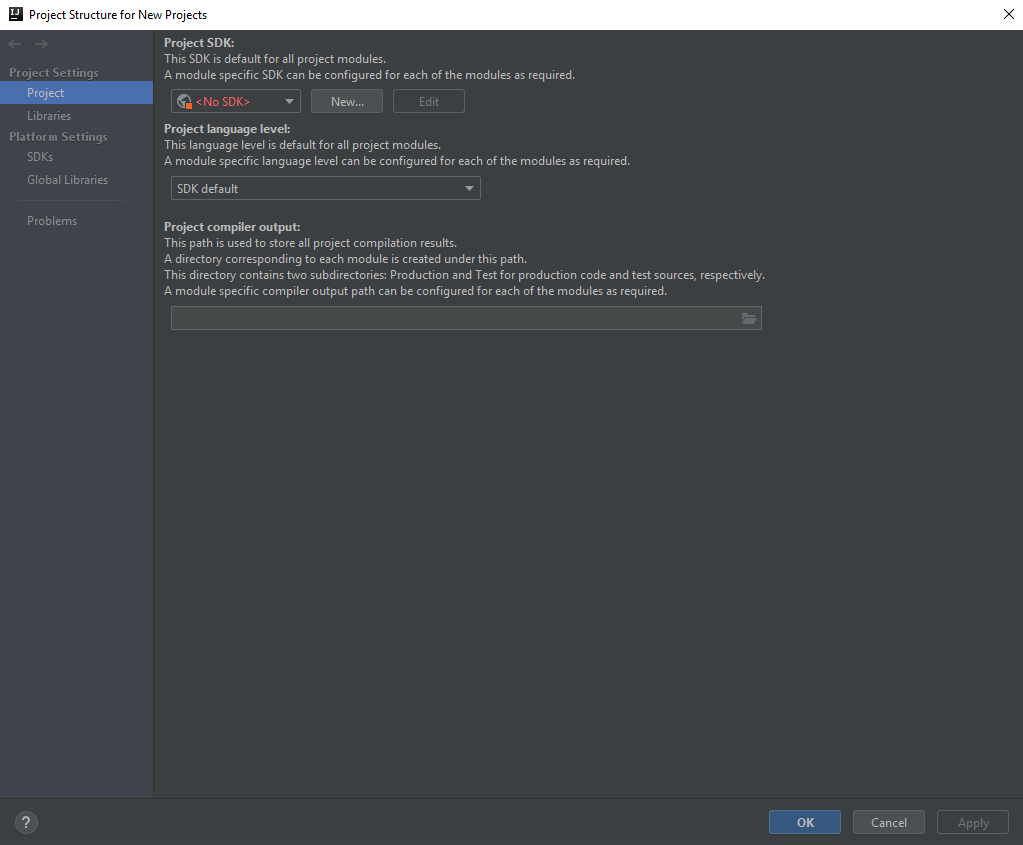
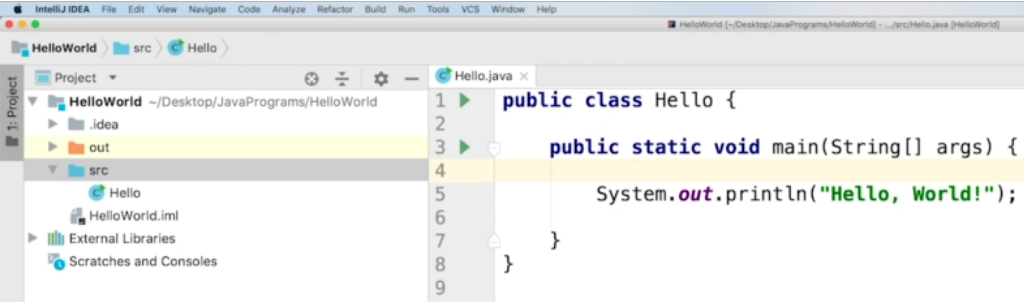
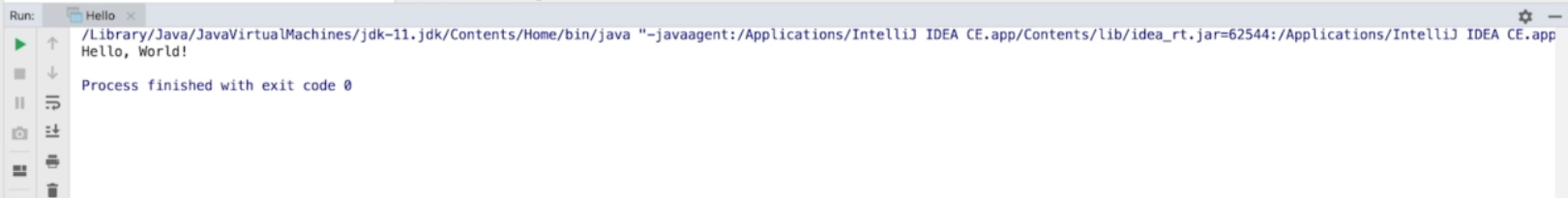
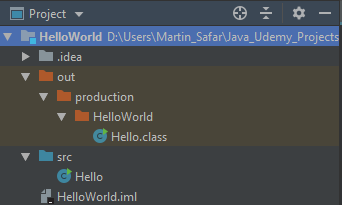
**Which Version of Java?**  
\* My advice is to grab the JDK 11 which is the latest version of the Java Development Kit right now. I’m gonna use the words Java and JDK interchangeably.  
\* The JDK helps you write or compile Java programs.  
\* You run your Java with something called Java Virtual Machine (JVM) with Java Runtime Edition (JRE).  
\* Java itself is a generic term and JDK is the Development Kit that you need to write Java programs.  
\* Which version of Java is now a bit more complicated than it used to be.  
\* The Oracle version of Java is paid only for business, so my advice is to use it. You don’t need to legally pay anything to use Oracle Java as a programmer.  
\* Oracle has moved to LTS model - long-term support. (In the past you would just download the latest version).  
\* Companies though will standardize on Java 11 in the future and that’s a good version for you to work with and that’s the reason I’m gonna be focusing on Java 11.  
\* So Java 11 or in our case JDK 11 is what we want.

