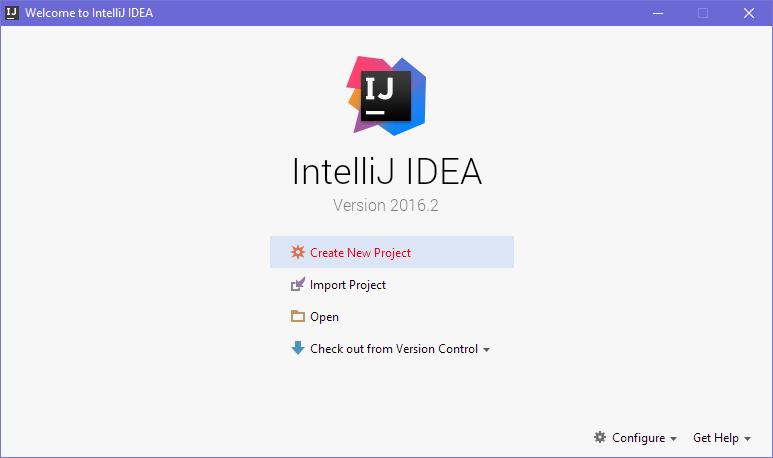
# Упражнения: Първи стъпки в коденето

Задачи за упражнение в клас и за домашно към курса [„Основи на програмирането“ @ СофтУни](https://softuni.bg/courses/programming-basics).

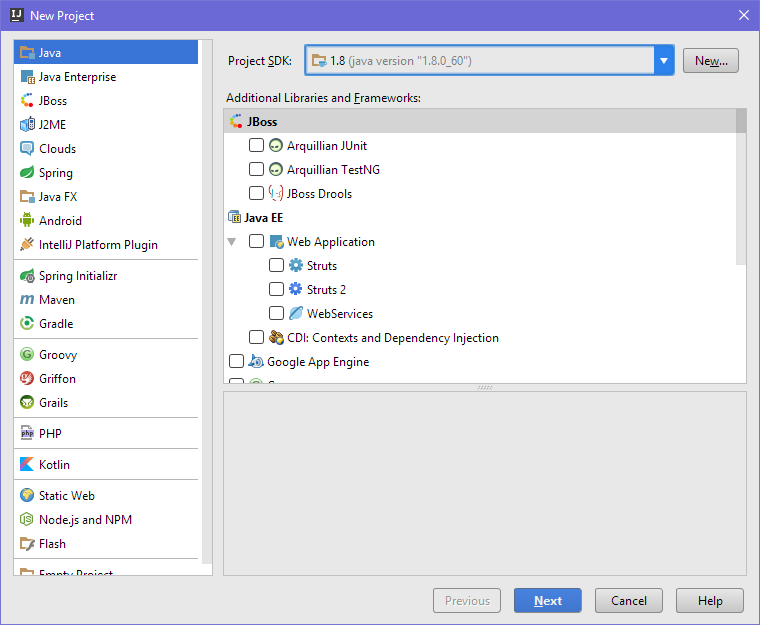
## Конзолна програмка “Hello SoftUni”

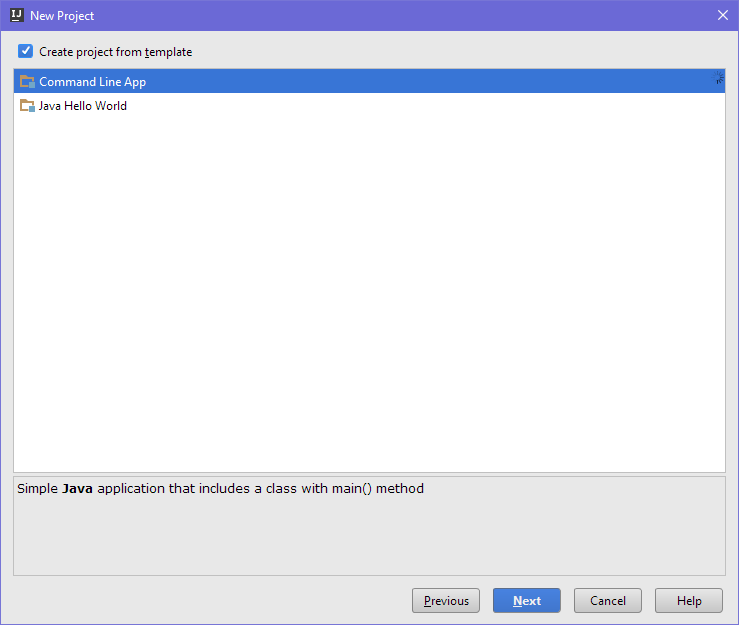
Напишете **конзолна Java програма**, която отпечатва текста “**Hello SoftUni**”.

