
# Introduction to Programming in Java

# Worksheet 1

# Worksheet 1

Introduction, motivation, setting up, and IDEs

## Introduction to the course

Welcome to the course **Introduction to Programming in Java**. In this section, we will discuss how this course will be managed, the opportunities that it provides for you, and the requirements and assessment to pass it.

### Course management

This is a fully online, do-it-yourself course that is divided into 10 sessions. You may progress through these sessions at your own pace – but the maximum recommended time for completion is 10 weeks (1 session per week). Each session will contain a worksheet that will first explain the theory behind the programming concepts being taught, followed by examples and then practical exercises. To pass this course, it is necessary to complete all sessions. No prior experience with programming is required, but you should be confident in using a computer to perform basic operations such as exploring files and folders, managing windows, changing settings, and so on. Prior knowledge of computer fundamentals will be beneficial.

### What is the course about?

In the course, you will learn about the basic concepts of programming by writing simple programs in a programming language called Java. With the rapid advances in technology in recent years, learning how to program will be beneficial for everyone, not just people who seek to become computer scientists. In the future, it is a very likely possibility that computer programming will be considered a core skill, on par with language and mathematics. The skills you learn in this course will be developed based on a single programming language (Java), but the fundamental concepts are transferrable to all other major programming languages used in the industry. Upon completion of this course, you should:

- Be familiar with the fundamental concepts of computer programming.
- Be able to write, read, and analyze basic programs.
- Feel confident in devising appropriate solutions for a given problem.
- Have the foundations to progress to more advanced programming courses.

### Who is this course for?

This course is meant for anyone and there is no need to have any prior experience with computer programming. However, it will be very helpful for those attending this course to have existing knowledge of computer fundamentals and intermediate mathematics. The course will teach programming skills that anyone with knowledge of the above can master, from aspiring computer scientists to those seeking to learn programming as a hobby or an extracurricular activity. For this course will you need a decent laptop or desktop computer, which is preferably running on Windows 7 or a later version. You will also need an internet connection for downloading the course's worksheets and the related programming tools we will use. However, after downloading these, an internet connection is no longer required.

## Assessment

The assessment for this course will involve a ***multiple-choice quiz***, taken at the end of the course, whenever you feel ready. The quiz will contain 25 questions worth 4 points each, with a total of 100 points. Each question will have four possible answers and only one correct answer and you will have 25 minutes to answer all questions. The questions in this quiz will assess your knowledge of the material taught in all the worksheets. To pass this assessment, you need to score at least 50 out of 100 points.

## Opportunities

By attending this course, you will learn things that may eventually turn you into a:

- **Software engineer** – someone who designs, develops, tests, and maintains computer software.
- **Web developer** – someone who designs and develops web pages/sites.
- **Games developer** – someone who designs and develops computer games.
- **Mobile app developer** – someone who designs and develops mobile applications.
- **Systems developer** – someone who creates computer systems for devices used in the industry.

And much more…

# Introduction to programming and motivation

In this section, we begin our journey into computer programming. To understand what computer programming is, we must first understand what computers are.

A **computer** *is an electronic device that stores, retrieves, and processes data according to instructions given to it*.

Some examples of computers are:

- A smartphone or tablet.
- A smartwatch or music player.
- A game console (Xbox, PS4 or PS5, X-Box, etc.)
- A car, smart fridge, smart washer, etc.
- Your laptop, desktop computer, etc.

All of these devices store, retrieve, and process data and are programmed in specific ways that make them useful to humans by executing a specific set of tasks. You may be familiar with some of the hardware that comes with/in a computer, such as hard disks, monitors, keyboards, and so on, but all a device needs to be a computer is:

1) A Central Processing Unit (CPU).
2) Memory.
3) Storage.

## The Central Processing Unit

The **Central Processing Unit** (CPU), also called a "processor", is a piece of electronic circuitry that executes very basic instructions. These instructions can be either arithmetic (e.g., finding the result of 1 + 2) or logical (e.g., evaluating if something is true or not). Because of that, a CPU is often considered the "brain" of the computer.

## Memory

Computer memory comes in two flavors: **ROM** (Read-Only Memory) and **RAM** (**Random Access Memory**). The differences between these two are not within the scope of this course. If you are not aware of what ROM and RAM are, you may want to do a bit of research at this point to learn more.

In general, you should be aware that ROM memory cannot be easily modified and is used for standard operating system tasks. We will not deal with ROM in this course. On the other hand, **RAM** *is the main memory of a computer – it allows a computer to remember data which can be later used to achieve a particular task*. We will discuss how data is stored in RAM in the next session.

## Are computers smart?

The typical instructions a computer's processor can execute are *very simple and mechanical*. The following are some examples of instructions that a CPU can execute:

- Copy the contents of one memory location to another.
- Retrieve the contents of a memory location and add it with those of another memory location.
- Store the result of an operation in a memory location.

And so on…

One may say that these are operations that an attentive first grader could do. Therefore, we can deduce that computers are not particularly smart and definitely not smarter than an average adult human. However, despite their inferior intelligence compared to humans, they have somehow managed to beat some of our best chess players. The reason is that *computers can execute these trivial operations extremely fast, allowing them to compute a large number of potential moves and outcomes within milliseconds*.

A CPU's speed is measured in terms of *Hertz*, which is a unit used to measure *how many times something can change in one second*. A CPU clocked at 1 Hertz is therefore able to execute 1 instruction per second. To demonstrate how fast today's computers are, a Core i9-7900X, which is a high-end commercial CPU can execute instructions at 4.3 billion Hertz – or 4.3 Gigahertz. That is 4.3 billion instructions *per second*. This is augmented by the fact that today's CPUs have multiple cores working in parallel at this frequency. The mentioned CPU contains 10 cores. At maximum efficiency, this means that the CPU can execute 10 x 4.3 = 43 billion instructions per second.

Similarly, computer memory (RAM) capacity has grown significantly in the last two decades. Computer memory works by assigning data to specific "locations" called *memory cells*. Each of these memory cells can contain a binary value – either 1 or 0 and can be used by programs that are executed by the CPU. You may have heard or read the term "4GB of RAM" during a commercial or in a computer sales ad. What this symbolizes is the total available amount of RAM in a computer. Information is stored in terms of **bytes** and the more bytes that are available in a computer's memory, the more information it can store and therefore the more programs it can run simultaneously. Computers today have at least 4 GigaBytes of RAM, which is about 4 billion memory cells.

To conclude, a computer is only able to execute basic instructions but can do so at extreme speeds which are unimaginable for humans. Similarly, a computer can only remember trivial bits of information but can remember much more than a human can remember at once. This combination makes computers powerful enough to beat people at chess.

## Programming a computer

Computer programming, known as simply **programming** in the context of computer science, *is the process of creating programs*. A **program** *is a set of instructions that tells a computer how to perform a task*. The act of programming is carried out by a **programmer,** a person who writes computer code that tells a computer what to do. Professional programmers are often called *software developers* or *software engineers*. The responsibilities of a software engineer are not just to write code from scratch, but to collaborate with other programmers, test code, evaluate its efficiency, find errors (a process known as *debugging*), and maintain code written by others.

## Computer languages

Telling a computer what to do is not as straightforward as instructing a human. We can communicate with humans using one of our ~7,000 languages. However, human language is not ideal for communicating with a computer for various reasons. First and foremost, computers can only understand a particular set of instructions – such as those mentioned in the previous section. Programmers need to communicate with a computer by speaking its processor's language, also known as a *machine language* or *machine code*. The following figure shows a fragment of machine language instructing a computer to add two numbers:

```
main:
   mov         eax, 1
   mov         ebx, 2
   add         eax, ebx
   push        eax
```

To a human, computer language is *very hard to read and tedious to write*.

## Human languages

The case against using human languages is their ambiguity. Sentences written in human languages may be hard to analyze by a computer and there may be scenarios where sentences can have multiple meanings. For example, consider the two following sentences:

"Kids make nutritious snacks"

"Red tape holds up new bridge"

In the first sentence, ambiguity arises from the fact that kids could be making nutritious snacks. Another interpretation of the same sentence, however, could be that kids *are* nutritious snacks. Similarly, in the second sentence, the phrases "red tape" and "holds up" could be easily misinterpreted to mean that a bridge is being held up by an actual tape with red color. Parsing human language requires context – which computers are not aware of. Therefore, it would be very inefficient for a computer to analyze all the possibilities of what the programmer is instructing it to do when using a human language.

## Programming languages

A programming language is a midpoint between a computer language and a human language. **Programming languages** *are languages that can be easily understood by humans who have had the appropriate training*. *They also can be easily translated into computer language* by a special type of program known as a *compiler*.

A **compiler** is a program that takes in code written in a programming language and translates it into machine code.

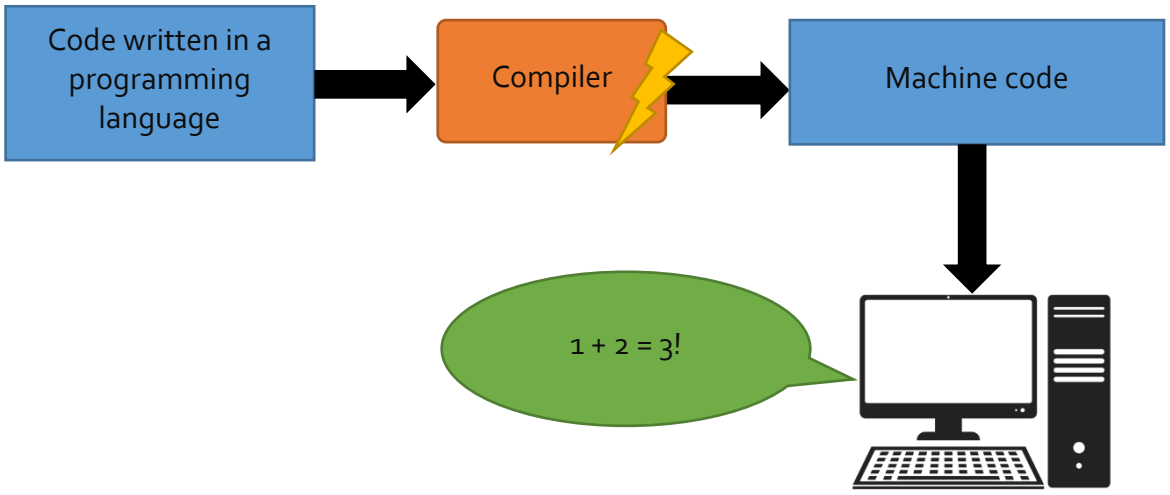



**Figure 1: How a program written in a programming language is compiled to machine code and executed by a computer.**

The process described above is known as **compilation**.

Just like with human languages, there is a huge variety of programming languages to choose from. In this course, we will use the **Java** programming language. Java is a commercial language that is widely used and widely available. Learning Java is relatively easy and will provide you with the experience to learn other programming languages later. The concepts taught in this course are not specific to Java and are thus transferrable to all other programming languages. Java embraces a set of abstractions that make it easier for the programmer to manage computer memory and has automated checks for catching mistakes in programs – which is also ideal for beginners. For this course, we will only use a minimal subset of Java, instead focusing on important transferrable programming principles.

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

**Figure 2: A simple program that prints the words "Hello World" written in Java - a modern programming language.**

# Setting up

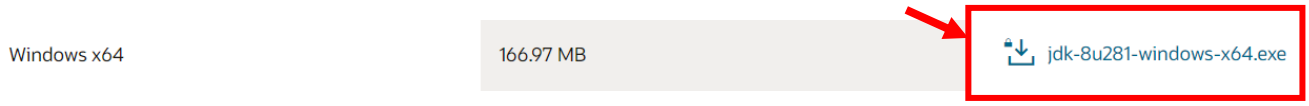
## Download and install Java

Before you can begin programming with Java, you must install it on our computer. For this course, we will use Java 8, a relatively advanced version of Java. Your first task is to navigate to the page below and download Java.

**Download Java:**

https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html

The page presents you with a variety of options. If you are on a computer running Windows, scroll to the bottom of the list and find "Windows x64". Click on the link to download the corresponding Java Development Kit (JDK).




Alternatively, if you are running Linux or MacOS, find the corresponding links and download the JDK from there. When the download completes, find the file and run it. An installation window will appear, allowing you to start the installation.

