

# Entry Point

#jvm

#jdk

#jre

#ide

#intellij

Wszystko ma swój początek, a programowanie zaczyna się od przygotowania środowiska do pracy.

## ⚠ Warning

Jeżeli masz problem z jakimś krokiem lub wyjaśnieniem, napisz do mnie!

## Java

Zacznijmy od wyjaśnienia najważniejszych rzeczy związanych z Javą.  
Zainteresowanych odsyłam do [wikipedii](https://pl.wikipedia.org), po więcej informacji.

## JVM - Java Virtual Machine



Każdy na pewno poznaje co najmniej jeden z tych seriali. Ich cechą wspólną jest to że udają, **imitują** prawdziwe życie. Nie znajdziemy w nich magicznych krain, cybernetycznej przyszłości czy epickich bohaterów ale zwykłe życie.

Rozumiemy też to, że wydarzenia w serialach nie dzieją się na prawdę, aktorzy nie są tacy sami w realnym życiu, a mimo to potrafimy się wciągnąć w losy bohaterów.

Gdyby osoba nie znająca pojęcia TV zobaczyła coś takiego, mogłaby pomyśleć że te wydarzenia są prawdziwe!

Podobną rzecz potrafi komputer, a dokładnie program - maszyna wirtualna, emulator. Imituje działanie innego urządzenia, wewnątrz siebie! Nie jest to prawdziwe urządzenia ale w ten sposób możemy *oszukać* inny program, że został uruchomiony na zupełnie innym urządzeniu.

Jeżeli graliście kiedyś na PC w gry przeznaczone na konsole lub starsze urządzenia to mieliście styczność z **emulatorem!** [DOS box](#) [PCSX2](#)

Jeżeli kiedyś korzystaliście z komputera, wewnątrz innego komputera to mieliście styczność z **maszyną wirtualną!** [VirtualBox](#)

Język java korzysta z podobnego rozwiązania. Programy napisane w Javie nie działają bezpośrednio na naszym systemie, ale działają za pomocą pośrednika. Program java myśli że jest uruchomiony na specjalnej maszynie - JVM.

Aha. Ale w sumie to po co?

W dzisiejszych czasach może nie jest to tak bardzo wyraźne ale napisanie programu, który działa wszędzie, nie zawsze jest takie proste i wymaga od nas specjalnych kroków, modyfikacji w kodzie pod konkretne urządzenie czy tworzenie specjalnych wersji pod różne systemy operacyjne.

Dzięki takiemu rozwiązaniu możemy uruchamiać nasze programy na dowolnym komputerze z dowolnym system operacyjnym (Windows, Linux, MacOS), bez żadnych komplikacji.

Write once, run anywhere.

## JRE - Java Runtime Enviroment

Wiemy już czym jest JVM, ale pozostaje jedno pytanie.

*Jak skorzystać z JVM, jak uruchomić program?*

Do tego celu służy JRE, oprogramowanie która pozwala nam na uruchamianie programów napisanych w Javie. JRE tworzy dla nas instancje (egzemplarz) JVM, dostarcza potrzebne narzędzia i uruchamia dla Nas program na tej maszynie.

Więcej informacji o tym jak działa JRE znajdziesz [tutaj](#).

---

### ⚠ Zapamiętaj!

JRE pozwala nam na uruchamianie programów, ale nie ich tworzenie.

### 📘 Instancja

Instancja to pojedynczy egzemplarz danego wzorca.

Możesz to sobie zaobrazować do posiadania własnego auta, TV czy laptopa. Każdy egzemplarz (instancja) jest tworzony według określonego modelu (wzorca), możemy nabyć instancję i to co z nią zrobimy nie wpływa na inne instancje.

Nie włączymy HBO na TV na wszystkich egzemplarzach danego modelu, a tylko i wyłącznie na swoim :)

### ⚠ Zapamiętaj!

Aby uruchomić program napisany w języku Java, potrzebujemy do tego zainstalować JRE na urządzeniu docelowym!

JRE pobierzemy w następnym punkcie.

## JDK - Java Development Kit

Chcemy się skupić na tworzeniu programów w Javie, a nie ich uruchamianiu. Chociaż to drugie oczywiście też będziemy robić.

Aby programować w Javie, potrzebny jest nam zestaw narzędzi developerskich - czyli JDK! Co on zawiera? A dokładniej, co nas interesuje na ten moment.

Narzędzie	Opis
Podstawowe biblioteki	Na ten moment nie musisz rozumieć co dokładnie to znaczy. Wiedz że dzięki temu będziemy mogli wypisywać komunikaty w oknie konsoli, odczytywać dane od użytkownika, korzystać z funkcji matematycznych, łączyć się z siecią internet i wiele więcej

Narzędzie	Opis
Kompilator	Kompilator służy tłumaczeniu kodu programu, zrozumiałego dla człowieka na tzw. bytecode, kod zrozumiały dla JVM. Także wytknie nam błędy i nawet powie gdzie one są :) Oczywiście nie przetłumaczy kodu na bytecode póki są błędy
JRE	Tą nazwę już znasz. JDK zawiera w sobie JRE więc nie ma potrzeby instalować ich osobno. Logiczne, prawda? W końcu gdy napiszemy nasz program to przecież będziemy chcieli zobaczyć jak działa!

## Instalacja JDK | Opcjonalnie

### Dlaczego opcjonalnie?

W następnym kroku będziemy instalować IDE, czyli program w którym piszemy programy. Umożliwia on automatyczne pobranie i zainstalowanie wybranej wersji Java.

Będziemy korzystać z Javy +11.0.2, a przynajmniej ja będę pisał w niej przykłady i zadania. Jeżeli pobierzesz wyższą wersję, też **powinno** działać.

JDK pobierzesz [tutaj](#)

#### **11.0.2 (build 11.0.2+9)**

<b>Windows</b>	<b>64-bit</b>	zip (sha256) 179 MB
<b>Mac</b>	<b>64-bit</b>	tar.gz (sha256) 174 MB
<b>Linux</b>	<b>64-bit</b>	tar.gz (sha256) 179 MB
	<b>Source</b>	Tags are jdk-11.0.2+9, jdk-11.0.2-ga

Wybierz zip jeżeli chcesz pobrać dla systemu windows.

Wypakuj zawartość w dowolne miejsce, ale tak byś pamiętał jego lokalizację. Narzędzia developerskie polecam trzymać na dysku C (Windows), np. `C:\Program Files\Java\jdk-11.0.2` (domyślna lokalizacja)

Ja trzymam JDK w: `C:\Java\jdk-11.0.2`

 **Warning**

Krok tylko dla użytkowników windowsa, jeżeli korzystasz z innego systemu to skontaktuj się ze mną. Nie przewiduję że ktoś z osób dla których tworzę to szkolenie, będzie korzystał z czegoś innego więc nie chce tracić czasu na opisywanie innych systemów.

