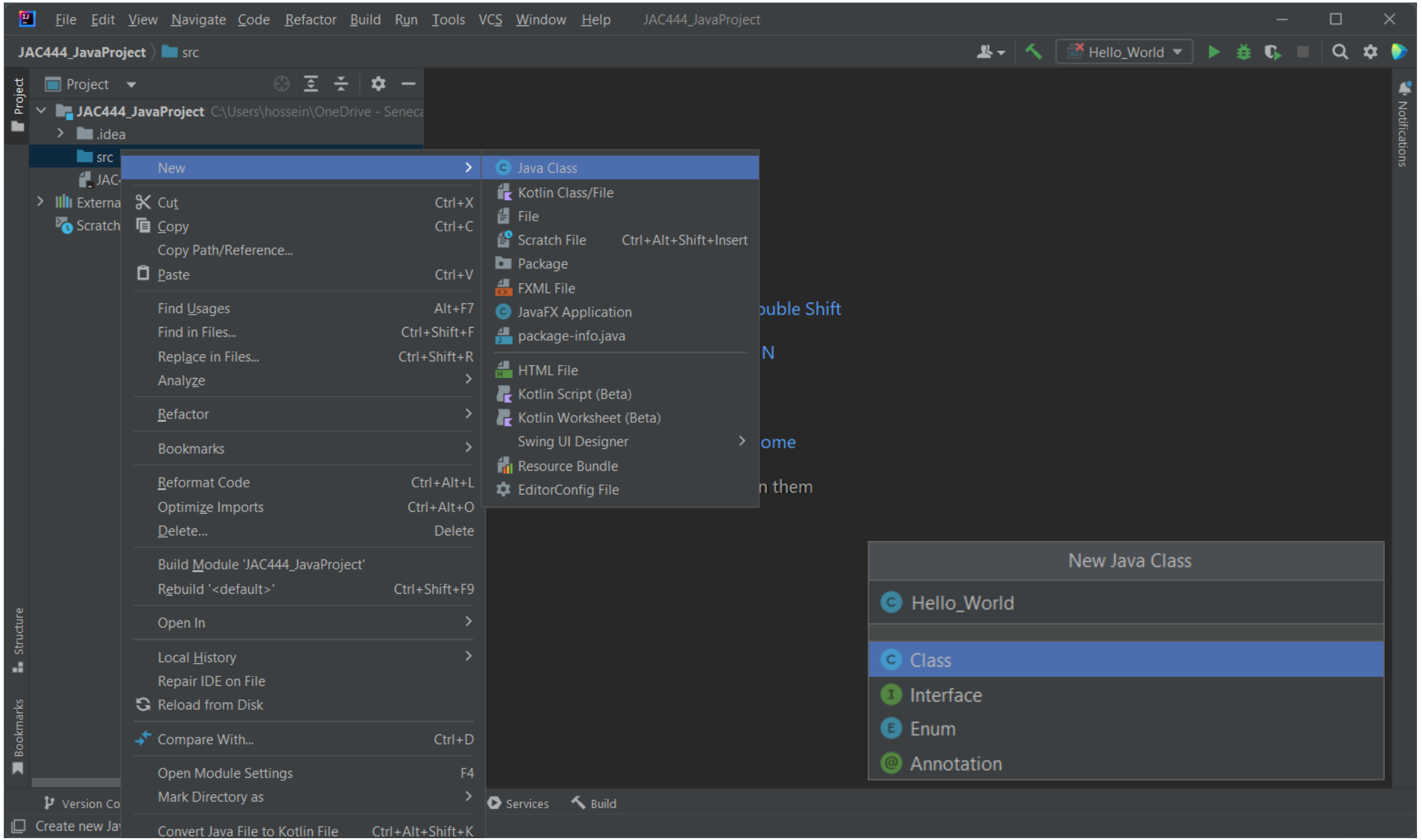
IntelliJ Installation and Configuration

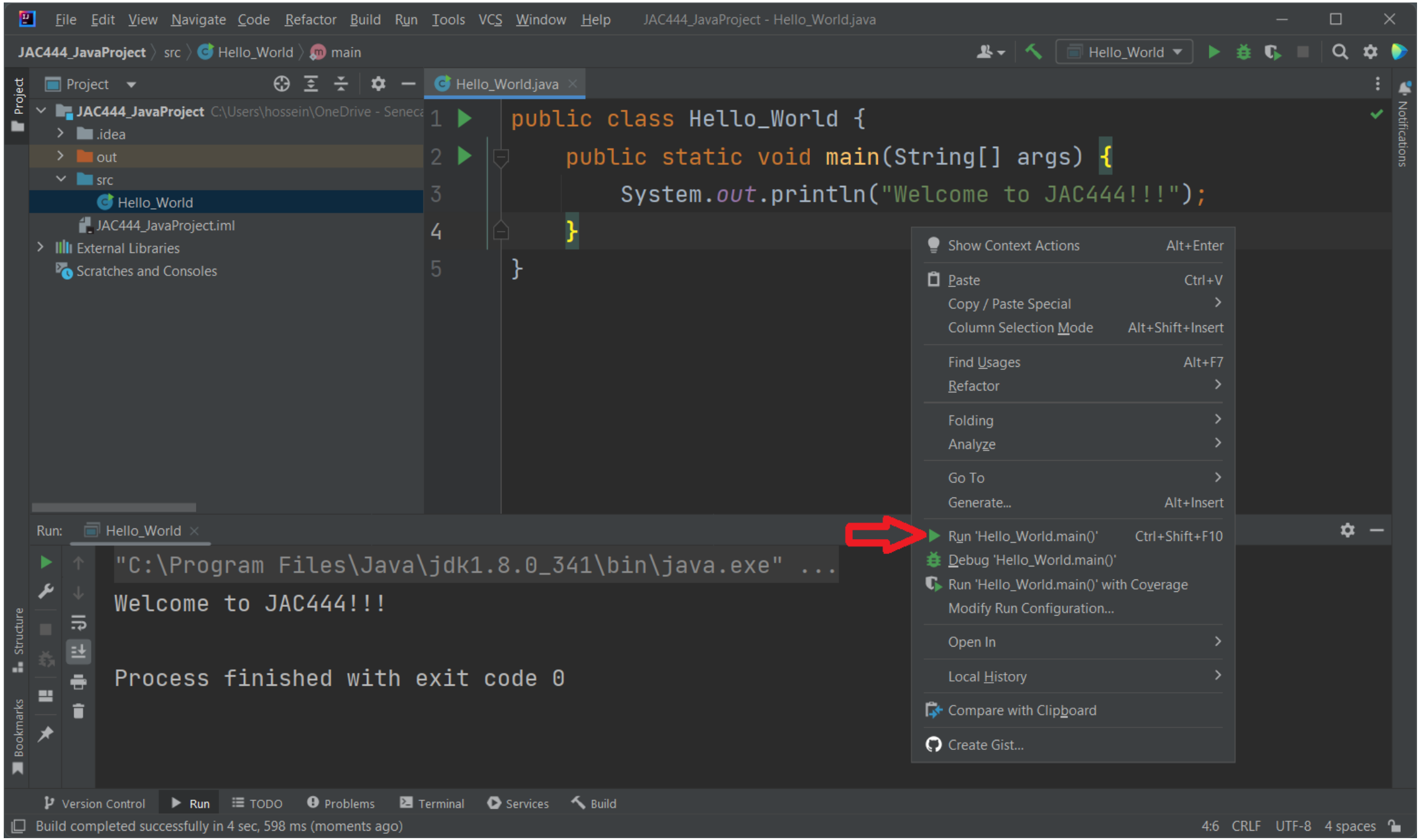
- Select, download and install the "Community" edition in the following page:

  https://www.jetbrains.com/idea/download/


Welcome to IntelliJ IDEA
IntelliJ IDEA
2022.2.1
Projects
Customize
Plugins
Learn IntelliJ IDEA
Welcome to IntelliJ IDEA
Create a new project to start from scratch.
Open existing project from disk or version control.
New Project
Open
Get from VCS
Take a quick onboarding tour
Get familiar with the IntelliJ IDEA user interface and learn how to code in Java with smart assistance in just 7 minutes!
Start Tour


New Project
New Project
Empty Project
Generators
Maven Archetype
JavaFX
Kotlin Multiplatform
Compose Multiplatform
IDE Plugin
Android
Name:
JAC444_JavaProject
Location:
<path_to_a_directory>
Project will be created in: \JAC444_JavaProject
Create Git repository
Language:
Java
Kotlin
Groovy
HTML
Build system:
IntelliJ
Maven
Gradle
JDK:
1.8 Oracle OpenJDK version 1.8.0_341
Add sample code
Advanced Settings
Create
Cancel