Press next to continue.

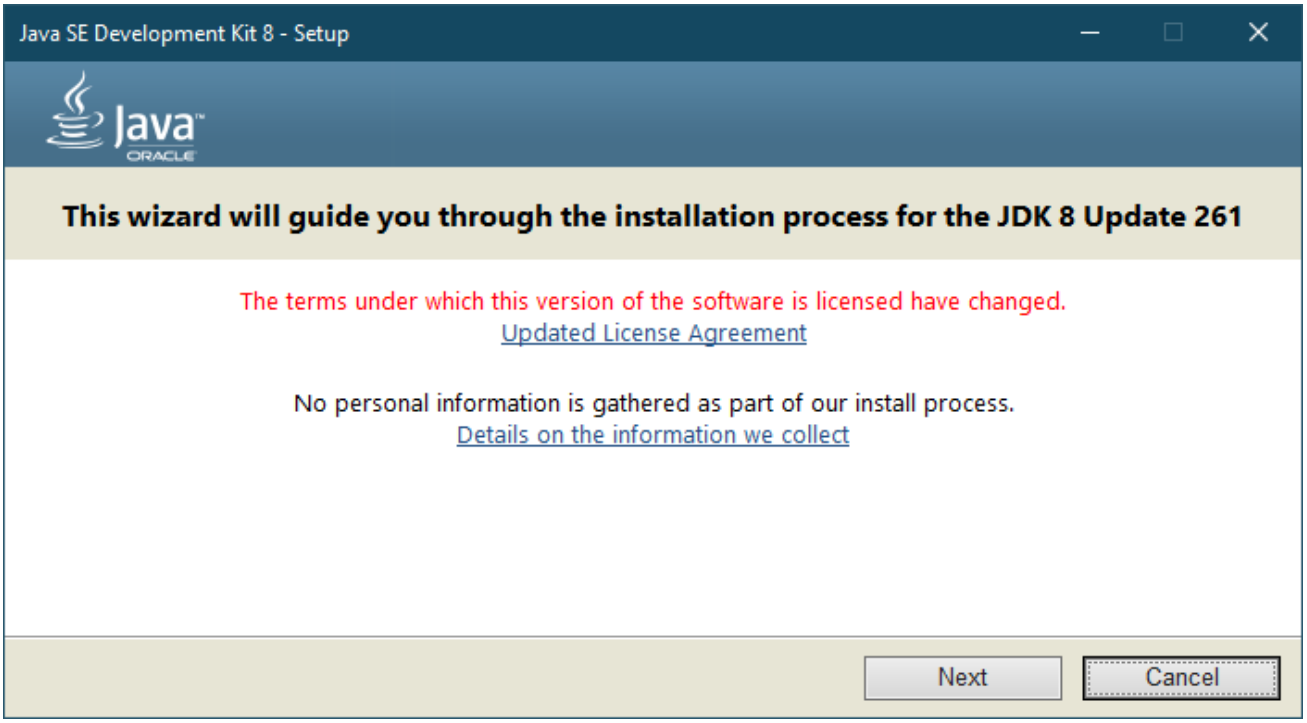



**Figure 3: Installing the Java Development Kit.**

In the next window, do not change any of the options. Pay particular attention to the folder in which the JDK will be installed. In the screenshot below, the installation folder for the JDK is:

```
C:\Program Files\Java\jdk1.8.0_261
```

This is the default installation folder. Note down this path – we will need it later.

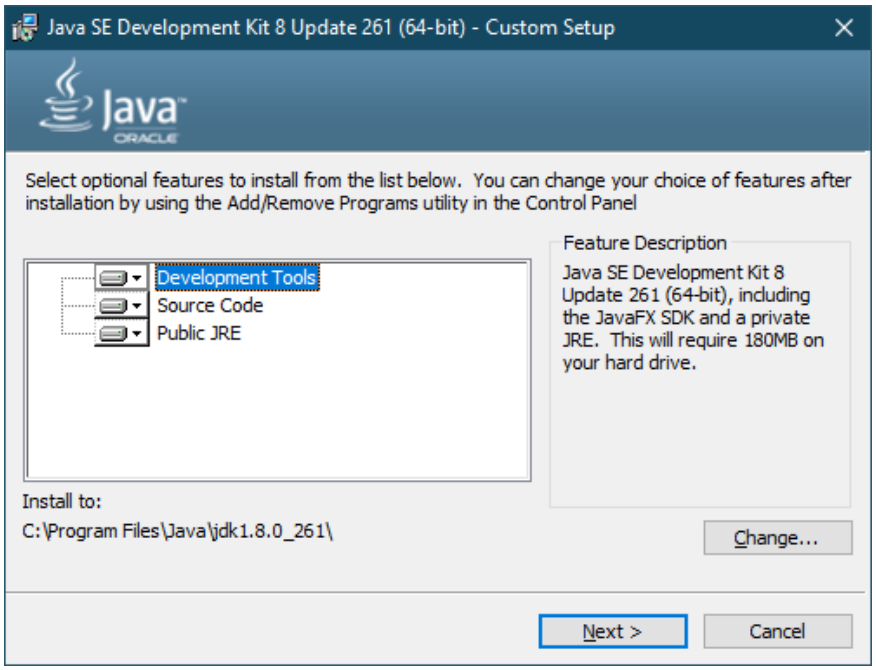



**Figure 4: Inspecting the installation path for Java.**

Click next to start the installation. After some files are copied, another installation screen will come up – click next to proceed through that as well:

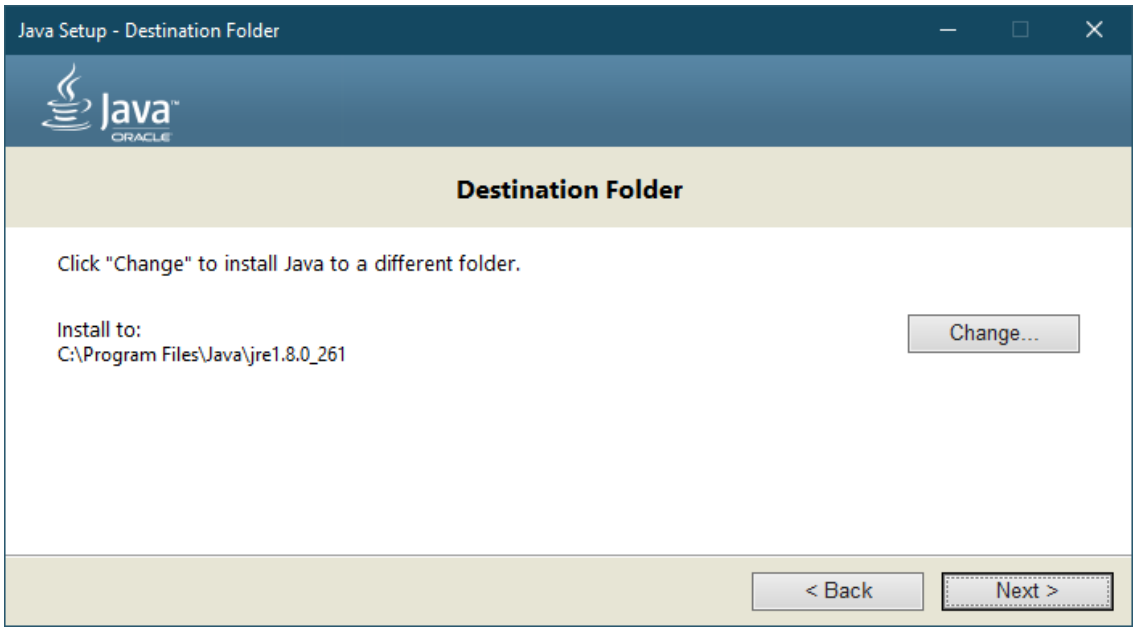
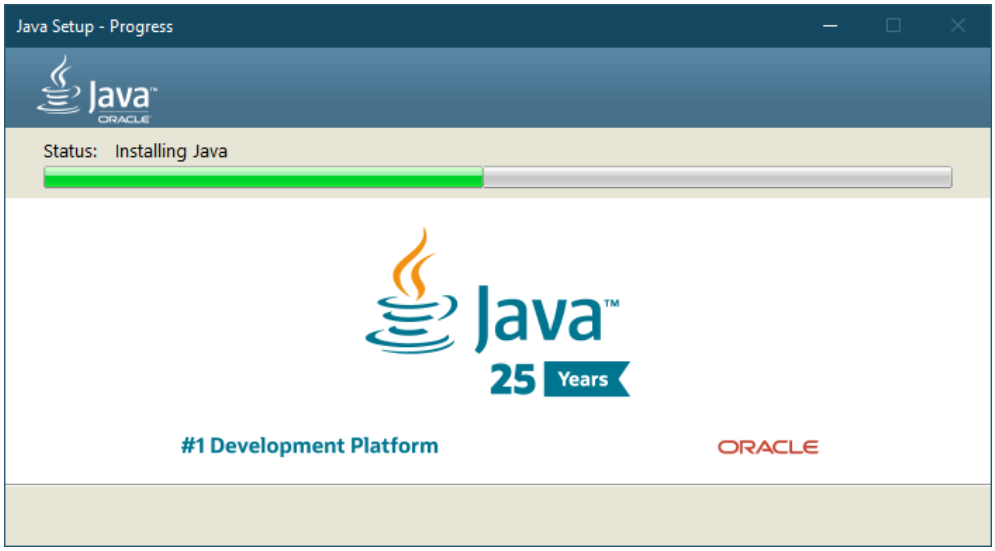



**Figure 5: Installing the Java Runtime Environment.**

Wait for the installer to complete its work:

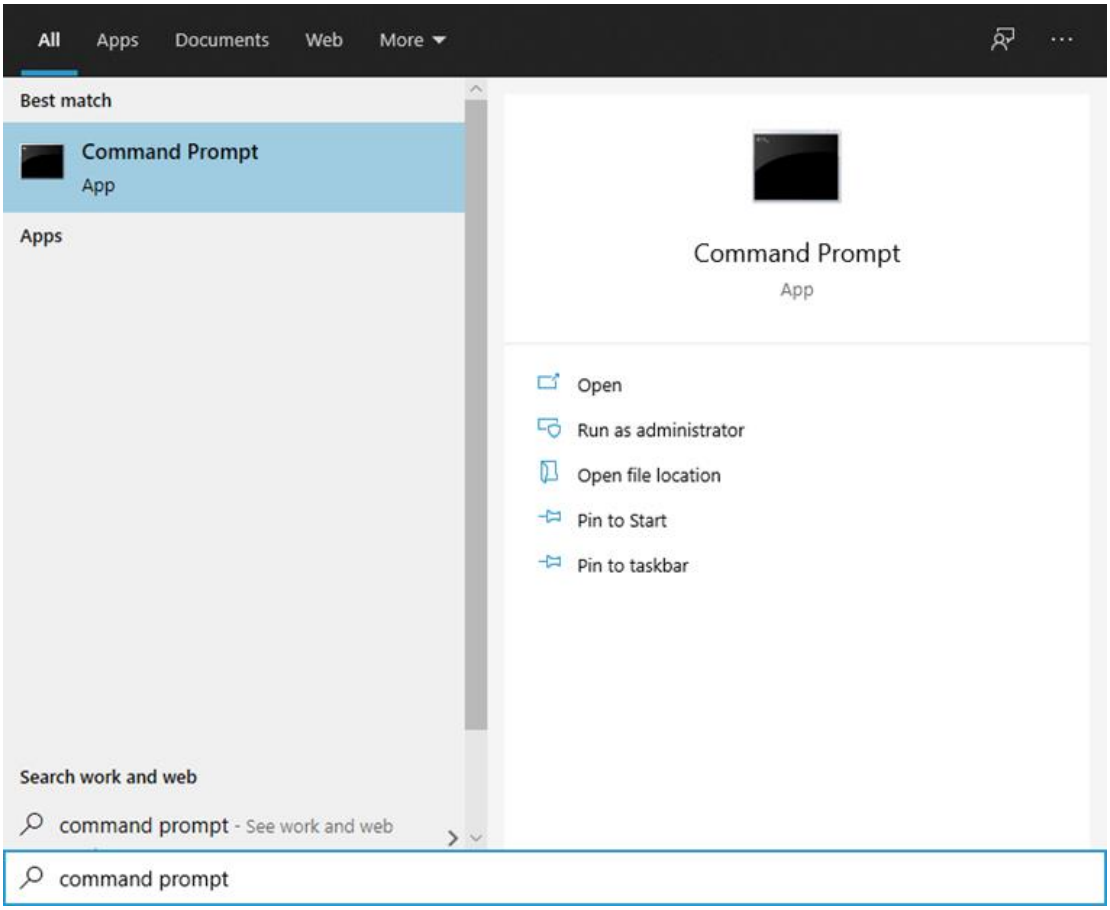



When the installation is done, click finish.