Ostatnim krokiem jest dodanie `.../jdk-11.0.2/bin` do zmiennych środowiskowych PATH.

## Terminal

Podczas programowania naturalnym staje się korzystanie z terminala, okna konsoli przez które korzystamy z komend.

Jeżeli chcemy uruchomić program przez okno konsoli to domyślnym zachowaniem jest to że musimy być w tej samej lokalizacji gdzie znajduje się program. Inaczej dostaniemy podobny komunikat błędu.

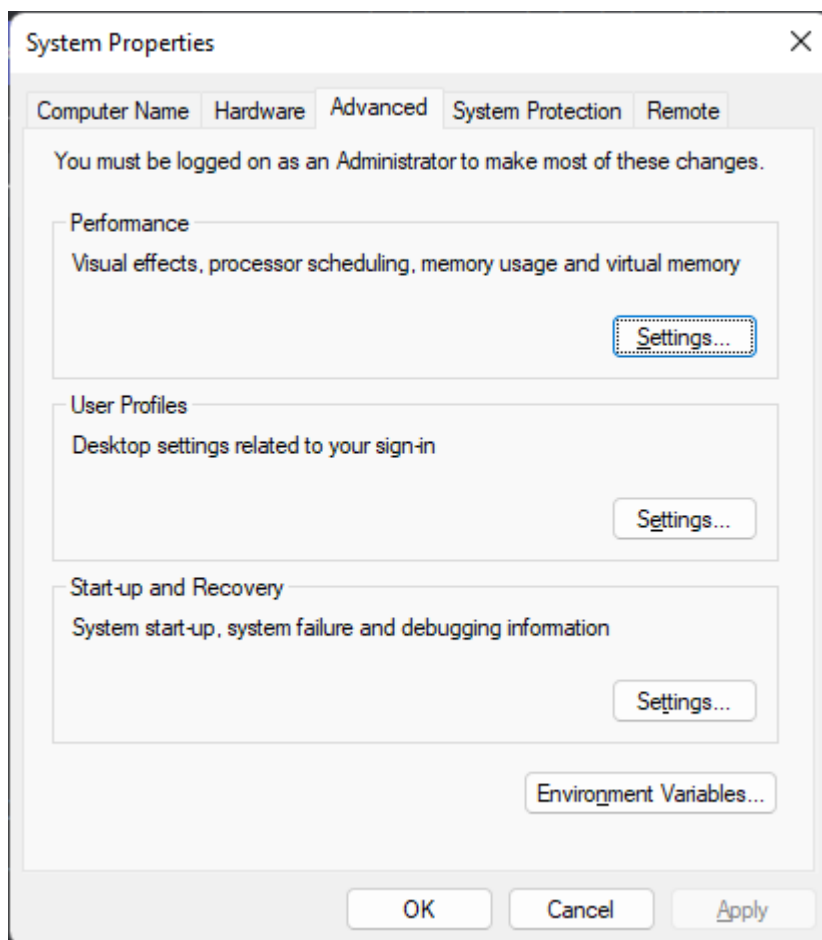
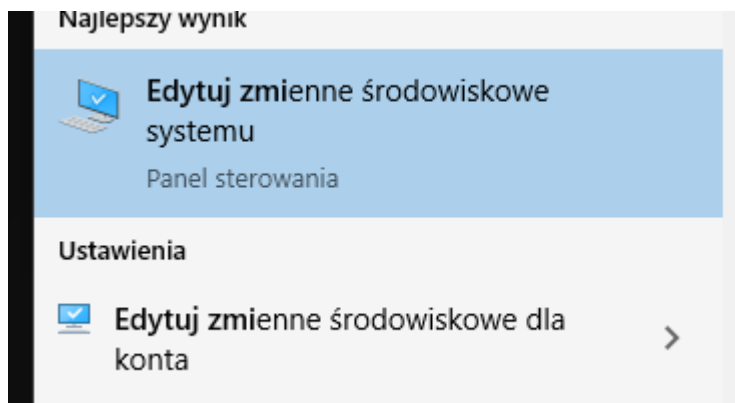
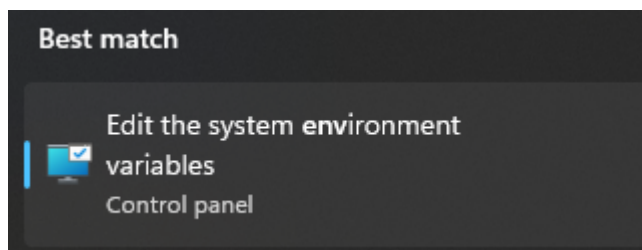
```
PS C:\Users\piotr> java --version
java : The term 'java' is not recognized as the name of a cmdlet, function, script file, or operable program. Check
the spelling of the name, or if a path was included, verify that the path is correct and try again.
At line:1 char:1
+ java --version
+ ~~~~~
+ CategoryInfo          : ObjectNotFound: (java:String) [], CommandNotFoundException
+ FullyQualifiedErrorId : CommandNotFoundException
```

Dodając ścieżkę do zmiennych środowiskowych PATH, sprawiamy że program jest widoczny z dowolnego miejsca na komputerze.