1. Стартирайте IntelliJ IDEA.
2. Създайте нов конзолен проект: [Create New Project].

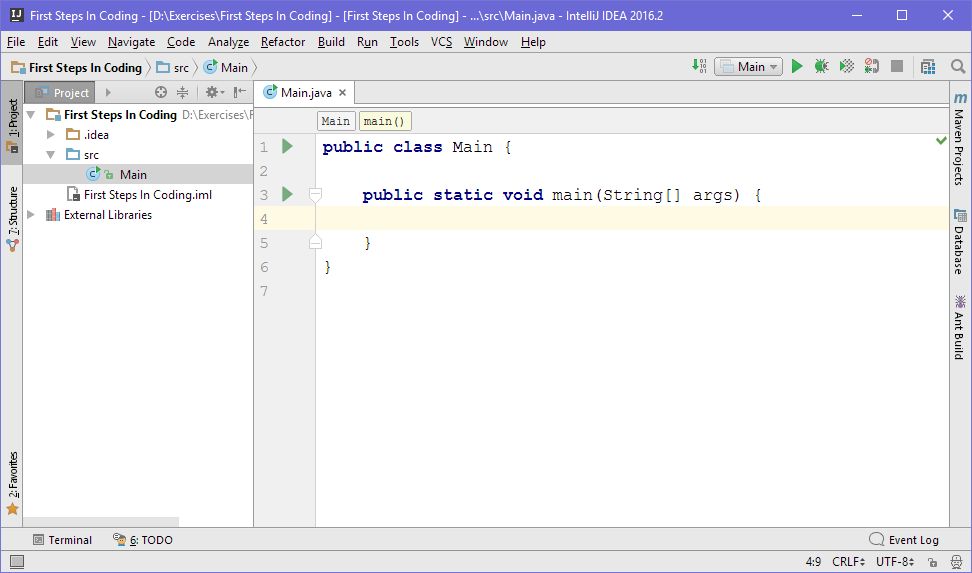


1. Изберете от диалоговия прозорец [Java] 🡪 [Windows] 🡪 [Console Application] и дайте подходящо име на проекта, например “HelloSoftuni”:





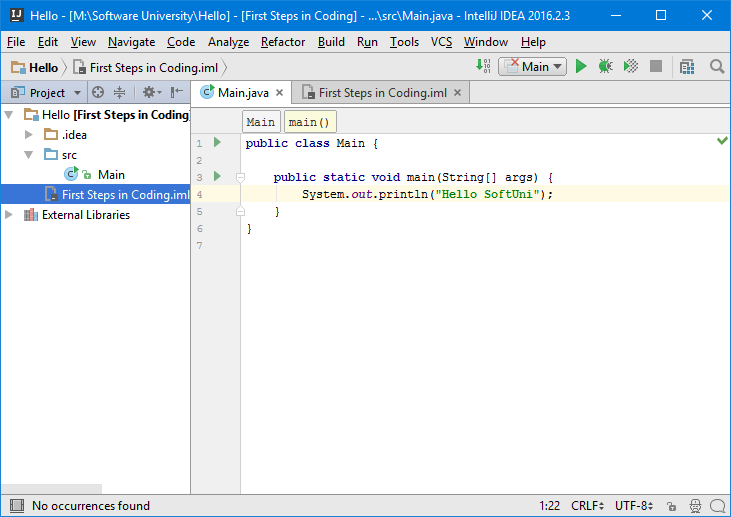
1. Намерете секцията main(String[] args). В нея се пише програмен код (команди) на езика Java.
2. Придвижете курсора между отварящата и затварящата скоба { }.
3. Натиснете **[Enter]** след отварящата скоба **{**.