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
JAC444_JavaProject
JAC444_JavaProject src
Hello_World
Project
JAC444_JavaProject C:\Users\hossein\OneDrive - Seneca
.idea
src
Externa
Scratch
New
Cut Ctrl+X
Copy Ctrl+C
Copy Path/Reference...
Paste Ctrl+V
Find Usages Alt+F7
Find in Files... Ctrl+Shift+F
Replace in Files... Ctrl+Shift+R
Analyze
Refactor
Bookmarks
Reformat Code Ctrl+Alt+L
Optimize Imports Ctrl+Alt+O
Delete... Delete
Build Module 'JAC444_JavaProject'
Rebuild '<default>' Ctrl+Shift+F9
Open In
Local History
Repair IDE on File
Reload from Disk
Compare With... Ctrl+D
Open Module Settings F4
Mark Directory as
Convert Java File to Kotlin File Ctrl+Alt+Shift+K
Java Class
Kotlin Class/File
File
Scratch File Ctrl+Alt+Shift+Insert
Package
FXML File
JavaFX Application
package-info.java
HTML File
Kotlin Script (Beta)
Kotlin Worksheet (Beta)
Swing UI Designer
Resource Bundle
EditorConfig File
New Java Class
Hello_World
Class
Interface
Enum
Annotation
Services
Build
Notifications
Structure
Bookmarks
Version Co
Create new Ja


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
JAC444_JavaProject - Hello_World.java
JAC444_JavaProject src Hello_World main
Hello_World
Project
Hello_World.java
JAC444_JavaProject C:\Users\hossein\OneDrive - Seneca
.idea
out
src
Hello_World
JAC444_JavaProject.iml
External Libraries
Scratches and Consoles
1 public class Hello_World {
2 public static void main(String[] args) {
3 System.out.println("Welcome to JAC444!!!");
4 }
5 }
Show Context Actions Alt+Enter
Paste Ctrl+V
Copy / Paste Special
Column Selection Mode Alt+Shift+Insert
Find Usages Alt+F7
Refactor
Folding
Analyze
Go To
Generate... Alt+Insert
Run 'Hello_World.main()' Ctrl+Shift+F10
Debug 'Hello_World.main()'
Run 'Hello_World.main()' with Coverage
Modify Run Configuration...
Open In
Local History
Compare with Clipboard
Create Gist...
Run: Hello_World
"C:\Program Files\Java\jdk1.8.0_341\bin\java.exe" ...
Welcome to JAC444!!!
Process finished with exit code 0
Version Control Run TODO Problems Terminal Services Build
Build completed successfully in 4 sec, 598 ms (moments ago)
4:6 CRLF UTF-8 4 spaces

First Java Program!!!

- Let’s create our first Java program called HelloWorld.java that just prints “Hello World!!!” into the output screen.

Comment Style

- Line comments
  - Beginning with //
- Block Comments
  - Begin with /* and end with */

Programming Errors

- Programming errors can be categorized into three types:
  - Syntax errors
  - Runtime errors
  - Logic errors

Syntax Errors

- Errors that are detected by the compiler are called **syntax errors** or compile errors.
- Syntax errors result from errors in code construction:
  - mistyping a keyword
  - omitting some necessary punctuation
  - using an opening brace without a corresponding closing brace

Example of Syntax Errors

```
1  public class ShowSyntaxErrors {
2    public static main(String[] args) {
3      System.out.println("Welcome to Java);
4    }
5  }
```

Runtime Errors

- **Runtime errors** are errors that cause a program to terminate abnormally.
- They occur while a program is running if the environment detects an operation that is impossible to carry out.
- Input mistakes typically cause runtime errors that are called input errors.
- Examples:
  - data-type error: if the program expects to read in a number, but instead the user enters a string.
  - division by zero: this happens when the divisor is zero for integer divisions.

Example of Runtime Error

```
1  public class ShowRuntimeErrors {
2    public static void main(String[] args) {
3      System.out.println(1 / 0);
4    }
5  }
```

Logic Errors

- **Logic errors** occur when a program does not perform the way it was intended to.

```
1  public class ShowLogicErrors {
2    public static void main(String[] args) {
3      System.out.println("Celsius 35 is Fahrenheit degree ");
4      System.out.println((9 / 5) * 35 + 32);
5    }
6  }
```

```
Celsius 35 is Fahrenheit degree
67
```