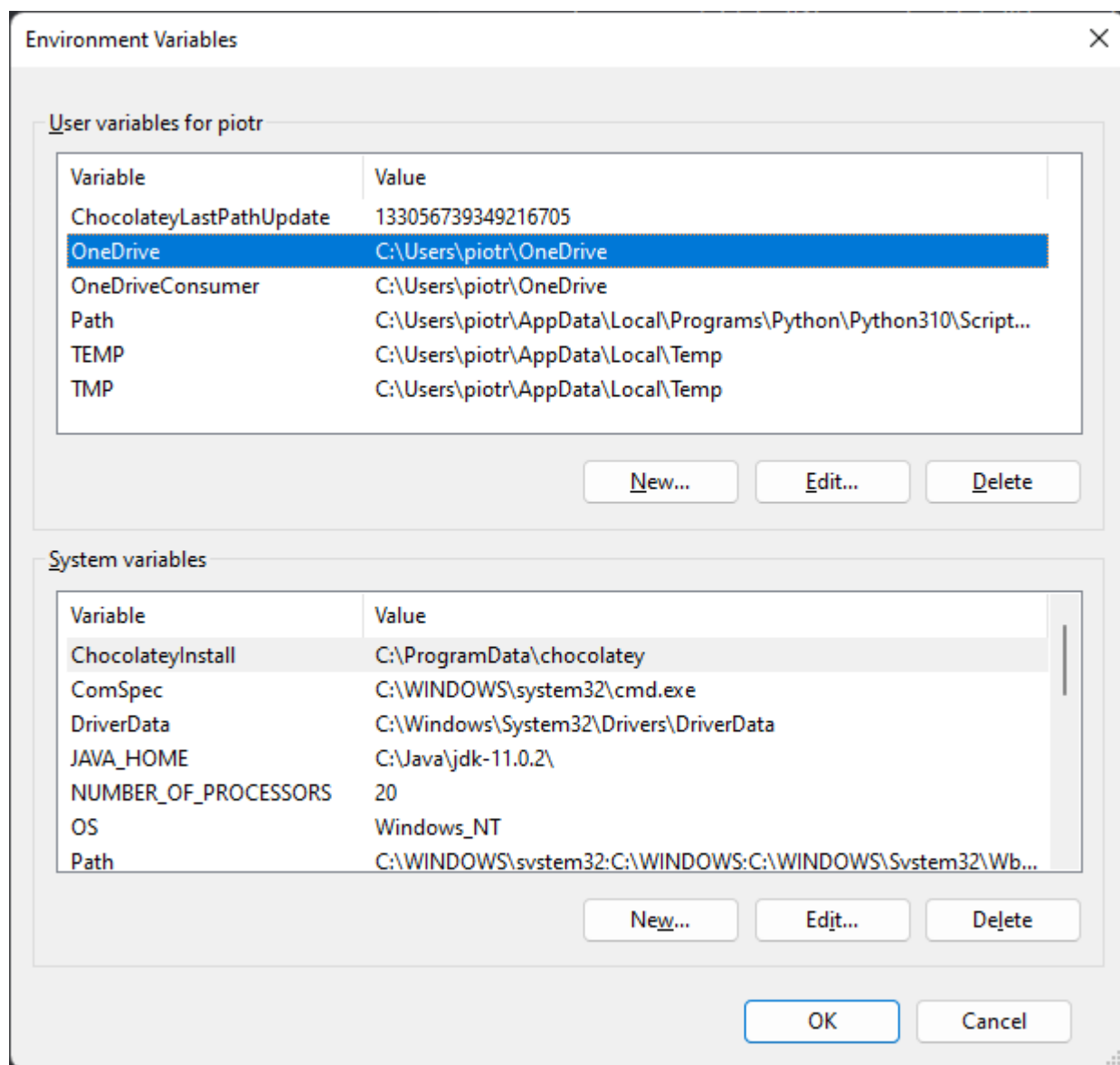
```
PS C:\Users\piotr> java --version
openjdk 11.0.2 2019-01-15
OpenJDK Runtime Environment 18.9 (build 11.0.2+9)
OpenJDK 64-Bit Server VM 18.9 (build 11.0.2+9, mixed mode)
```

Zwróć uwagę że oba wywołania znajdują się w lokalizacji `C:\Users\piotr`

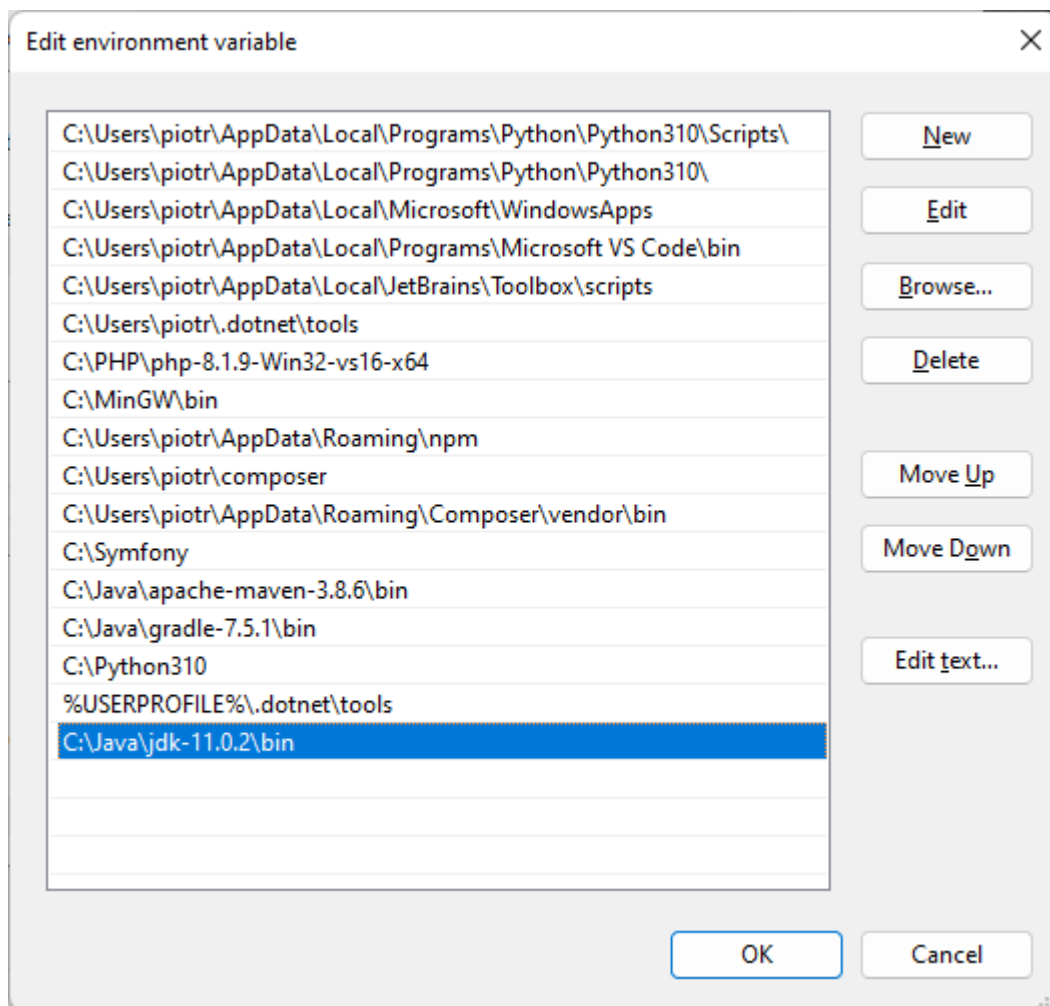
Aby dodać ścieżkę do zmiennych środowiskowych PATH, wyszukaj w programach "Edit the system environment variables" / "Edytuj zmienne środowiskowe systemu"



Następnie > "Environment Variables..."

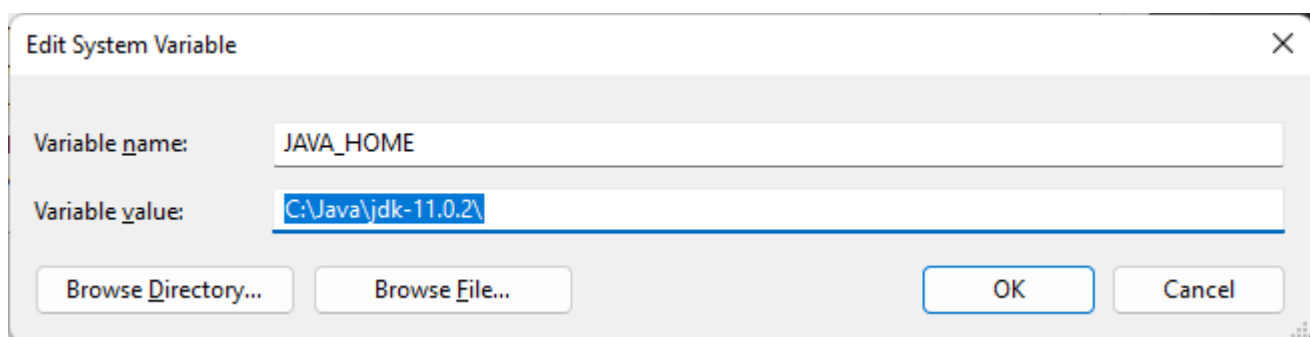


Z górnej tabeli (jeżeli chcesz ustawić tylko dla jednego użytkownika) lub z dolnej (jeżeli chcesz ustawić dla wszystkich) wybierz "Path", a następnie z opcji pod tabelą "Edit..."



Teraz "New" i wklej/wybierz ścieżkę do `.../jdk-11.0.2/bin`

Dobrze jest jeszcze dodać **JAVA\_HOME** dlatego naciśnij "OK" i w poprzednim oknie zamiast "Edit...", wybierz "New..." i uzupełnij pola w analogiczny sposób.



Zauważ że tutaj ścieżka jest bez `\bin` na końcu.

To wszystko!

Teraz możesz otworzyć okno konsoli i wpisać