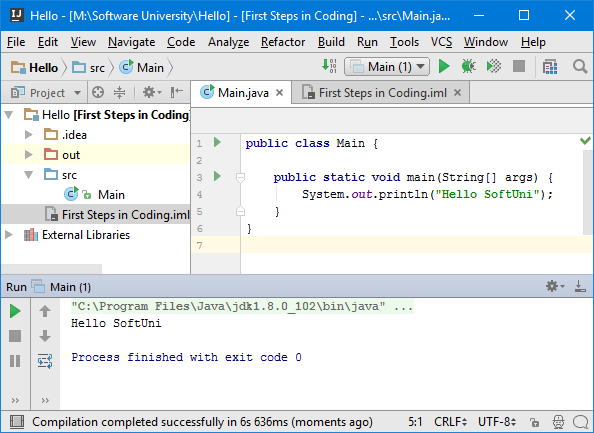
1. Напишете следния програмен код (команда за печатане на текста **"Hello SoftUni"**):

|  |
| --- |
| System.out.println("Hello SoftUni"); |

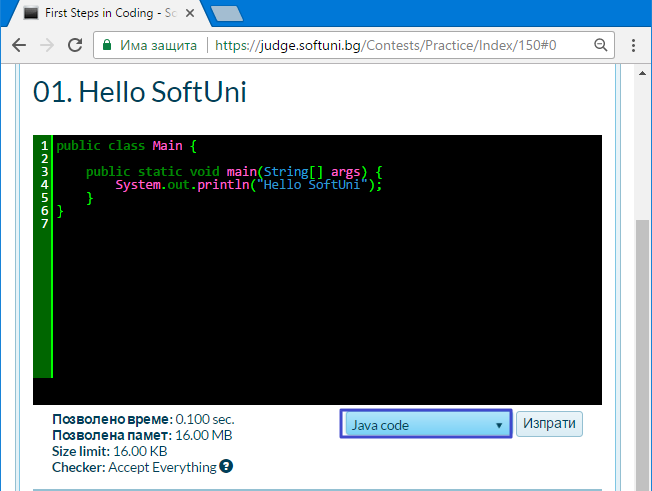
Кодът на програмата се пише отместен навътре с една табулация спрямо отварящата скоба {.



1. **Стартирайте** програмата с натискане на **[Ctrl+Shift+F10]**. Трябва да получите следния резултат:



1. **Тествайте** решението на тази задача в онлайн judge системата на СофтУни. За целта първо отворете <https://judge.softuni.bg/Contests/Practice/Index/150#0>. Влезте с вашето потребителско име в СофтУни. Ще се появи прозорец за изпращане на решения за задача “**Hello SoftUni**”. Копирайте сорс кода от IntelliJ и го поставете в полето за изпращане на решения:



1. **Изпратете решението** за оценяване с бутона [Submit]. Ще получите резултата след няколко секунди в таблицата с изпратени решения в judge системата:



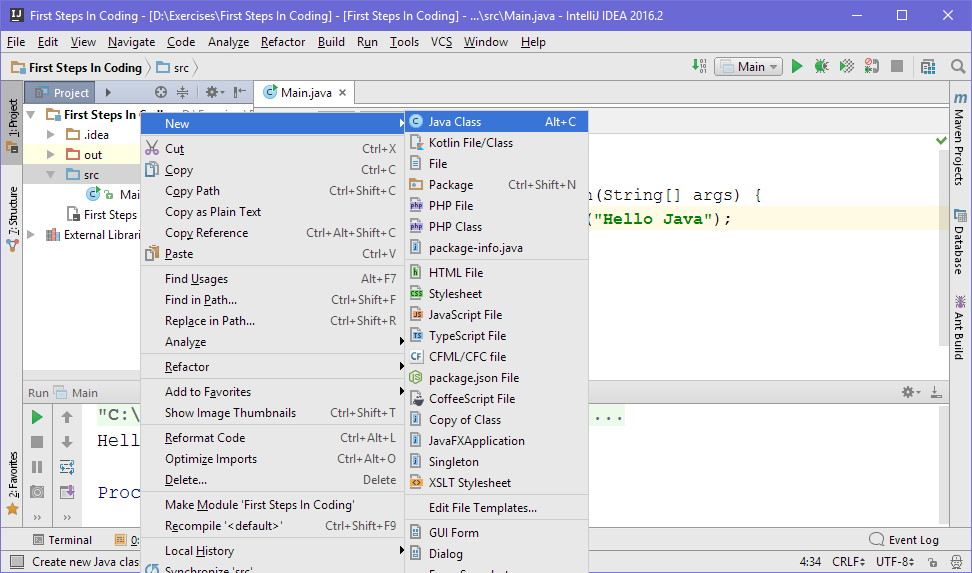
## Конзолна програма “Expression”