## Configuring the classpath

After installing Java, we must configure the classpath. This is an **environment variable** in operating systems that allows them to easily access the program which they need to run. For this course, we need to configure this environment variable so that Java can also be accessed through the command line.

The **command line** – known in Windows as Command Prompt – *is a text interface that can read commands and pass them to the operating system of your computer to run*. It is possible to navigate to folders on your computer, run programs or perform a variety of other tasks using the command prompt. To find the command prompt on your Windows computer, click on the start menu and search for the term "command prompt":




Click on the app icon to start the Command Prompt. If you are on a Linux or MacOS based computer, you will have to find and start the *Terminal*, which is the corresponding tool for these operating systems.

When started, the command prompt looks like this:

```
Microsoft Windows [Version 10.0.19042.867]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\user>
```

The command prompt executes commands and allows you to navigate between different folders/directories on your computer. When started, the command prompt will automatically navigate to the user's folder – in this case C:\Users\user. For you, it may be different, such as C:\Users\John or C:\Users\Christina. This is known as the *current path*.

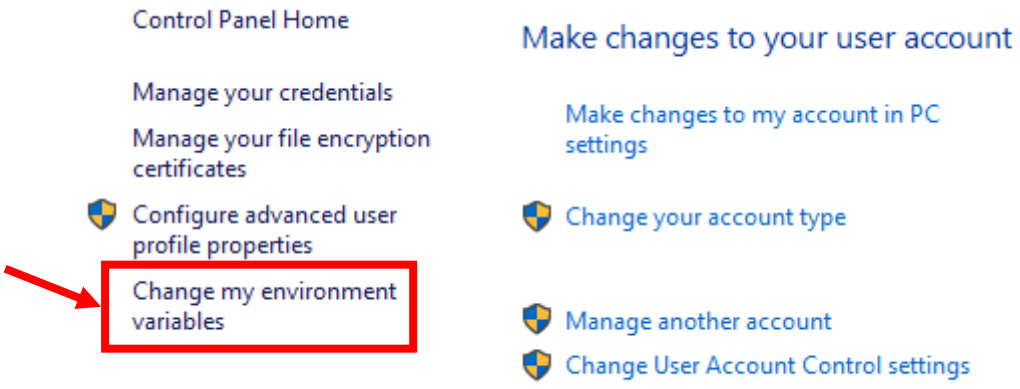
Commands can be written in the command prompt by typing them after the arrow symbol which follows the current path ( **>** ). To run Java programs, we use the command `java`. Type in this command in the command prompt and press Enter.

Unfortunately, even though we have installed Java, the operating system does not recognize this command:

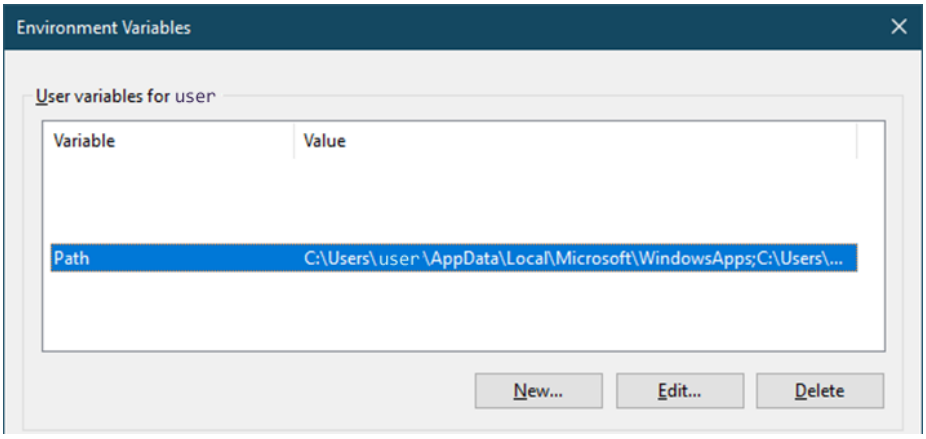
```
Microsoft Windows [Version 10.0.19042.867]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\user>java
'java' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\user>
```

This is because we have not configured our classpath yet, and the operating system cannot find this command. To fix this, we must change some settings on our computer through a series of steps:

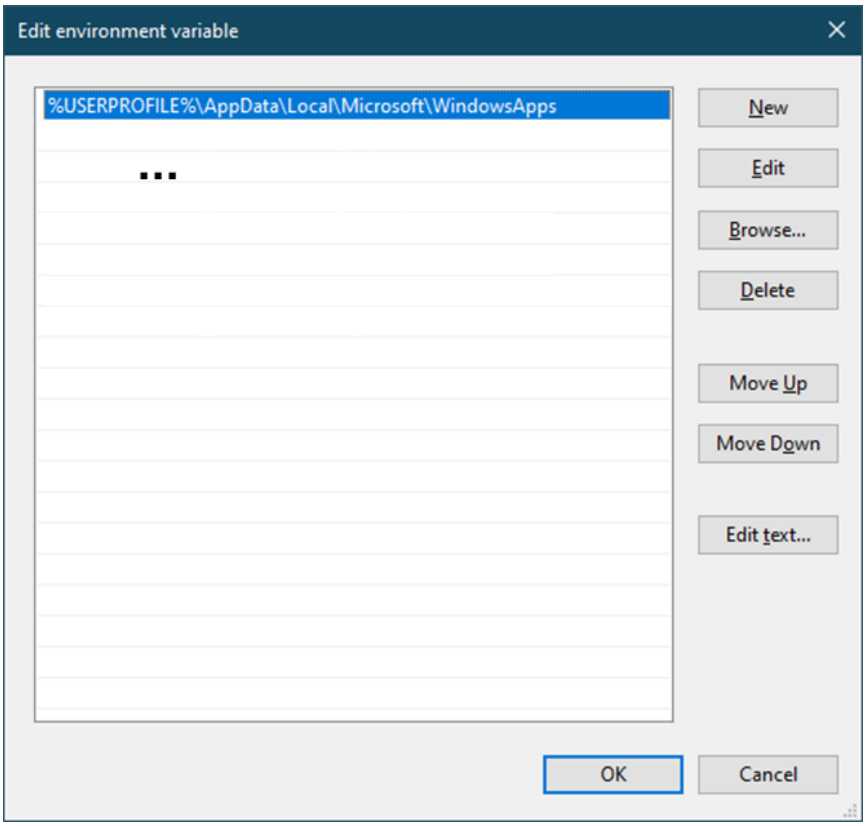
1) On Windows, find the Control Panel and then select User Accounts.
2) From the options displayed, find "Change my environment variables" on the left pane and select it.




3) From the User variables list, find the "Path" variable:




4) Select the "Path" variable and click the Edit... button. You will be presented with a list of paths for this variable.

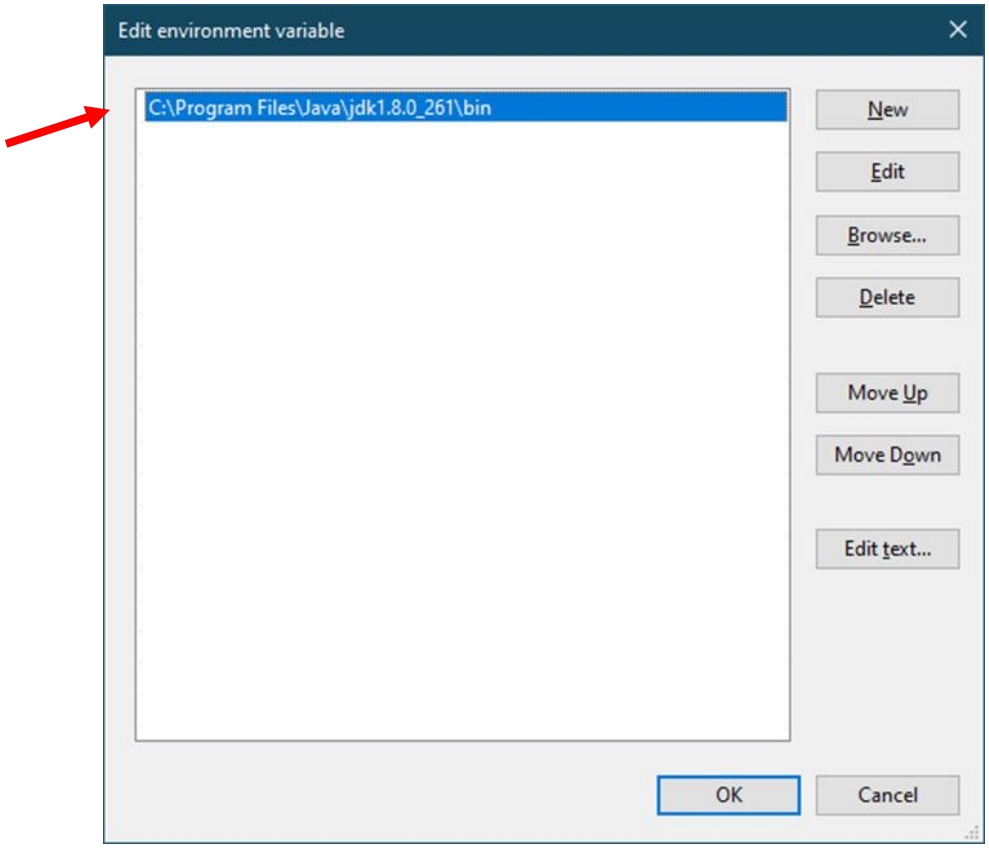



5) Ignore the rest of the entries and click "New".
6) Find the installation path of Java on your computer. It is usually located under C:\Program Files\Java. For example, on my computer it is:

C:\Program Files\Java\jdk1.8.0_261

7) In the entry, enter the path of your Java installation folder, but add an additional path to the folder `bin`. For example:

C:\Program Files\Java\jdk1.8.0_261**bin**




8) Press OK in all opened dialogs and then close the Control Panel.
9) Close any Command Prompts you have already opened.
10) Start the Command Prompt again, type the command "java" and press Enter.

If you have correctly configured the path by following the instructions above, the operating system should now recognize the command and show something like this:

```
Microsoft Windows [Version 10.0.19042.867]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\user>java
Usage: java [-options] class [args...]
           (to execute a class)
   or  java [-options] -jar jarfile [args...]
           (to execute a jar file)
where options include:
    -d32          use a 32-bit data model if available
    -d64          use a 64-bit data model if available
    -server       to select the "server" VM
                  The default VM is server.

    -cp <class search path of directories and zip/jar files>
    -classpath <class search path of directories and zip/jar files>
                  A ; separated list of directories, JAR archives,
                  and ZIP archives to search for class files.
    -D<name>=<value>
                  set a system property
    -verbose:[class|gc|jni]
```

## Familiarizing with the command prompt

You should already be familiar with the concept of files and folders in Windows and other operating systems. As mentioned before, the command prompt works based on the *current path*, which is a path to a specific folder. Using the command prompt we can manage files and folders and execute the commands needed to run our programs, but first let us take a look at how the command prompt enables the management of files.