```
java --version
```

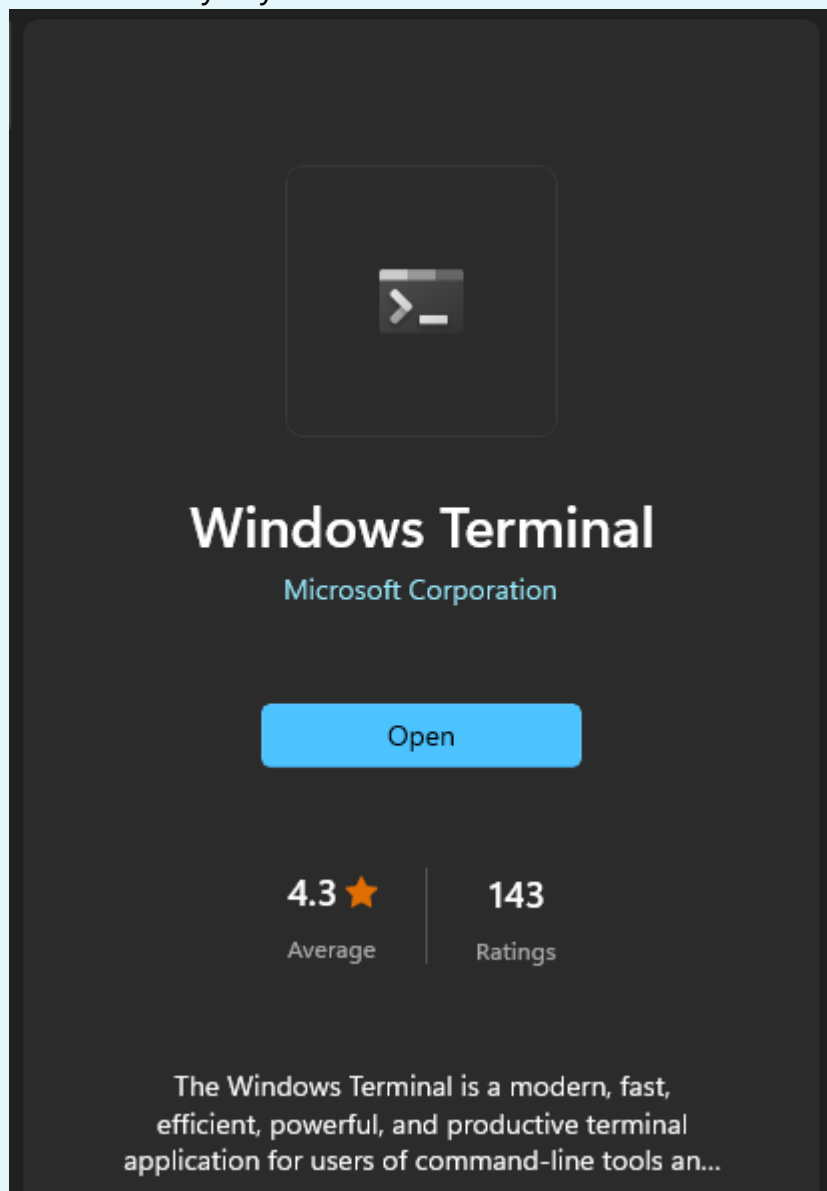


Powinieneś otrzymać coś podobnego do poniższego

```
PS C:\Users\piotr> java --version
openjdk 11.0.2 2019-01-15
OpenJDK Runtime Environment 18.9 (build 11.0.2+9)
OpenJDK 64-Bit Server VM 18.9 (build 11.0.2+9, mixed mode)
```

## 🕒 Windows Terminal

Już teraz polecam Tobie pobrać Windows Terminal z Microsoft Store i używać go zamiast domyślnych okien konsoli.



Windows Terminal posiada obsługę wielu okien, wielu rodzajów konsoli (CMD, Powershell, Bash), a nawet można go modyfikować wizualnie. Przydatną opcją jest

dotadnie przezroczystości, aby widzieć co jest ZA oknem konsoli ;)