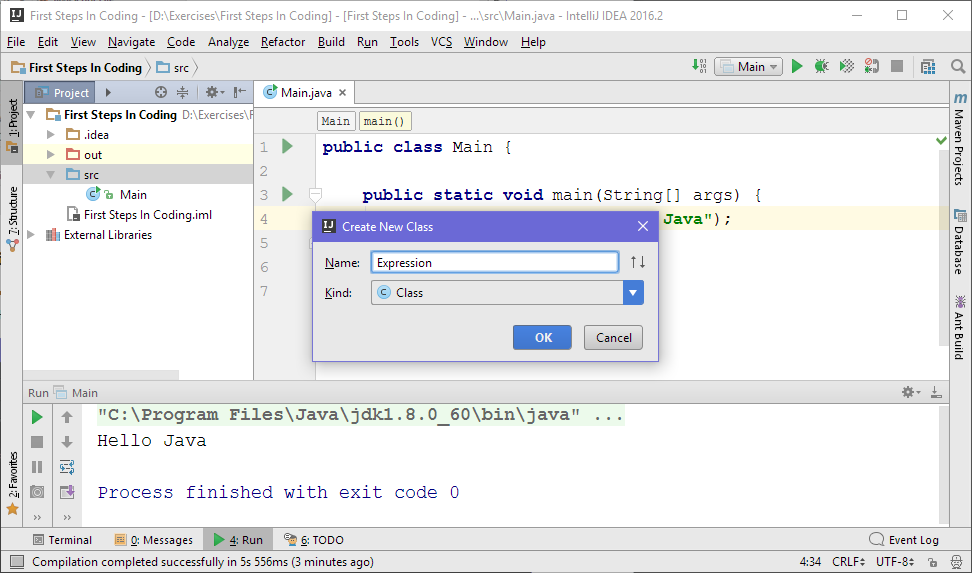
Напишете **конзолна Java програма**, която пресмята и отпечатва стойността на следния **числен израз**:

|  |
| --- |
| (3522 + 52353) \* 23 - (2336 \* 501 + 23432 - 6743) \* 3 |

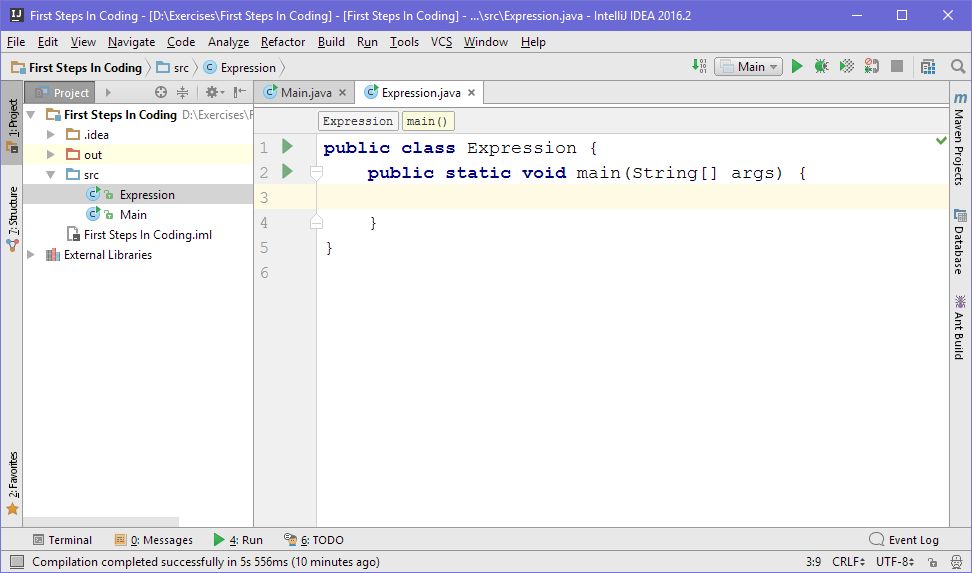
Забележка: не е разрешено да се пресметне стойността предварително (например с Windows Calculator).