## The dir command

One of the commands supported by the command prompt is the `dir` command. *This command shows the contents (files and folders) in the current path*. Try this out on your system:

1) Open a command prompt and type the command `dir`.
2) The output of this command will look similar to this:

```
C:\Users\user>dir
 Volume in drive C has no label.
 Volume Serial Number is 46AA-263B

 Directory of C:\Users\user

16-Nov-20  10:34    <DIR>          .
16-Nov-20  10:34    <DIR>          ..
27-Jul-20  14:36    <DIR>          3D Objects
23-Aug-20  17:50    <DIR>          Apple
13-Nov-20  17:49    <DIR>          CLionProjects
27-Jul-19  21:16                30 composer.bat
03-Mar-20  00:09               107 composer.json
03-Mar-20  00:11           127,997 composer.lock
27-Jul-19  21:08         1,915,161 composer.phar
16-Nov-20  10:34    <DIR>          Desktop
25-Jul-20  11:45    <DIR>          Documents
19-Mar-21  19:04    <DIR>          Downloads
```

This is a list of all the files in the current folder, which is your personal user folder. Note that there is various information about the files:

- The date and time of creation of the file or folder.
- Whether or not is it a directory (folder) – indicated by `<DIR>`.
- The size of a file in bytes.
- The name of the file or folder.

The `dir` command is useful because it allows us to see what the current folder contains. In turn, we can know the contents of the folder we are in, and therefore if any of our programs are there. As programmers, it is important to feel comfortable with and know how to use the command line.

The use of the `dir` command is not limited to the current directory. We can also specify the name of another folder within the current path, and the command will list the files and folder in that folder. For example, given that there is a folder called Desktop in our current folder, we can list all the files in the Desktop folder:

```
>dir Desktop
```

```
C:\Users\user>dir Desktop
```

```
 Volume in drive C has no label.
 Volume Serial Number is 46AA-263B

 Directory of C:\Users\user\Desktop

16-Nov-20  10:34    <DIR>          .
16-Nov-20  10:34    <DIR>          ..
18-Nov-20  23:57    <DIR>          Assignment
25-Oct-19  21:30    <DIR>          flutter
14-Nov-19  00:58    <DIR>          java-docs-samples
09-Jun-20  16:08    <DIR>          Rabbits
22-Sep-20  10:22    <DIR>          stuff
29-Jun-20  10:59    <DIR>          SwingTest
08-Oct-19  15:33    <DIR>          Test
09-Oct-19  23:21    <DIR>          untitled
               1 File(s)         15,741 bytes
              16 Dir(s)  27,237,474,304 bytes free
```

This lists all the files that are on the desktop.

## The cd command

The `cd` command stands for change directory and allows us to change the current path to a given directory. We must specify which directory we would like to move to, after this command. For example, to move from the current path to the Desktop, we can do:

```
>cd Desktop
```

```
C:\Users\user>cd Desktop

C:\Users\user\Desktop>
```

Notice how the current path has changed to `C:\Users\user\Desktop` after running this command. Try this on your computer as well.

It is also possible to provide an entire path to this command. For example, if we know there is a folder called Rabbits within the Desktop folder, we can directly run the following command to change the directory to that folder in one step:

```
>cd Desktop\Rabbits
```

The path given to the command line is known as a relative path, because it is relative to the current path (C:\Users\user\ + Desktop\Rabbits = C:\Users\user\Desktop\Rabbits).

We have seen how it is possible to move forward in a path and into other folders within the current path. We can also go backward – visiting the parent folder of the current path. We can do this by running the `cd` command but instead of giving it the name of the folder to move to, we can give it a special directory symbol called the two dots (**..**). This symbolizes that we are referring to the *parent directory* of the current path.

Given the current directory is C:\Users\user\Desktop, we can therefore move back to the user's folder by running:

```
C:\Users\user\Desktop>cd ..
C:\Users\user>
```

Notice how the path has changed back to the user's folder.

> **Note**
>
> On Linux and MacOS operating systems, the equivalent of the `dir` command is `ls`. The `cd` command is the same on Linux/MacOS and Windows.

## Changing drives

Finally, it may be useful at times to be able to switch between different hard drives on your computer. This is not possible using the `cd` command. Instead, we can use the name of the drive we would like to change to. Given that we know that a drive with the letter D: exists on our computer, we can switch to this drive in command prompt by simply entering the drive's name followed by a colon (:). For example, to switch to drive D, we can use:

```
C:\Users\user>D:

D:\>
```

Note how after running this command the path of the command line changes to the drive D. To explore the contents of this drive, we can use the `dir` and `cd` commands.

## Creating files

It is also possible to create a file using the command line and place some text inside it. To do this, we can use the `echo` command. This command allows us to write a piece of text to a specified file. Let us first move back to the C: drive and into the Desktop folder:

```
D:\>C:

C:\Users\user>cd Desktop

C:\Users\user\Desktop>
```

It is time to create our file on the Desktop (where we can easily access it). The name of the file will be "MyFile.txt"

> **Information**
>
> There is a lot more to the Windows command line. Feel free to explore the `dir` and `cd` commands and learn about other commands as well.

# Creating your first program

## Creating, writing, and saving text in simple files

As we have mentioned in the previous sections, computer programs are written in a programming language. Such a program is no different than any other piece of text. You may already be familiar with some of the text editors available in various operating systems, such as Notepad, Sublime, Gedit, and many more. These can be used to write text in files. To demonstrate how this works, you can follow these instructions:

1. Right-click on your desktop and select New, and then Text Document.

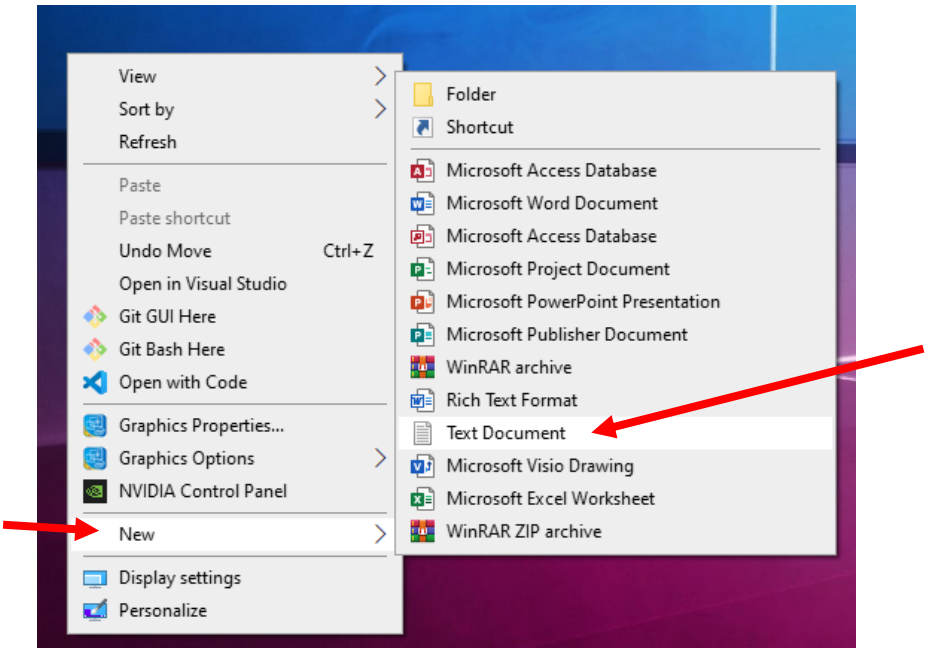
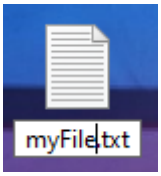



**Figure 6: Creating a text document in Windows 10.**

2. Give the name 'myFile' to your text document file:




3. Notice the extension .txt at the file's end. A file's ***extension*** is a suffix, entered after its name, which identifies the type of the file. Text files have the extension '.txt'. Other types of extensions are: '.doc' for Microsoft Word documents, '.html' for web pages, and '.pdf' for PDF documents. If this extension is not visible on your computer, you can follow a simple procedure to enable it. As a programmer, it is often useful to know the extensions of files.
    a. To make file extensions visible in Windows 10, you can use the start menu to search for the term 'extensions'.

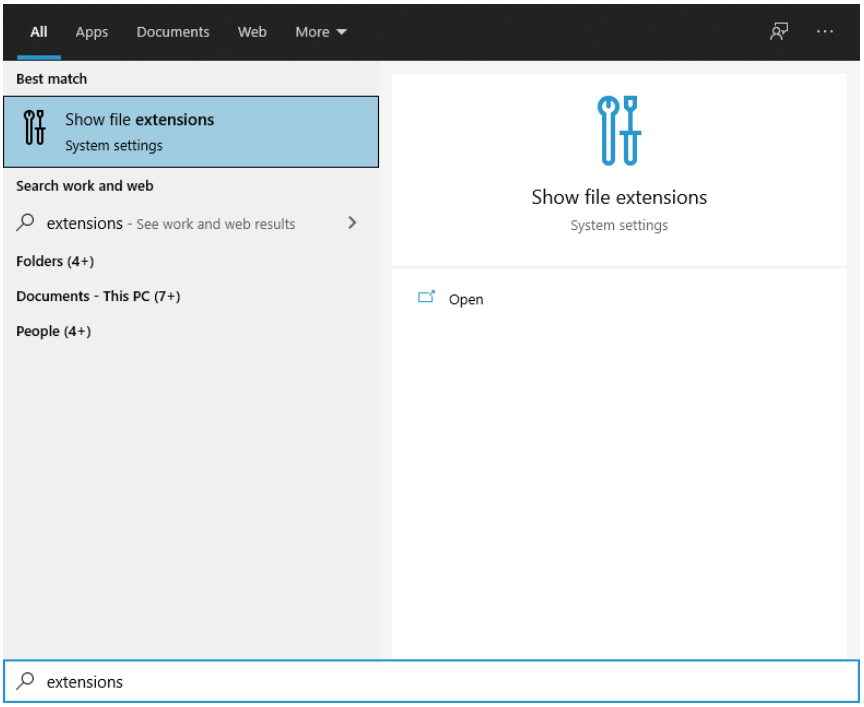
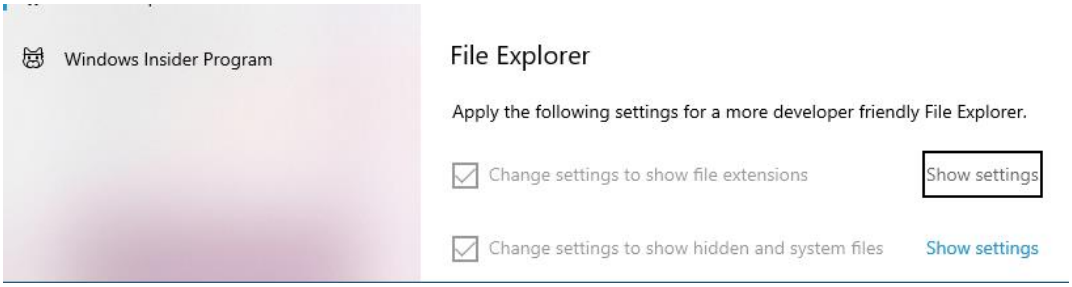



**Figure 7: Searching for extension settings.**

b. After starting the system settings, find the File explorer settings and select 'Show settings'.



c. When the explorer settings window is opened, find the entry 'Hide extensions for known file types. This option will be enabled by default in Windows. If you would like to have file extensions enabled, make sure that this option is disabled. Click OK when you are done.

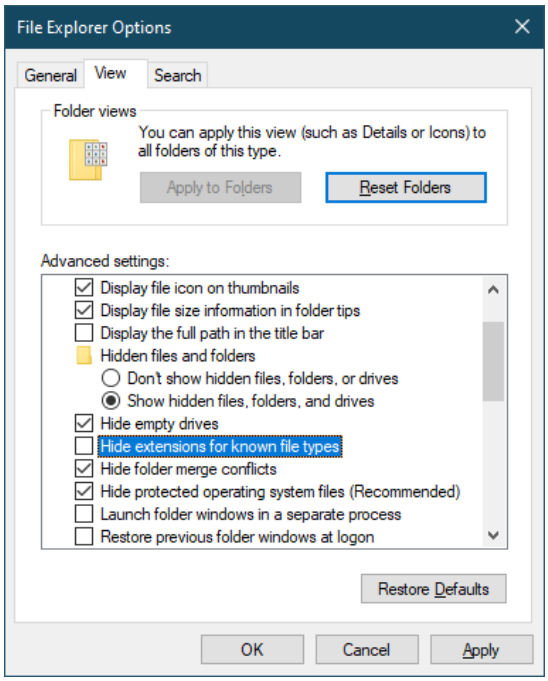



**Figure 8: Disabling the hide extensions option.**

**Extensions**

Having extensions enabled (visible) in files allows us to quickly see what type of files they are. Programmers usually deal with a variety of files, so it is useful to recognize their types quickly.