**Installing JDK 11 on Windows**  
\* How do you know if your computer is running a 32-bit or a 64-bit version of Windows?  
=> Settings => System => About Your PC to find our if you have 32-bit or 64-bit.  
=> java.sun.com => Java SE (Standard Edition) => Oracle JDK DOWNLOAD => scroll down and Accept License Agreement => choose the Windows .exe  
**Installing and Configuring IntelliJ IDEA on Windows**  
=> Jetbrains.com => Tools => IntelliJ IDEA => Download  
\* It actually has 2 version: the Community Edition is free + open-source.  
\* During the installation process you have an option to import the settings if you’ve already installed it at some point.  
\* IntelliJ allows you to install a variety of different tools - we don’t need any of these tools at this point of the course. Same for the featured plugins.  
\* Now what we need to do is a once-only process - we need to assoaciate the JDK that we’ve previously downloaded and installed with IntelliJ so that IntelliJ is able to access and use that JDK.  
=> Configure => Project Defaults => Project Structure  
\* If you’re installing IntelliJ IDEA for the first time, you’ll probably get something like this:  
  
=> Project => New (SDK - Software Development Kit) => JDK =>   
\* Now in some cases IntelliJ IDEA will automatically find the JDK that we installed, otherwise we need to navigate to the folder where that was installed.  
\* Now you need to match the Project language level to the version you downloaded.  
\* It’s just a few other settings we need to do now which are once-only which just makes life a bit easier for you when you’re starting out with Java with IntelliJ IDEA.  
=> Configure => Settings => Editor => General => Auto import => check the 2 boxes: Add unambiguous imports on the fly, Optimize imports on the fly (for current project) => Code Folding => Uncheck: Imports, One-line methods, “Closures” (anonymous classes implementing one method, before Java 8), Generic constructor and method parameters --- essentially what we’re doing here is we’re setting ouverselves up to make sure that the editor IntelliJ doesn’t hide any code. As you become more experienced as a developer, you’ll want to hide portions of the code to focus on what you’re working on but when we’re starting out, we want to access and see everything so this is the reason why we’ve actually unchecked those options so that all the code will actually be shown.  
\* At this point we’re ready to start programming.