1. Направете нов Java клас с име “Expression” (**десен бутон** върху папката „**src**”):



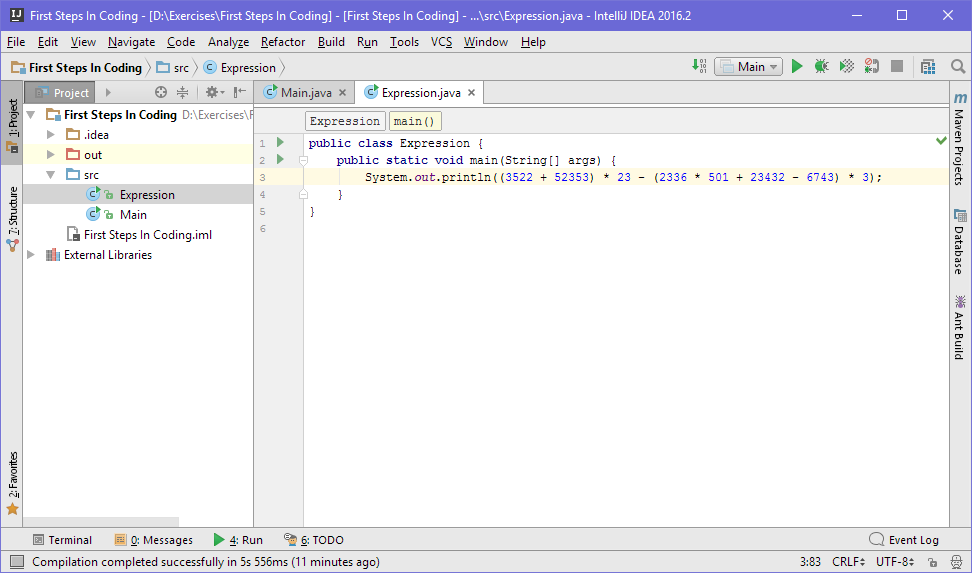


1. Направете си **main метод** в класа, за да има от къде да тръгне вашата програма и влезте в неговото тяло между { и }:

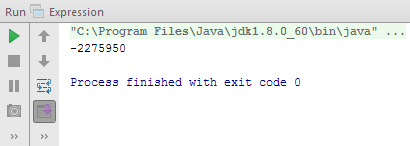


|  |
| --- |
| static void main(String[] args) {  } |

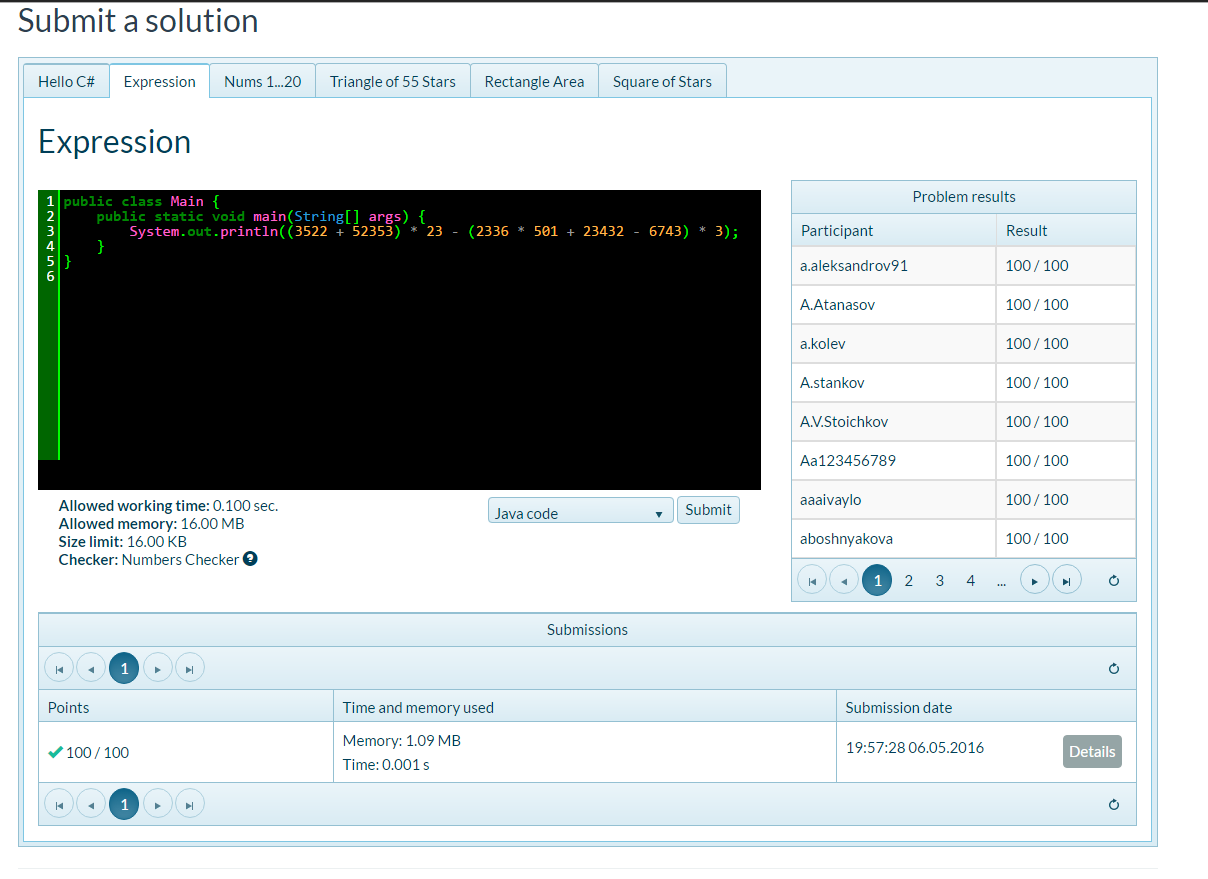
1. Сега трябва да напишете кода, който да изчисли горния числен израз и да отпечата на конзолата стойността му. Подайте горния числен израз в скобите на командата System.out.println():



1. Стартирайте програмата с [Ctrl+Shift+F10] и проверете дали вашият резултат прилича на нашия:



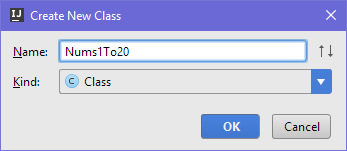
1. Тествайте вашата програма в judge системата: <https://judge.softuni.bg/Contests/Practice/Index/150#1>.



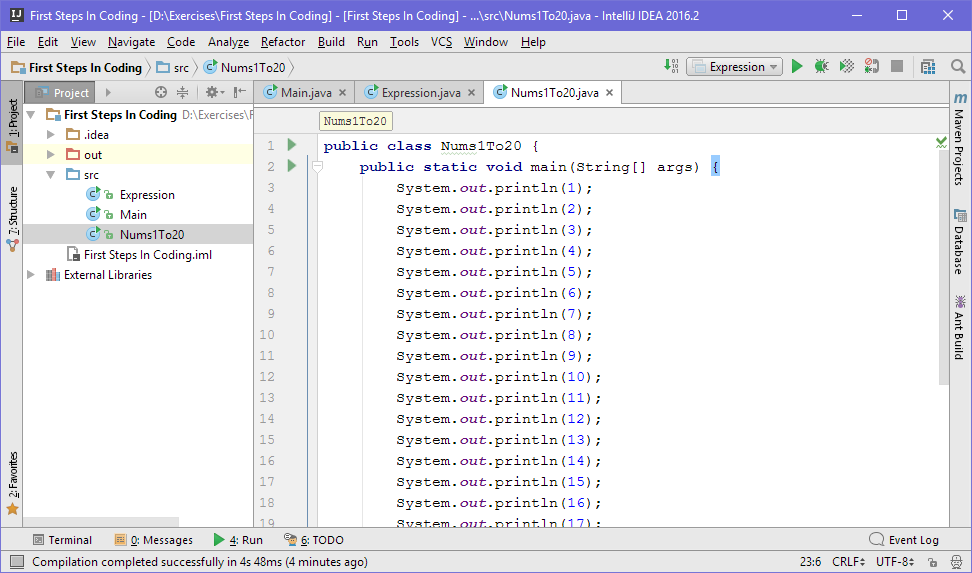
## Числата от 1 до 20

Напишете Java конзолна програма, която отпечатва числата от 1 до 20 на отделни редове на конзолата.