```
PS C:\Users\piotr> java --version
openjdk 11.0.2 2019-01-15
OpenJDK Runtime Environment 18.9 (build 11.0.2+9)
OpenJDK 64-Bit Server VM 18.9 (build 11.0.2+9, mixed mode)
PS C:\Users\piotr> |
```

## IntelliJ IDEA

### ❓ Ale gdzie będziemy pisać?

Aby napisać program, musimy... mieć gdzie go napisać.

Może na kartce?! .. yy, no nie. To tak nie działa.

Notatnik ? Jest możliwe.

### ✓ IDE

Ale wiecie co? Istnieją lepsze narzędzia do tego celu i nawet mają swoją nazwę - IDE (Integrated development environment).

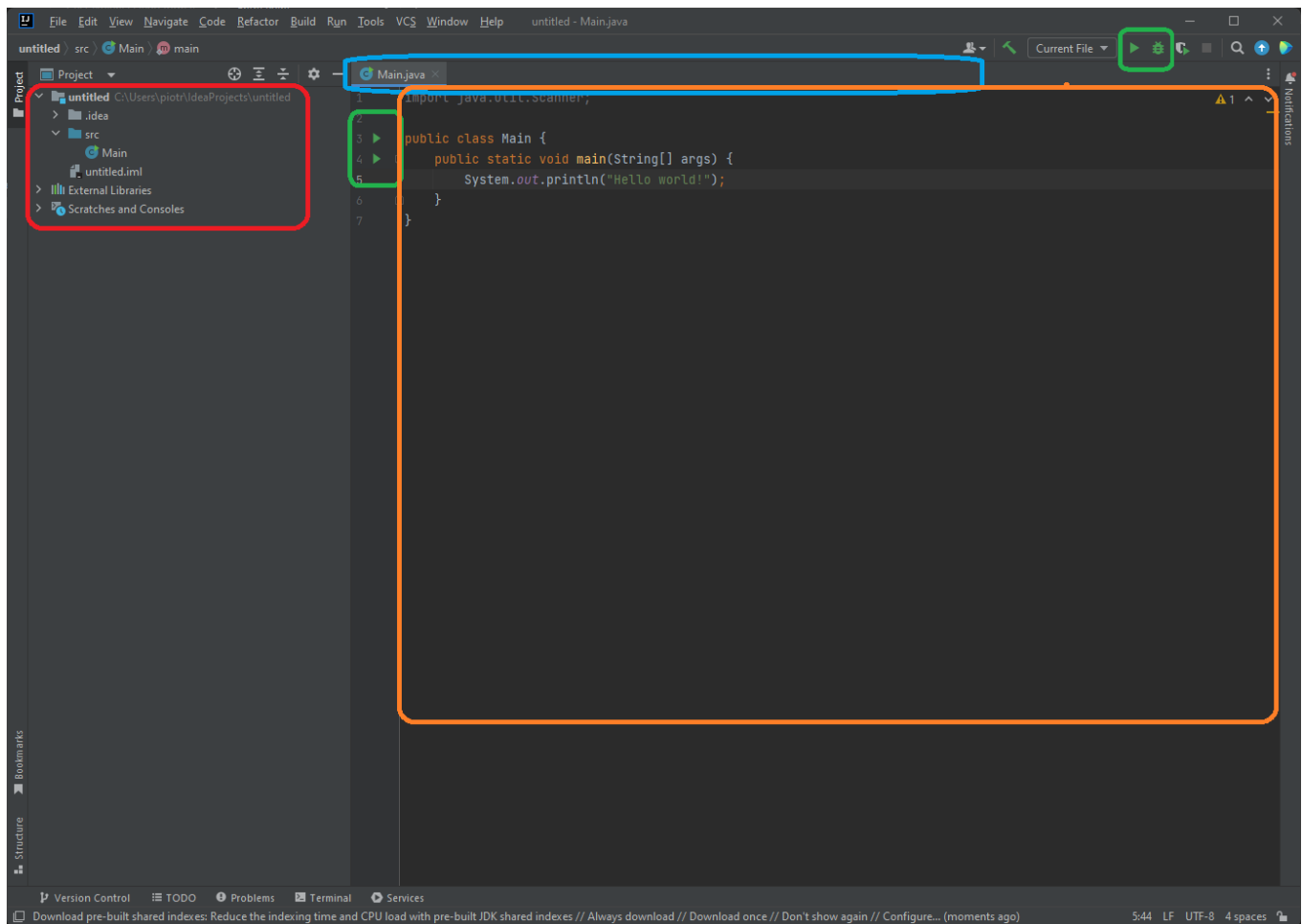
### ❓ Co to za straszna rzecz?

Po co nam takie cudo skoro możemy użyć notatnika i nie robić sobie dodatkowych problemów ? Przecież już zainstalowaliśmy JDK..

### ✓ To przyjacieli!

No właśnie po to żeby nie robić sobie jeszcze więcej problemów.

Pozwólcie że wyjaśnię.



IDE składa się z kilku (najważniejszych) funkcji.

**Katalog plików** - pisząc programy, tak na prawdę pracujemy z setkami plików, a nie tylko jednym. Dobrze jest mieć to poukładane i zawsze pod ręką.

**Mutli-files** - Często pracujemy na wielu plikach na raz, dobrze jest mieć wsparcie do tego, zamiast otwierać ciągle jeden i ten sam program w setkach okienek.

**Kompilowanie i uruchamianie** - to nie do końca prosty proces, zwłaszcza gdy mamy mnóstwo plików i zewnętrznych bibliotek, a testować działanie naszego programu będziemy na prawdę często.