**First Steps - Creating your first Java Program**  
\* In computing terms it’s been a tradition that the very first program you create in a new programming language performs a simple function - it prints out a message that says “Hello, World!”.  
=> Create New Project => Java (top left hand corner) => Project SDK: 11 (java version “11”) => Next   
(you have an option to Create project from template - we’ll be using that a lot in the course later)   
=> Next => Project name: HelloWorld (generally it’s not a good idea to use spaces in project names or path names in general - computers depending on your OS can run into difficulties) => More Settings (it gives you more detail about module names, content root etc., right now you don’t need to know about it, we’ll talk about it later in the course) => Project location => Finish

\* src stands for source and that’s going to be the folder within our project where we’re gonna create our Java programs.  
=> Right Click the “src” => New => Java Class => Name: Hello (note that I typed it with a capital H and that’s a convention for a Java Class name - to start with a capital letter) => Ok  
\* The “c” icon means that it’s a class.  
\* Our IDE automatically indents the code for us.  
  
\* Essentially what we’ve defined here is a common entry point for Java so that we can run some code in it so we’ll be talking more about this public, static and void in later videos.  
\*println is print line for short.  
\* **Right click** somewhere **in the file** and click **Run ‘Hello.main()’**. Usually I suggest you make sure your cursor is in the main function area when you do it.  
\* We can see down in the bottom left hand corner we’ve got this long line of blue output which is the command that IntelliJ IDEA has sent to the JDK or to the Java run-time to run our program and in black there is our message.  
  
\* We also got this output here that was generated not by us but by the program to say process finished with exit code 0. In this case exit code 0 means it wasn’t an error.  
>>> Challenge: how would you change this program that we’ve created to instead of printing “Hello, World!” to print “Hello, yourName!”.

**Exploring IntelliJ IDEA**  
\* A good way of understanding how IntelliJ works is having a look at how it stores the files as we create our programs.  
\* At the top of the screen it gives you the location where IntelliJ IDEA has placed the program.  
=> You can right-click on the project name and click Show in Explorer.  
=> **CTRL + N** => Navigate (search for files, classes, …)  
  
\* **HelloWorld.iml** => that’s a file used to keep track of of the state of this project. So it’s a file used by IntelliJ to keep track of things like the files that are part of this project, the names and so on. So it’s not something you ever need to edit or open or use in any way but you certainly wouldn’t want to delete it either, it’s really critical to IntelliJ IDEA to get your project working.  
\* We also have this `**out**` folder and the `**source**` folder.  
\* Hello.class file - we can force the file to be open with a text editor. It’s got some really strange characters in it as you can see.  
  
=> A **.class** file is the compiled version of our program and that would be the one that you would use to distribute and send your code out to other people.  
\* What do I mean be compiled? It’s that we’ve taken the Java code that we’ve written in Hello.java file and we had it converted into a format that the computer understands and can then run or execute.  
\* So when you saw the output down in IntelliJ - that happened because IntelliJ created a class file and then it executed that Hello.class and followed the instructions.  
\* So the left side of IntelliJ is what’s called the **Project Pane** - we’re at the moment in what’s called Project View and if you click the Project at the left top, you can select a different View but Project View is the default view and it mirrors very closely what we can see in that folder, a very similar folder structure to what we saw earlier in the Explorer.  
\* You don’t need to worry about saving your project. In some programming languages and in some IDEs you may need to actually manually save your files. (such as Eclipse - won’t save your changes automatically). But IntelliJ will do that for you, it’s automatically saving things for you. You can do it yourself too by clicking File -> Save All but generally it’s happening for you behind the scenes.  
\* These panes and windows have `**gear/cog - settings**` icons you can click and manipulate them.   
\* You can open panes/windows by clicking **View** => **Tool Windows**.  
\* **Floating Mode** - particularly useful if you have more than 1 screen. By clicking on the gear icon and choosing Floating Mode. You now have a window that’s independent of other windows. You can click on the Floating Mode again to take it back to where it was prior to you setting that.  
\* Now the other thing you can do is you can save a layout. You’ve got a particular way, you wanted to save the windows to always go back to that particular state click:  
=> **Window** -> **Store Current Layout as Default**  
=> **Window** -> **Restore Default Layout**