4. Double click your text file to open it in Notepad. You should be presented with a blank, white area in which you can write text. This is called a **text editor**. Text editors are programs that are used to write text, including computer programs.

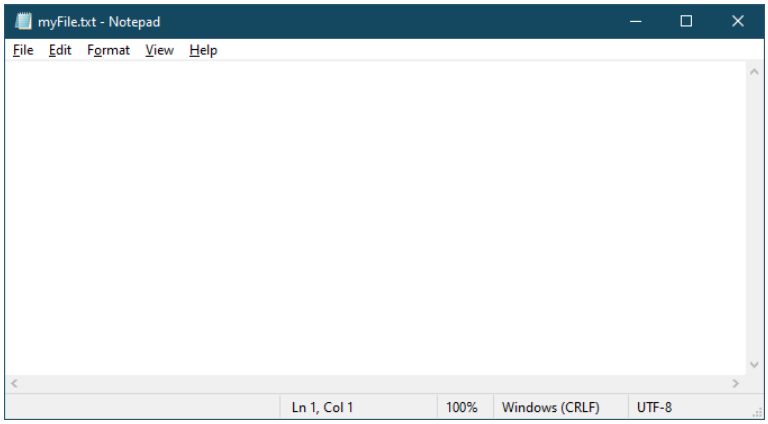



**Figure 9: Opening a text document in Notepad.**

5. Write your first and last name in the file and then select 'Save' from the 'File' menu.

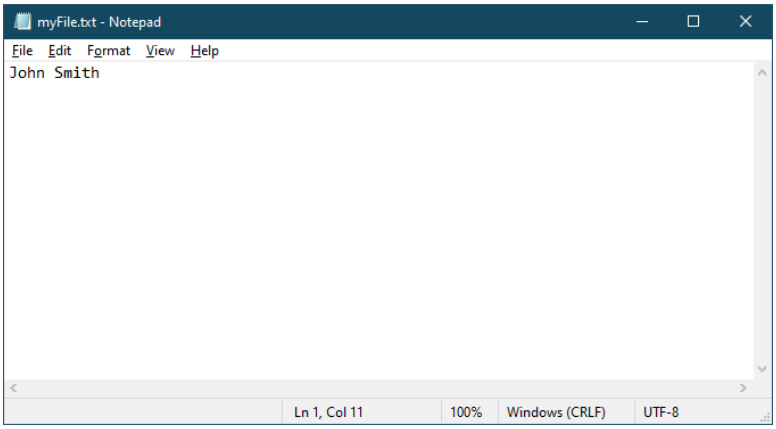



**Figure 10: Writing content in a text editor.**

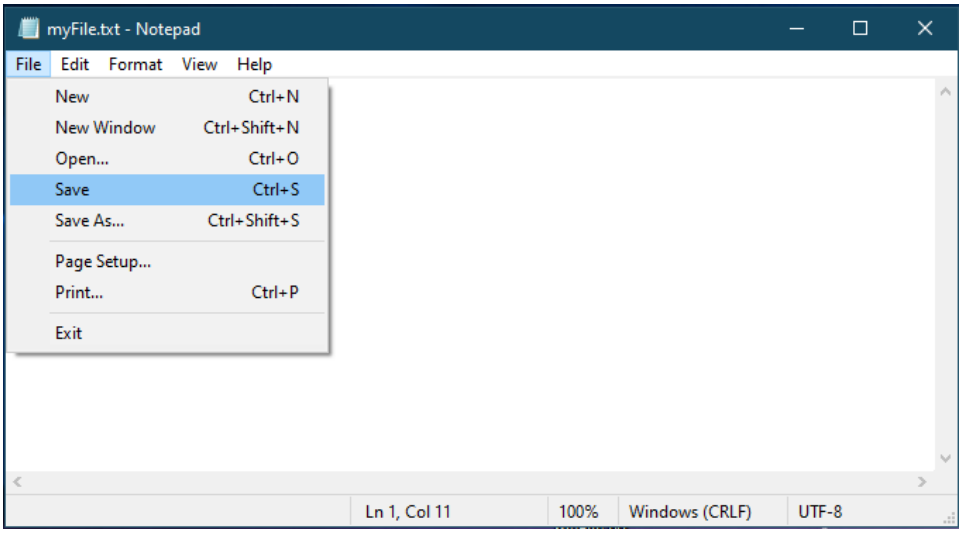



**Figure 11: Saving a text document.**

6. Close Notepad and then re-open the file by double-clicking it. Your text should be saved in the file.

## Creating a computer program file

The steps involved in creating a file that stores a computer program are the same as those for creating a simple text file since a computer program is just text, written in a specific programming language.

1. To create your first computer program, create a text file called 'MyFirstProgram.txt'.

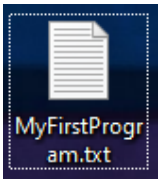



**Figure 12: Creating a text file for a program.**

This file will store a program written in Java. Files that contain Java code have a different extension that distinguishes them from normal text files. The extension for files written in Java is '.java'. We must therefore change the extension of this file to '.java' instead of '.txt'.

2. To do this, right-click on the file and select "Rename…" (or simply press **F2** which is a *shortcut* for renaming a file).

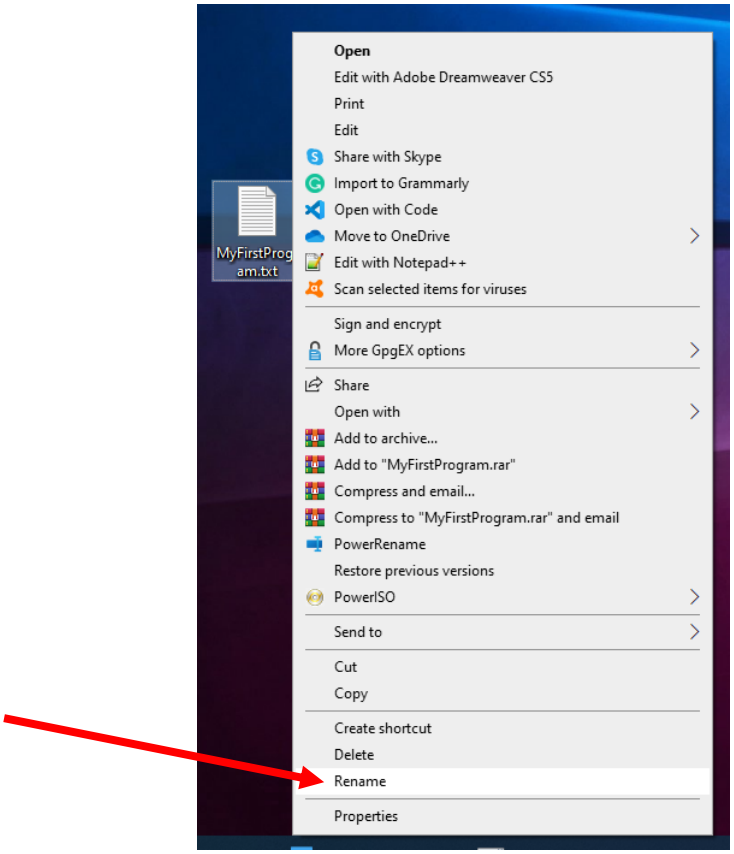



**Figure 13: Renaming a file.**

3. Move the cursor past the dot which starts the extension prefix. Change 'txt' into 'java' and press Enter to rename the file.

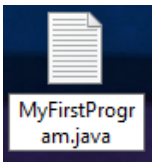



**Figure 14: Changing a file's extension.**

4. A dialog will appear asking you to confirm whether you would like to change the extension of this file to .java. We want this file to be a Java file, so click Yes to change the extension.

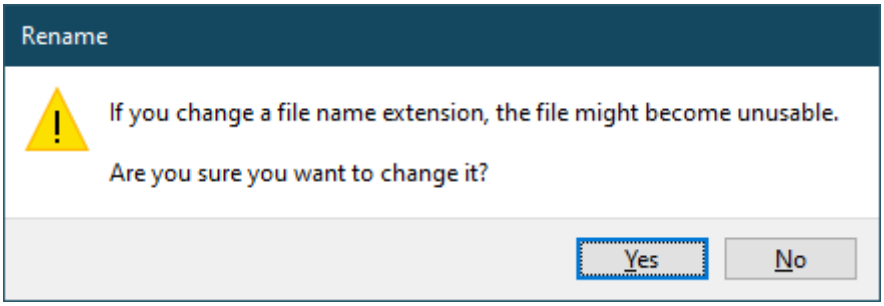
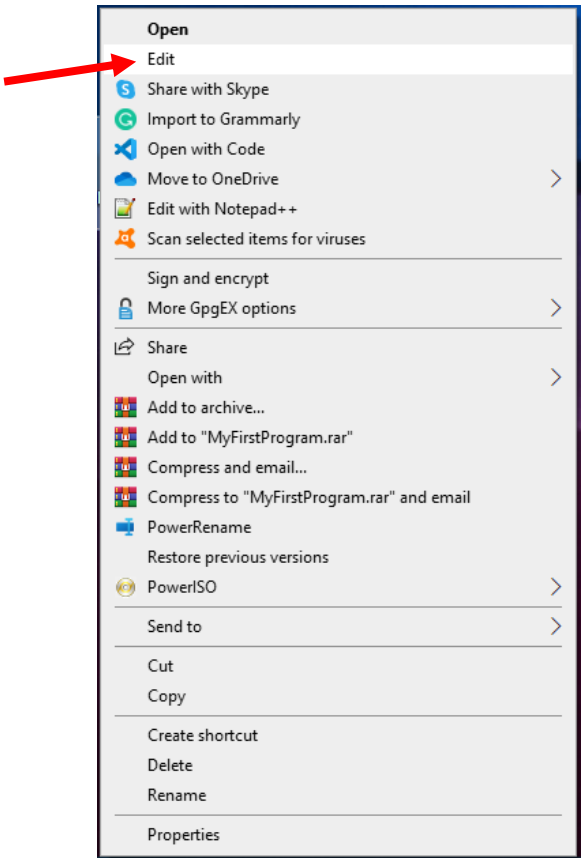



**Figure 15: Confirming the extension change.**

5. Changing the extension of a file from '.txt' to '.java' will cause the file to not open when you double-click on it. Instead, to open the file now, right-click on it and select "Edit".




You have now created a Java file. In the next steps, you will write your first program within this file.

## Writing your first program

The following code creates a program called 'MyFirstProgram'. You do not need to know what this code resembles and how it works yet.

```
public class MyFirstProgram {

    public static void main(String[] args) {

        System.out.println("This is my first program!");

    }

}
```

1. Open the file you created (MyFirstProgram.java) by right-clicking on it and selecting "Edit". Then, copy and paste the code above into the text editor.

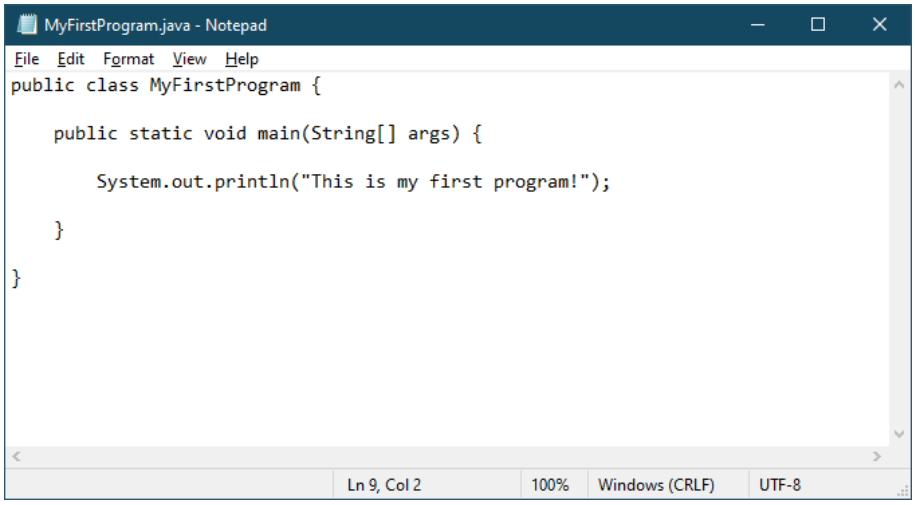



**Figure 16: Copy and paste the code into the text editor.**

You have now written your first program in Java. Click File and then 'Save' (or press CTRL + S, which is the shortcut to save a file) to save the program. As we mentioned before, programs that are written in high-level languages cannot be directly executed by a computer and need to be translated or **compiled** first, into machine language.