**Edytor tekstu** - IDE posiada specjalny edytor tekstu z wsparciem dla danego języka. Podpowie nam nazwy funkcji, parametrów, pokoloruje nam składanie tak że nawet bez czytania będziemy w stanie odgadnąć przeznaczenie danej linii kodu.

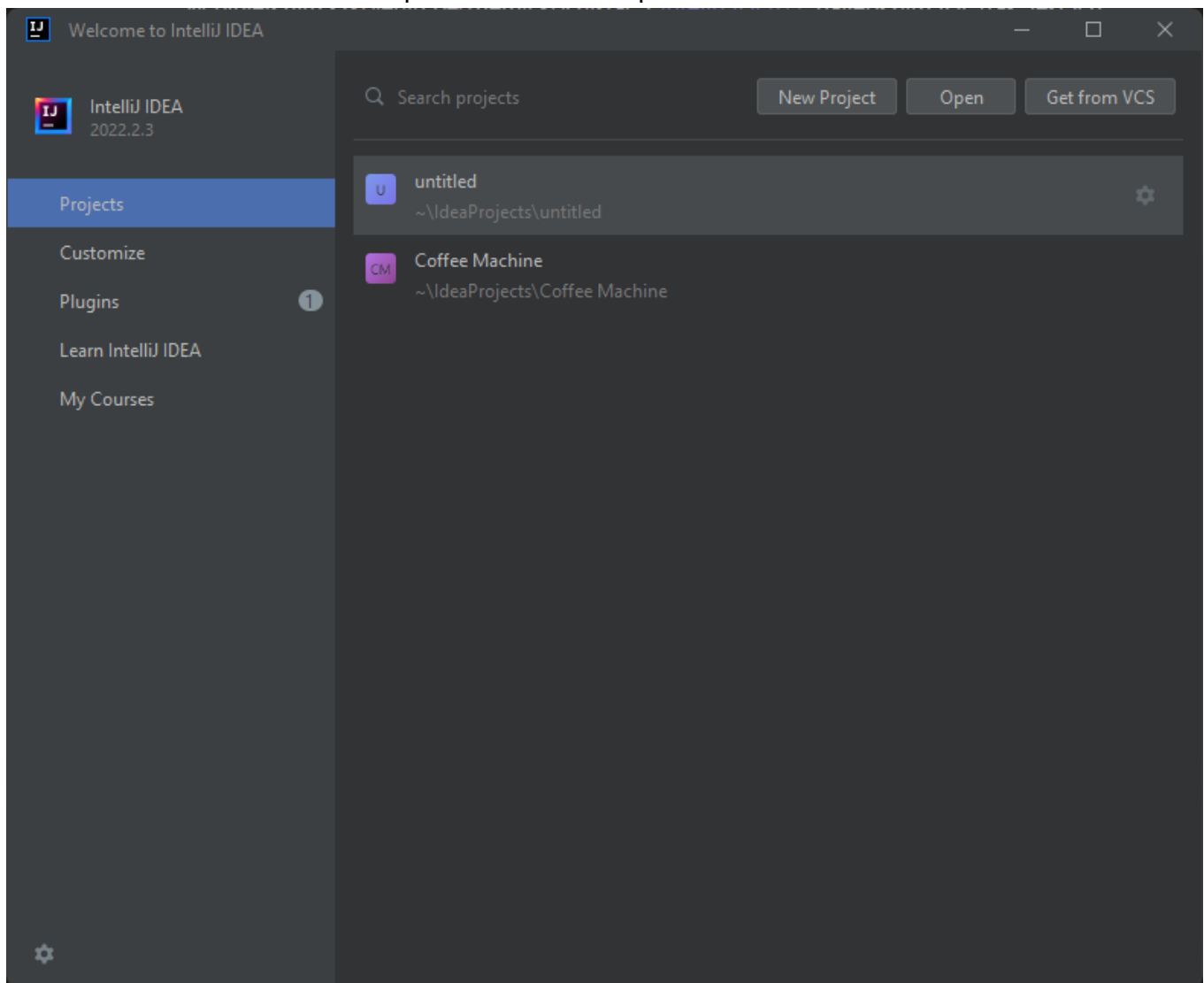
W niniejszym szkoleniu będziemy korzystać z [IntelliJ IDEA](#), najlepszym IDE (tak, jest ich więcej) do Javy! Instalacja jest prosta.. kilkasza next, next, instaluj i tyle.

Z ważniejszych rzeczy polecam wybrać *theme Dracula* żeby Wam oczu nie wypaliło od programowania.



[source](#)

Po uruchomieniu IntelliJ IDEA powinniście mieć podobne okno.



## Hello World!

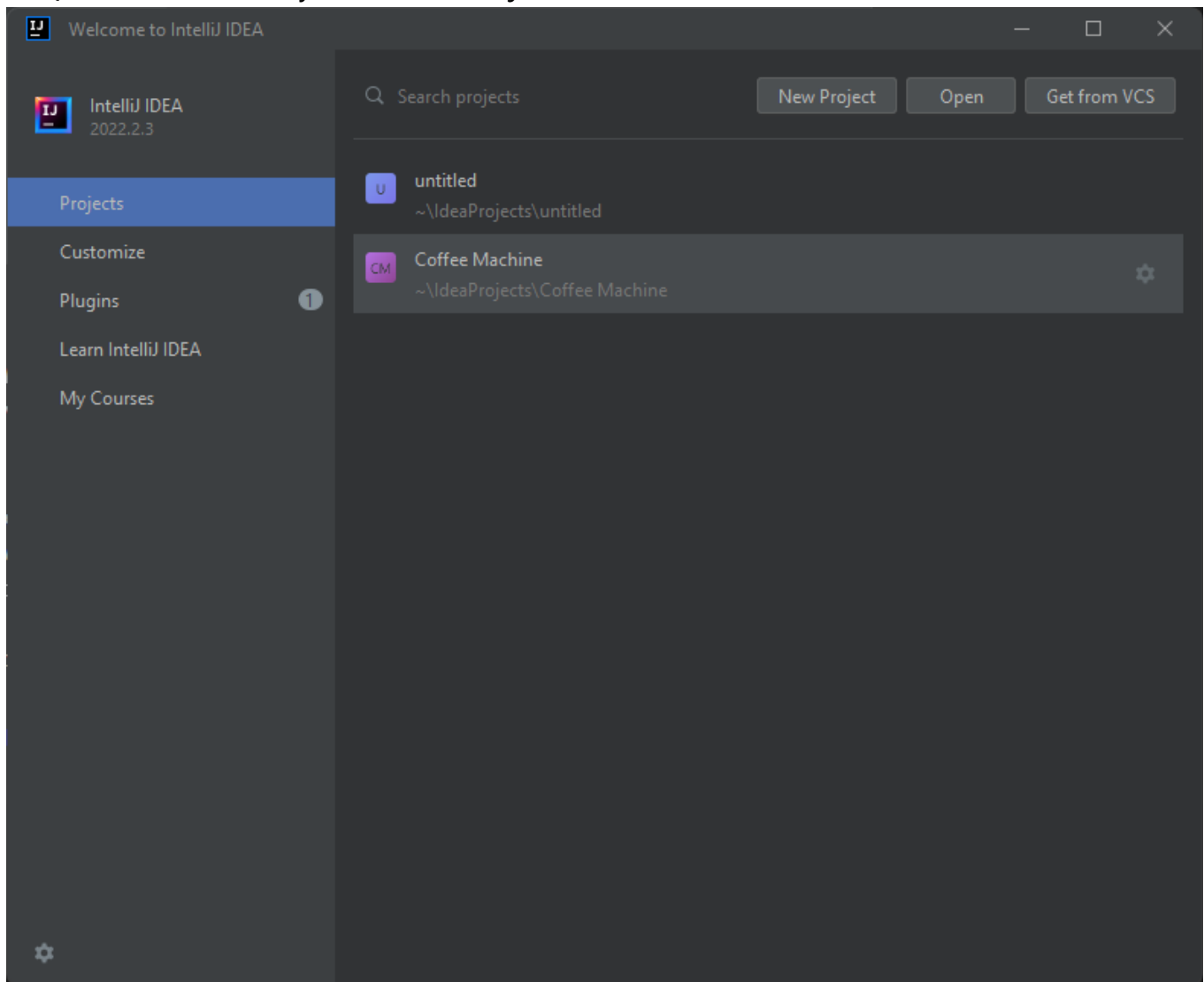
Jeżeli już wszystko zainstalowaliśmy i jesteśmy gotowi do pracy to czas napisać nasz pierwszy program!

Przyjęło się że jeżeli zaczynamy naukę nowego języka programowania, frameworku itp. to nasz pierwszy program powinien wyświetlić nam **Hello World**.  
Ma to na celu sprawdzenie czy zainstalowaliśmy i skonfigurowaliśmy wszystko poprawnie.

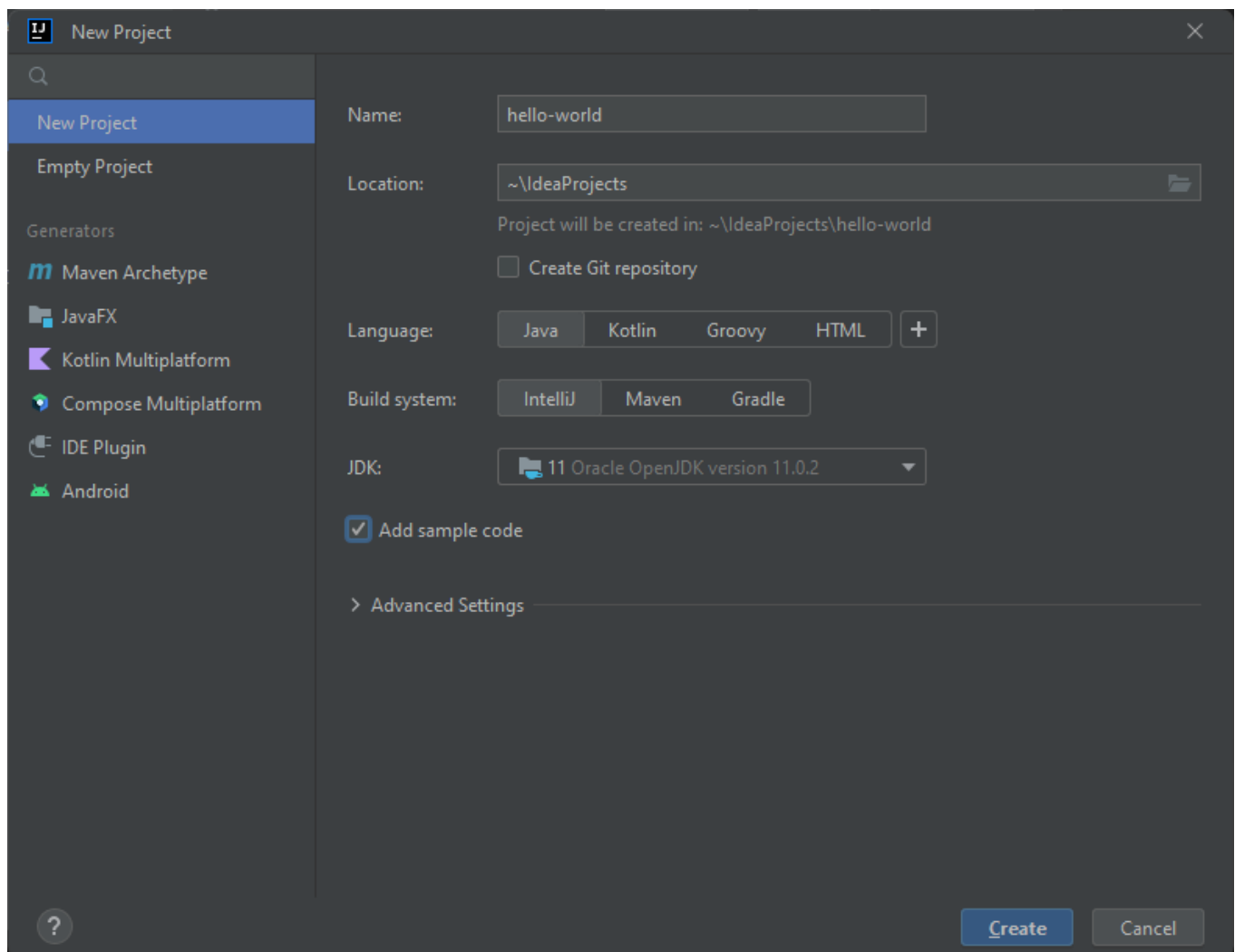
Także zaczynamy.

## Stwórz nowy project

Załącz IntelliJ IDEA i wybierz "New Project"

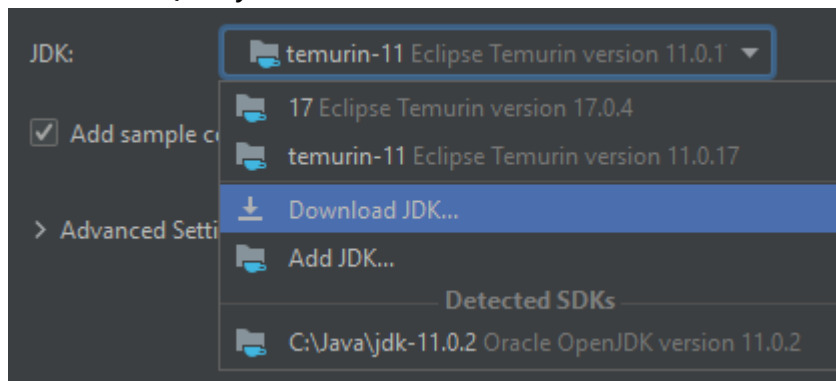


Twoim oczom powinien pokazać się taki widok. Wybierz "New Project" (jeżeli nie był domyślnie wybrany) i uzupełnij pola

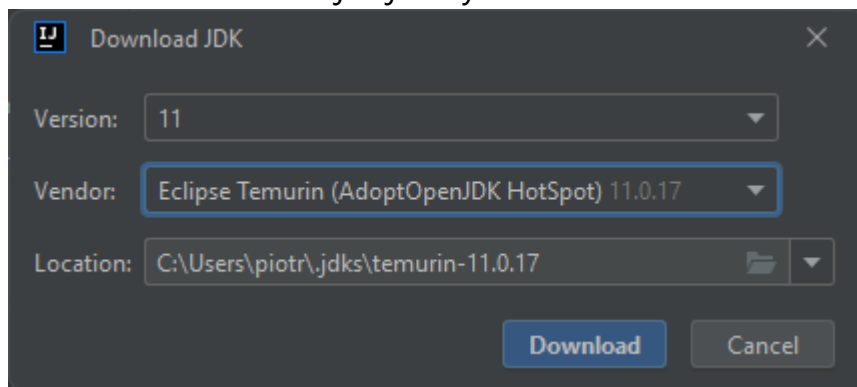