**A Java file (.java)** *is a source code file, which can be used to create code but not run it.*

## Compiling a program

To compile our program, we need to start up the Command Prompt. If you have done everything correctly in the third section (Setting up), then you should have access to a command called `javac`. This command stands for Java Compile and is used to compile '.java' files into machine code which the computer can execute. The `javac` command expects us to also provide the name of the file to be compiled.

1. Open a Command Prompt and use the `dir` and `cd` commands we have talked about before to navigate to the folder containing the program you created.

```
D:\Users\user>cd Desktop

D:\Users\user\Desktop>dir
 Volume in drive D is Windows HDD
 Volume Serial Number is 561E-D905

 Directory of D:\Users\user\Desktop

29-Mar-21  15:24    <DIR>          .
29-Mar-21  15:24    <DIR>          ..
08-Feb-21  13:49    <DIR>          bin
23-Mar-21  22:55             4,767 context.txt
07-Feb-21  12:42    <DIR>          ExampleProject
29-Mar-21  15:09                10 myFile.txt
29-Mar-21  15:36               151 MyFirstProgram.java
              10 File(s)      4,952,961 bytes
              11 Dir(s)  66,151,546,880 bytes free
```

2. Once you have located your file, you can run the `javac` command to compile it:

```
javac <ProgramName>.java
```

where `<ProgramName>` is the name of the file in which your program is stored. Therefore, we need to run the following command:

```
D:\Users\user\Desktop>javac MyFirstProgram.java

D:\Users\user\Desktop>
```

If the file compiles successfully, there should be no additional printed text in the console. When the program is compiled, a new file is created within the same folder as your code. Use of the `dir` command to inspect the contents of the current path:

```
D:\Users\user\Desktop>dir
 Volume in drive D is Windows HDD
 Volume Serial Number is 561E-D905

 Directory of D:\Users\user\Desktop

29-Mar-21  15:48    <DIR>          .
29-Mar-21  15:48    <DIR>          ..
08-Feb-21  13:49    <DIR>          bin
23-Mar-21  22:55             4,767 context.txt
07-Feb-21  12:42    <DIR>          ExampleProject
29-Mar-21  15:09                10 myFile.txt
29-Mar-21  15:48               447 MyFirstProgram.class
29-Mar-21  15:36               151 MyFirstProgram.java
              11 File(s)      4,953,408 bytes
              11 Dir(s)  66,151,546,880 bytes free
```

3. Note that the Command Prompt output indicates that a new file was created, with the same name as your program file, but with a different extension of '.class'.

**A Class file (.class)** *is a compiled version of a Java file* that can be executed by the computer.

**The Java Virtual Machine**

The class files created by the `javac` command are not executed by the computer's processor directly. Instead, they are executed by another, low-level program called the Java Virtual Machine (JVM), which is responsible for running Java programs on the computer's processor. By using this technique Java can run the same code on a huge variety of processors, thus allowing programs written in Java to be easily executed across a huge variety of devices.

## Running a program

We have so far written our program in Java and compiled it into a Class file. Now it is time to execute our program. To do this, we will use the Command Prompt once again. In this case, we need to call the `java` command *which is used to execute compiled Java programs*.

1. In the command prompt, we can execute the following command:

```
java <ProgramName>
```

where `<ProgramName>` is the name of our program, *without any extension*.

```
D:\Users\user\Desktop>java MyFirstProgram
This is my first program!

D:\Users\user\Desktop>
```

Executing this command for our program will print out the text "This is my first program!" on the command line.

**The console**

The general term for a Command Line or Terminal is **console**. From now on, we will refer to the Command Prompt as the console.

**Congratulations, you have successfully written, compiled, and executed your first Java program!**

# Integrated Development Environments

You may think that the process of creating a new program is rather tedious, and you might be right. To create a very simple program that just outputs a piece of text on the console, we have gone through many steps that took a lot of time to complete. Fortunately, we do not have to do this whole process every single time we create a new program.

**Integrated Development Environments**, or shortly **IDEs**, enable programmers to easily create, manage, compile, and run computer programs without having to manually do all these steps. An IDE is a software application itself that provides a large variety of useful tools for computer programmers to do their job.

## Tools in IDEs

An IDE may include some of the following tools:

- A **code editor** that allows code to be written, viewed and edited.
- A **compiler** that automatically compiles programs without the need to manually run commands on the console.
- A **class browser** that allows programmers to inspect the contents of specific source code files.
- **Support for Version Control Systems** (VCS), which enables software developers to work collaboratively.
- A **Debugger**, which allows programs to be stopped at specific points for inspection.
- A **Profiler**, which measures the performance of programs.

## IDE features

An IDE will generally support the following features, which make programming much easier to do:

- **Syntax highlighting**, which can clearly show the structure of the code and important keywords in the code with distinct colors and other font effects. One of the most useful aspects of this feature is the ability of the IDE to detect errors in the code, often underline them with a red line, and suggest possible solutions.
- **Code completion**, which can speed up programming by suggesting useful options to complete a partially written piece of code.
- **Code search**, which allows programmers to quickly search through code to find references of a specific name.

## Examples of commercially used IDEs

Some examples of IDEs that are currently used in the industry are:

- IntelliJ IDEA, one of the most popular Java IDEs.
- Visual Studio, used mainly for developing applications running on Visual Basic, C++, C# and lately other languages as well.
- NetBeans, a general-purpose IDE.
- Eclipse, a general-purpose IDE.
- JetBrains WebStorm, an IDE used mainly for front-end web development.
- JetBrains PhpStorm, an IDE used for creating PHP programs and scripts.

# IntelliJ IDEA

## Downloading IntelliJ IDEA

For this course, we will download and use an IDE called IntelliJ IDEA. IntelliJ IDEA is the most popular IDE for programming in Java and has a free community version. Our first step is to download IntelliJ IDEA by visiting this link, clicking 'Download', and then downloading the **Community** edition:

**Download IntelliJ IDEA**

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

## Installing and configuring IntelliJ IDEA

When your download is finished, click on the file to start the installation.

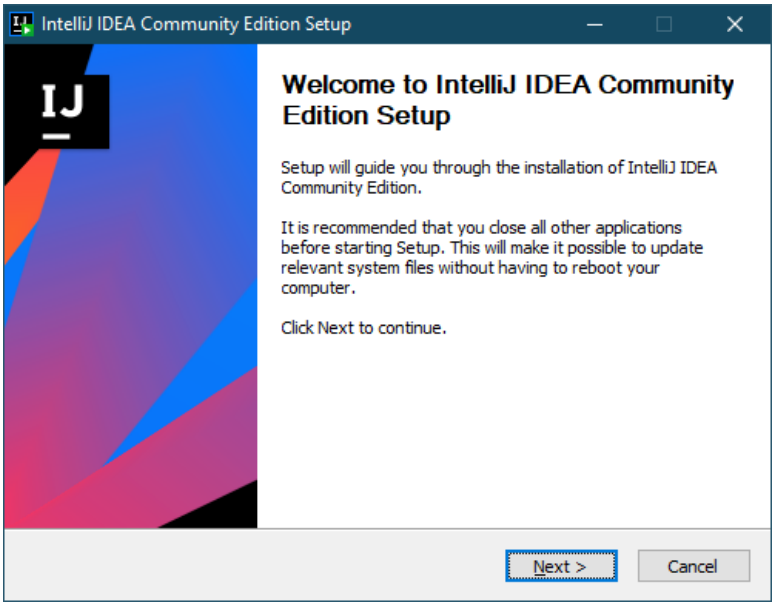



**Figure 17: Installing IntelliJ IDEA.**

1. Click next on the first installation window to proceed.
2. Confirm the destination folder for the installation and click Next.

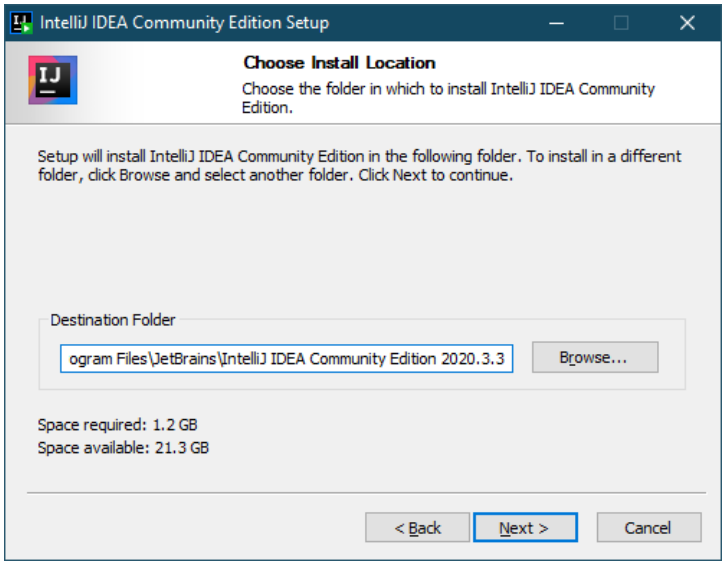



**Figure 18: Confirming the installation directory for IntelliJ IDEA.**

3. Then select the 64-bit launcher to be created on the desktop and click Next.

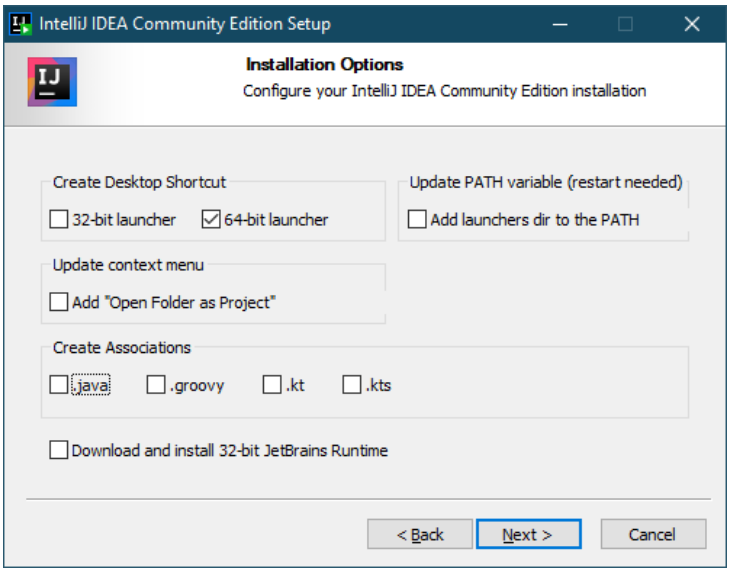



**Figure 19: Selecting installation options.**

4. Click install in the next window to install IntelliJ IDEA.
5. When the installation is complete, run IntelliJ IDEA by clicking the shortcut on the desktop or by searching for it in the Start menu.
6. Accept the privacy policy to continue.