- Name - nazwa twojego projektu, zmień ją na 'hello-world'
- Location - lokalizacja twojego projektu, możesz ją zmienić lub pozostawić domyślną. Zwróć uwagę na komunikat pod polem. IntelliJ IDEA stworzy nowy folder w podanej lokalizacji o nazwie Twojego projektu.
- Language - Wybierz/pozostaw Java, nie będziemy się zajmować innymi językami na tym szkoleniu.
- Create GIT repository - Tym narzędziem będziemy się zajmować się w dalszej części tego szkolenia. Pomiń, nie zaznaczaj.
- Build system - Na ten moment po prostu pozostaw/wybierz IntelliJ. Ten temat będziemy omawiać w dalszej części tego szkolenia.
- JDK - tutaj wybieramy nasze JDK z którego będziemy korzystać. Jeżeli rozwiniesz to zauważysz że możesz w tym miejscu zainstalować JDK. Więc jeżeli tego nie zrobiłeś jeszcze to możesz zrobić to za pomocą IntelliJ IDEA.

Rozwiń listę i wybierz "Download JDK..."



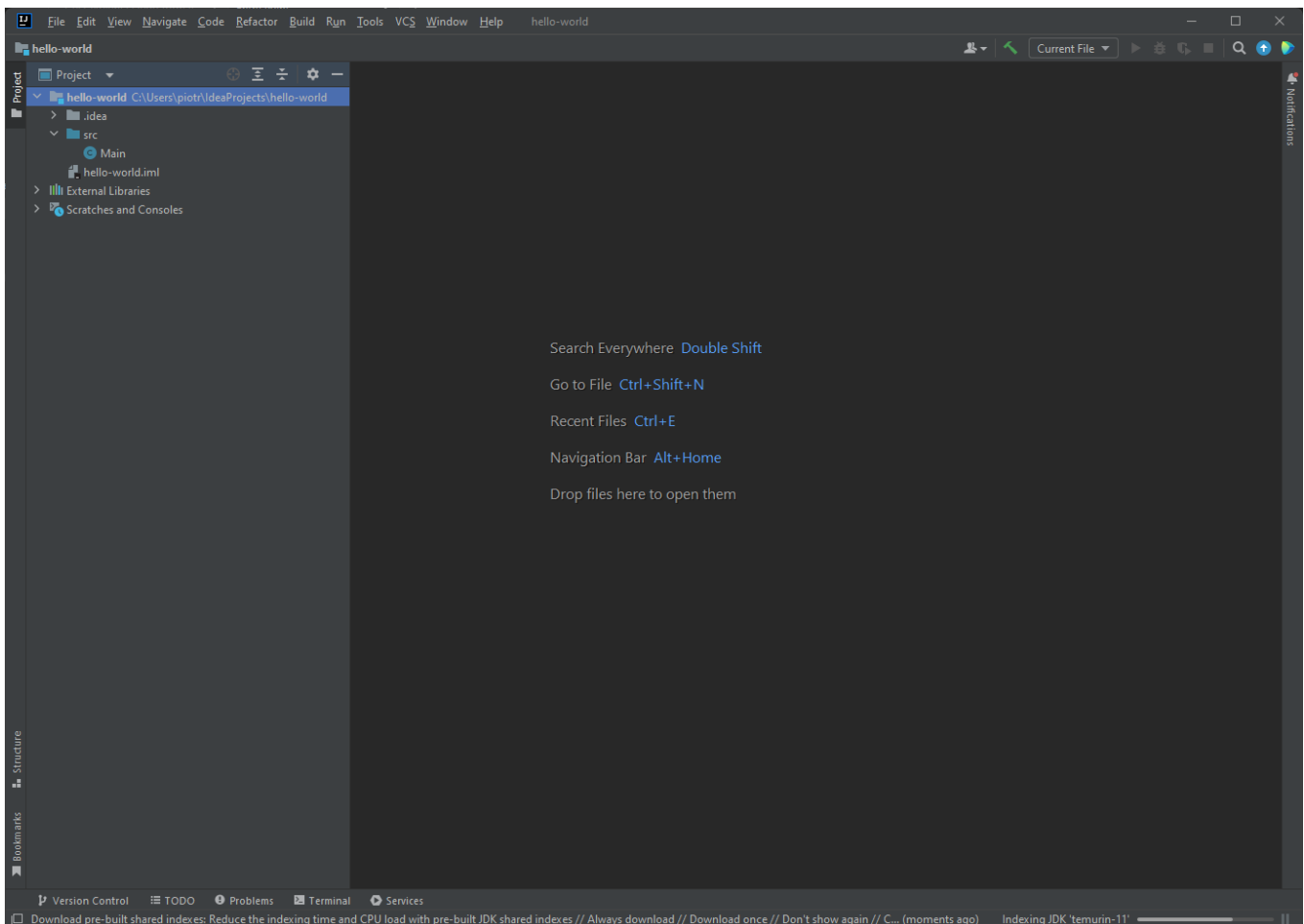
Z opcji wybierz wersję co najmniej 11. Jeżeli nie jesteś pewien to możesz uzupełnić Version & Vendor tak jak ja. Wybierz "Download".



Po zainstalowaniu nowego JDK, powinna zostać automatycznie wybrana, jeżeli tak się nie stało to rozwiń ponownie listę i wybierz JDK z listy( np. *temurin-11 Eclipse...*).

- Advanced Settings - Pomiń.
- Add sample code - Zaznacz.

Jeżeli uzupełniłeś wszystko to wybierz "Create" w prawym dolnym rogu.



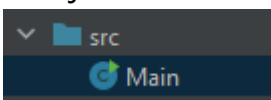
Twoim oczą powinien ukazać się taki widok.

### ⚠ Warning

Zwróć uwagę że IntelliJ po stworzeniu projektu jeszcze coś dla nas robi(o tym co dokładnie kiedy indziej), możesz to zobaczyć w prawym dolnym rogu "Indexing....". Najlepiej poczekać aż skończy zanim przystąpimy do pracy - puste pole w tym miejscu.

## Uruchom program

Otwórz plik **Main** klikając w niego dwa razy LPM (Lewy przycisk myszki). Znajdziesz go po lewej stronie w katalogu plików.




Plik powinien zawierać podobny kod.



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

Teraz naciśnij zielony trójkąt (patrz wcześniej [Entry Point > IntelliJ IDEA](#)) i wybierz **Run**.

A blue button with a green play icon on the left and the text "Run 'Main.main()' Ctrl+Shift+F10" on the right.

Po chwili powinieneś zobaczyć w wbudowanym oknie konsoli w IntelliJ coś podobnego.



Brawo ! Właśnie uruchomiłeś swój pierwszy program!

## A co jeżeli nie dodałem przykładowego kodu podczas tworzenia projektu?

W takim przypadku zapewne folder **src** masz pusty



Naciśnij na niego PPM (Prawy przycisk myszki) i wybierz "New" > "Java Class".

Wpisz "Main" i naciśnij Enter.

A następnie wklej następujący tekst.

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

# Podsumowanie

W tej lekcji nauczyłeś/aś się jak skonfigurować środowisko do pracy z Javą i uruchomiłeś/aś swój pierwszy program!