When this process is completed, the start window of IntelliJ IDEA will be displayed:

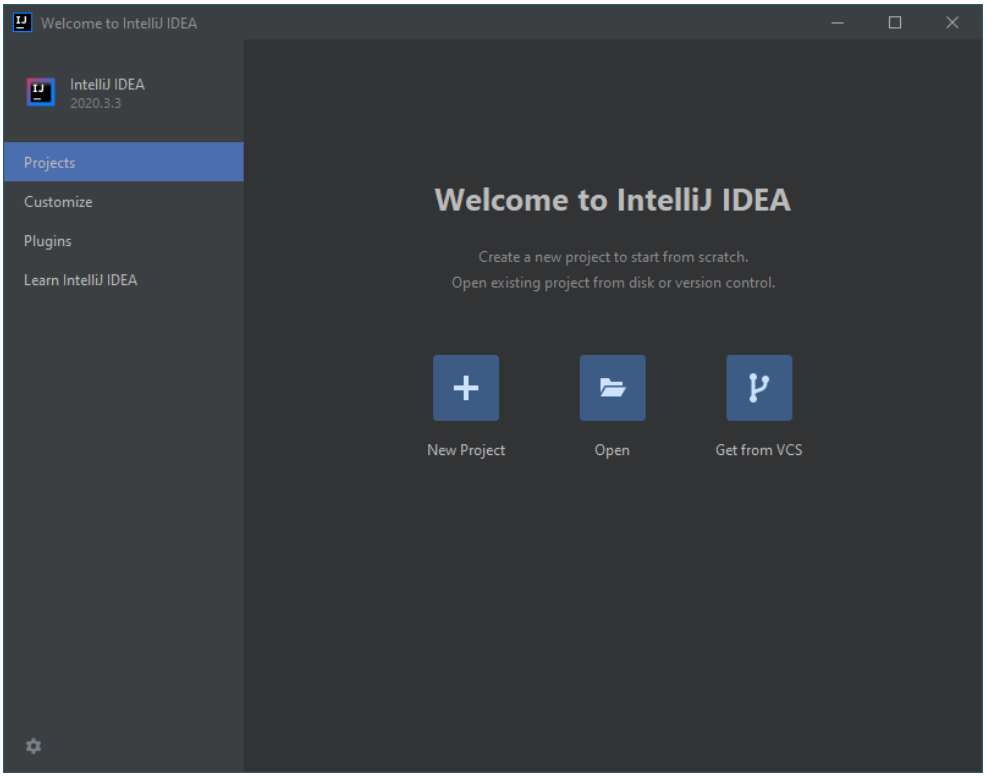



**Figure 20: The main screen of IntelliJ IDEA.**

This start screen allows you to create a new IntelliJ IDEA project, open an existing project and change the IDE's settings. We will not modify any settings for now. Instead, the task for the remainder of this worksheet is to simply create and run a simple "Hello World" program.

1. Click on "New project" in the main screen to create a new project.

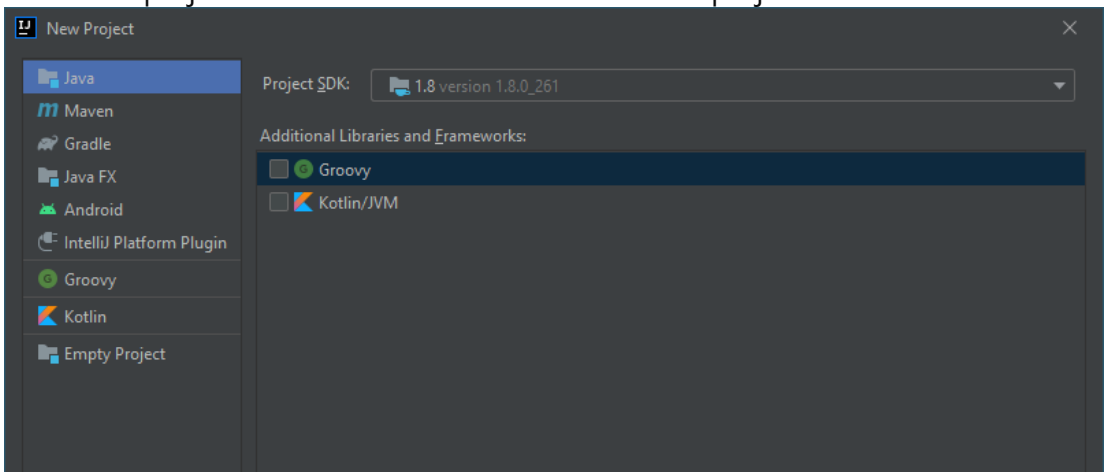



**Figure 21: Creating a new project.**

2. Leave the existing options selected (Java from the left menu and no options selected from the middle menu).
3. Pay particular attention to the project SDK dropdown menu at the top of the window. If an SDK is selected by default, you can proceed to the next step. Otherwise, we need to find the location of our Java JDK and add it to IntelliJ.

## Adding the Java JDK to IntelliJ IDEA

You may skip this section if a JDK was already selected in IntelliJ.

1. To add the Java JDK, you installed previously in IntelliJ IDEA, click on the Project SDK dropdown menu and select 'Add JDK'.

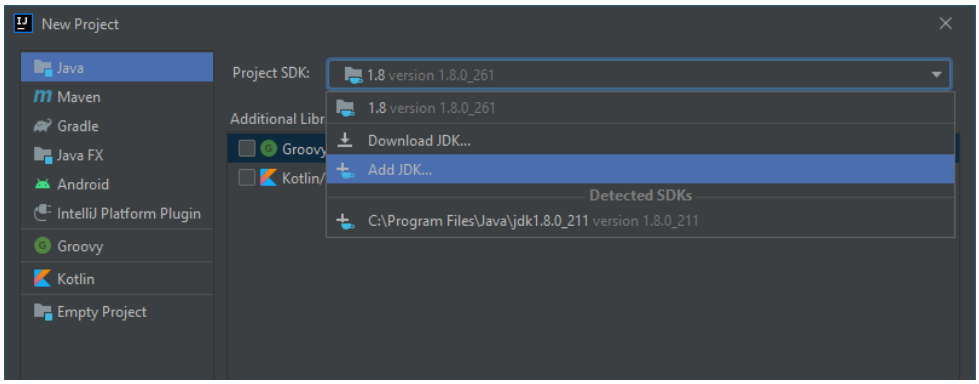



**Figure 22: Adding a new JDK.**

2. In the dialog presented, find the installation location of your Java JDK. The JDK is typically located at C:\Program Files\Java. Select the appropriate subfolder containing the JDK installation (e.g., for me it was "jdk1.8.0_211").

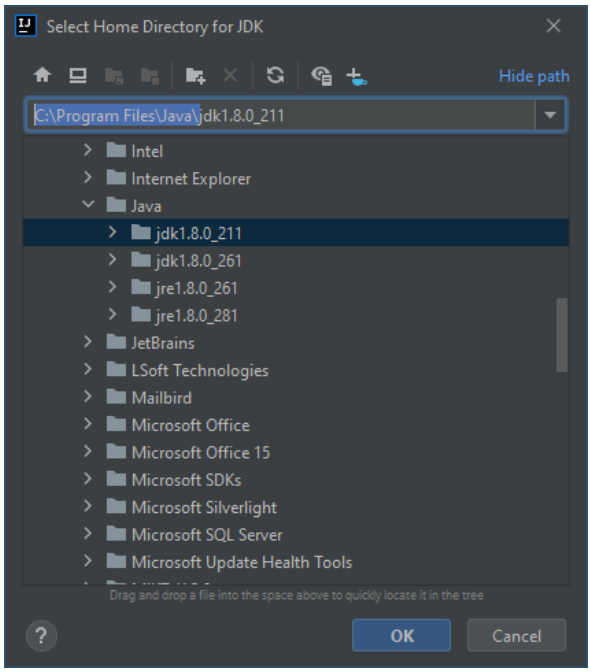



**Figure 23: Locating the installation folder of the JDK.**

3. Click OK to add the JDK. If you selected the correct folder, the new project window will add the entry in the available SDKs. Click next to continue.

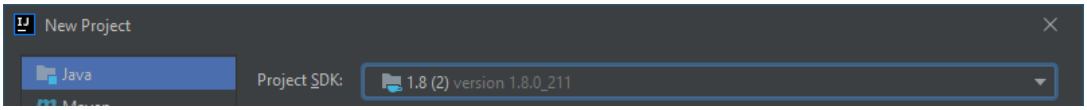



**Figure 24: The selected JDK should be added to the list of available project SDKs.**

## Creating a new project

On the next screen, leave the default options and press Next.

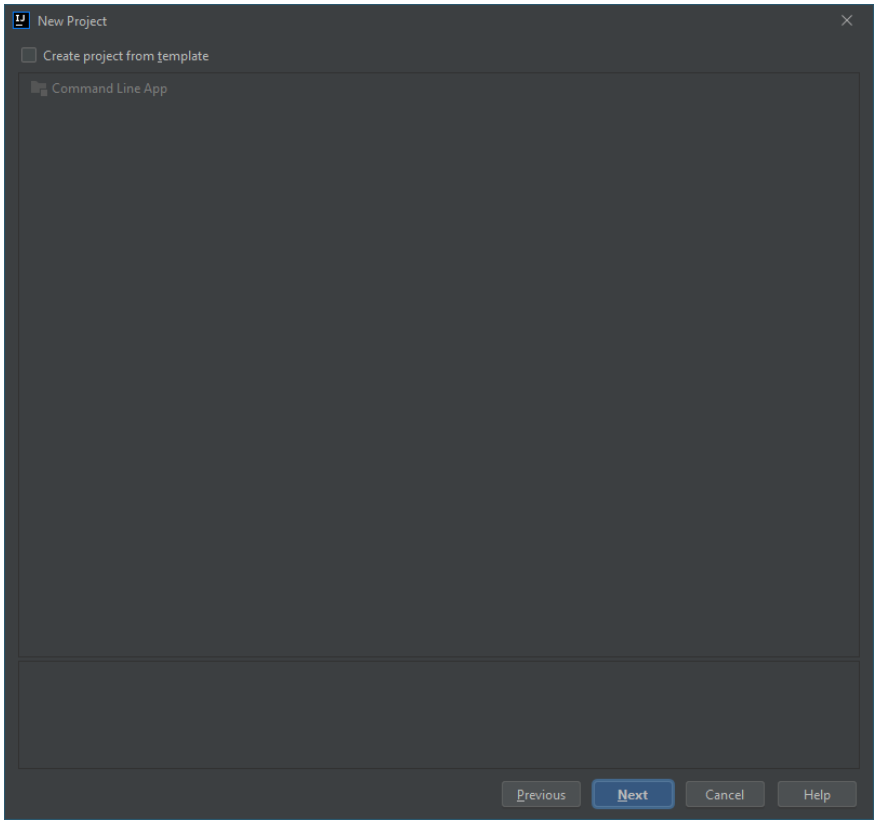



**Figure 25: Leave the default options selected.**

After clicking next, you will be asked to give a name to your project. Enter the name "HelloWorld" and select a location where you would like to store the project files. Ideally, it should be a location where you can easily access the files.


**Figure 26: Giving a name to the project and selecting a location.**

When you click Next, IntelliJ IDEA should create your project and start indexing your JDK. Allow it to complete the indexing process and you should be able to see the following:

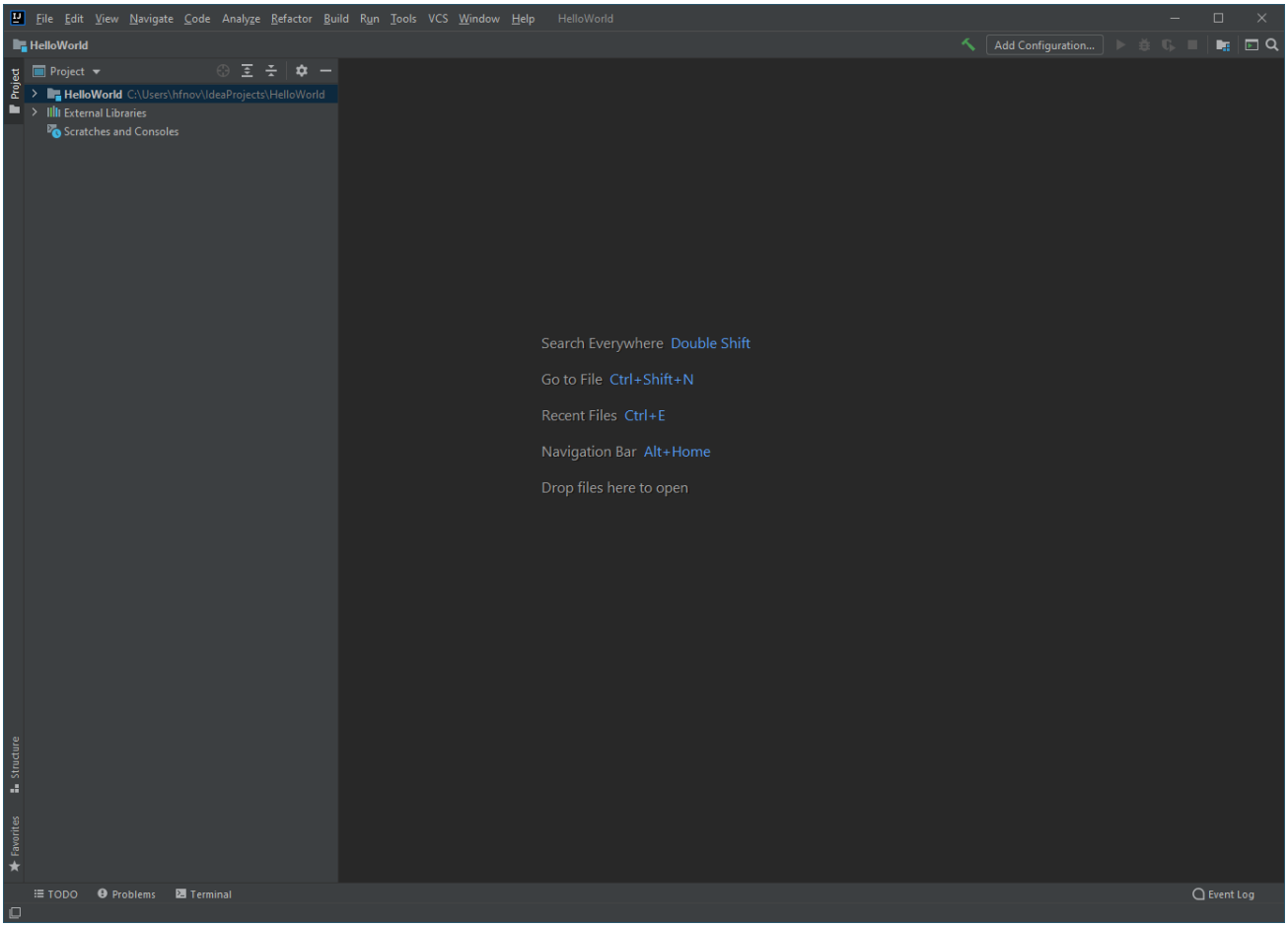



**Figure 27: The project view of IntelliJ IDEA.**

## The project view

On the left-hand side of the project window, you should see a menu called "Project". This menu allows you to explore the project and all of the files within it. By default, there is a folder called HelloWorld – with the same name as your project. This is called the **project folder**.

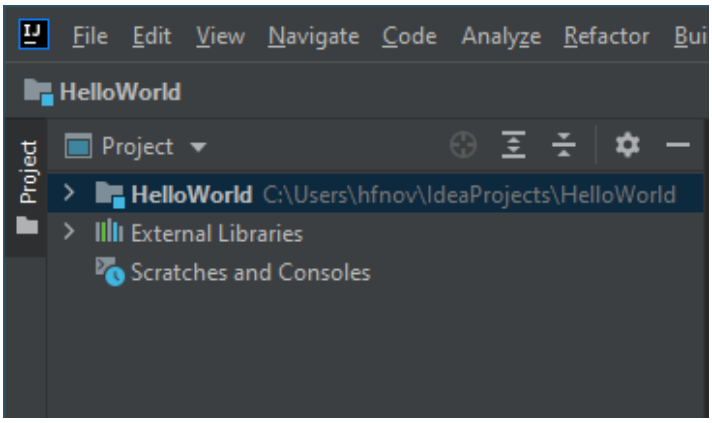



**Figure 28: The project folder.**

We can expand the project folder to view its contents by double clicking on it or clicking the arrow pointing towards the right on the left of its name. Upon expanding this folder, we can see that there are two other folders called '.idea' and 'src' inside it, as well as a file called 'HelloWorld.iml'.

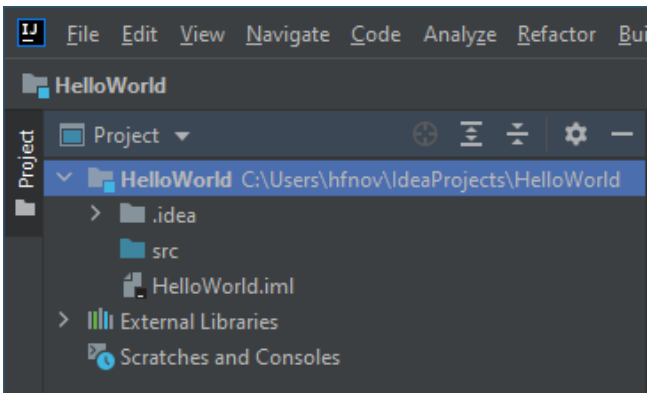



**Figure 29: Expanding the project folder.**

The `.idea` folder is a folder containing configuration files for this project. You *should not open or edit any of the files in this folder*.

The `src` folder is an empty folder that is marked with a blue color, which indicates that it can contain *source code files*.

The `HelloWorld.iml` file is an IntelliJ module file that contains the configuration about this project/module you created. *You should not try to open or edit the contents of this file*.

## Creating a Java file

IntelliJ IDEA allows us to quickly create a Java file by right-clicking on the `src` folder (in which all of the source code exists) and then selecting New, Java class.

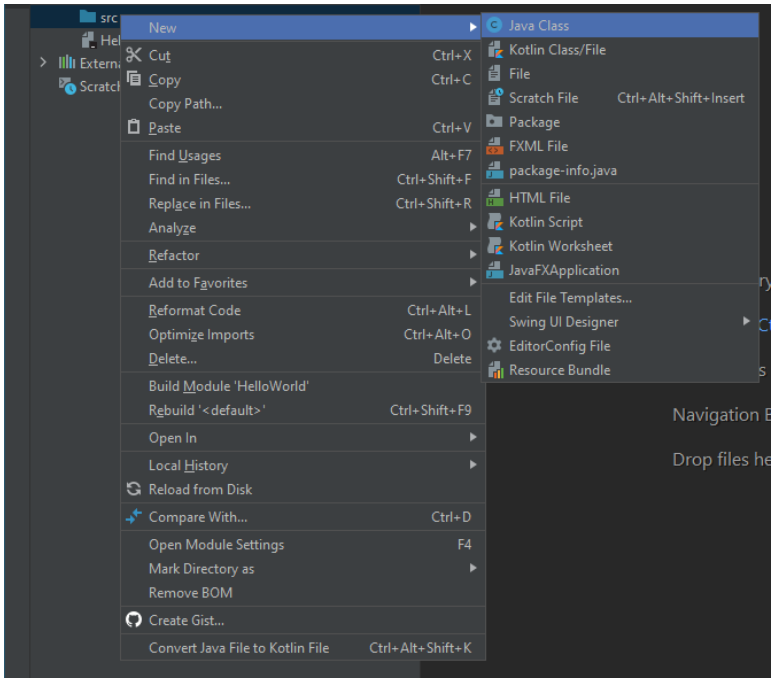



**Figure 30: Creating a new Java class.**

**Important**

A Java class is simply another name for a Java program. It does not refer to a compiled .class file but rather to the source code (.java).

1. Create a new Java class called `HelloWorld`.
2. When this class is created, IntelliJ IDEA will automatically open its corresponding .java file in the editor:

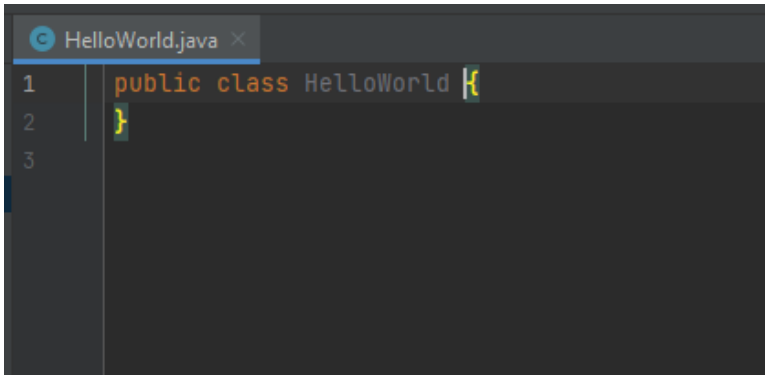



**Figure 31: HelloWorld.java, opened in the IntelliJ IDEA text editor.**

3. The text editor of IntelliJ IDEA behaves just like Notepad does, allowing us to write, view and edit code. However, it also supports a plethora of other helpful features, such as those mentioned previously.
4. Delete all the text in this file and replace it with the following code. *As with previous examples, you do not need to know what the code does for now*.

```
public class HelloWorld {

    public static void main(String[] args) {

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

    }

}
```

5. It is time to run our program within IntelliJ IDEA. Click on the Run menu at the top and then select Run:

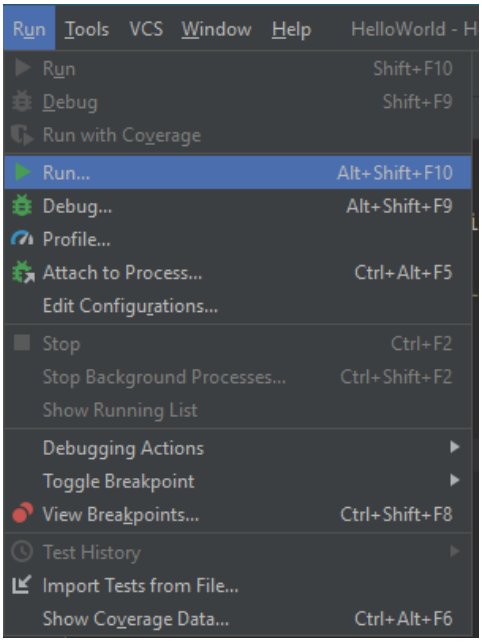



**Figure 32: Selecting run to run a program in IntelliJ IDEA.**

6. Select the name of your program from the list to run it:

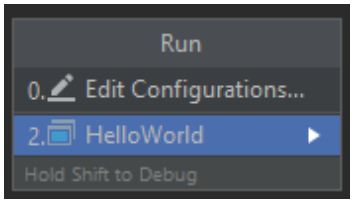



**Figure 33: Running a program.**

7. Give some time to IntelliJ IDEA to automatically compile and run your program. After this process is completed, you should see a console output window appear at the bottom of the project view:

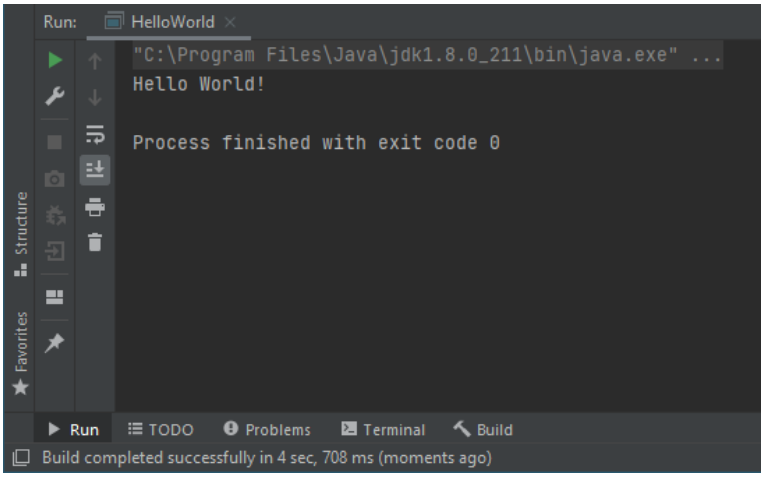



**Figure 34: The console view.**

The console output within IntelliJ IDEA is similar to the Command Prompt output and displays the output of your program. Congratulations, your program printed the words "Hello world!" successfully!

**Hello World**

Printing the text "Hello World" is considered the most basic program possible in any programming language. It is often used as an entry point for beginners to see how a program is compiled and executed, as well as by experienced programmers to test if a certain programming configuration works correctly.

**Automated compilation and execution**

IntelliJ IDEA automatically compiles and runs your program for you and displays the results within the same window. This makes the process of compiling and running a program easier and faster.

**Run shortcut**

Alternatively, you can quickly run a program by pressing **SHIFT + CTRL + F10**.

**Important**

Make sure you have understood the concepts discussed in this worksheet and installed the Java JDK, JRE and IntelliJ IDEA before moving on to the next sessions.

## Congratulations on taking your first steps as a programmer!

## FOR MORE INFORMATION:


+357 99801631

info@estudyquals.com

www.estudyquals.com