1. Създайте нов Java клас със име “Nums1To20“ (десен бутон върху “**src**”папката 🡺 New 🡺 Java Class):



1. Направете си **main** метод
2. Напишете 20 команди System.out.println(); една след друга, за да отпечатате числата от 1 до 20:



1. **Тествайте** вашето решение на задачата в judge системата: <https://judge.softuni.bg/Contests/Practice/Index/150#2>
2. Можете ли да напишете програмата по **по-умен начин**, така че да не повтаряте 20 пъти една и съща команда? Потърсете в Интернет информация за „[**for loop Java**](https://www.google.com/search?q=for+loop+java)“.

## Триъгълник от 55 звездички

Напишете Java конзолна програма, която отпечатва **триъгълник от 55 звездички**, разположени на 10 реда:

|  |
| --- |
| \*  \*\*  \*\*\*  \*\*\*\*  \*\*\*\*\*  \*\*\*\*\*\*  \*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*\* |

1. Създайте ново конзолно Java приложение с име “TriangleOf55Stars”.
2. Напишете код, който печата триъгълника от звездички, например чрез 10 команди, подобни на System.out.println("\*").
3. **Тествайте** кода си в judge системата: <https://judge.softuni.bg/Contests/Practice/Index/150#3>.
4. Опитайте да подобрите решението си, така че да няма много повтарящи се команди. Може ли това да стане с **for цикъл**?

## Лице на правоъгълник

Напишете Java програма, която прочита от конзолата две числа **a** и **b**, въведени от потребителя, пресмята и отпечатва **лицето на правоъгълник** със страни **a** и **b**. Примерен вход и изход:

|  |  |  |
| --- | --- | --- |
| **a** | **b** | **area** |
| 2 | 7 | 14 |
| 7 | 8 | 56 |
| 12 | 5 | 60 |

1. Направете конзолна Java програма. За да прочетете двете числа, използвайте следния код:
2. Допишете програмата по-горе, за да пресмята лицето на правоъгълника и да го проверява.

|  |
| --- |
| static void main(String[] args)  {  Scanner console = new Scanner(System.in);  int a = Integer.parseInt(console.nextLine());  int b = Integer.parseInt(console.nextLine());    // TODO: calculate the area and print it  } |

1. Тествайте решението си в judge системата: <https://judge.softuni.bg/Contests/Practice/Index/150#4>.

## \* Квадрат от звездички

Напишете Java конзолна програма, която прочита от конзолата цяло положително число **N,** въведено от потребителя, и отпечатва на конзолата **квадрат от N звездички**, като в примерите по-долу:

|  |  |
| --- | --- |
| **вход** | **изход** |
| 3 | \*\*\*  \* \*  \*\*\* |
| 4 | \*\*\*\*  \* \*  \* \*  \*\*\*\* |
| 5 | \*\*\*\*\*  \* \*  \* \*  \* \*  \*\*\*\*\* |

1. Направете конзолна Java програма. За да прочетете числото **N** (2 ≤ N ≤100), използвайте следния код:

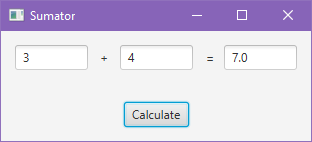
|  |
| --- |
| static void main(String[] args) {  Scanner console = new Scanner(System.in);  int n = Integer.parseInt(console.nextLine());    // TODO: print the rectangle  } |

1. Допишете програмата по-горе, за да отпечатва квадрат, съставен от звездички. Може да се наложи да използвате **for-цикли**. Потърсете информация в Интернет.
2. Тествайте решението си в judge системата: <https://judge.softuni.bg/Contests/Practice/Index/150#5>.

# Упражнения: Графични и Web приложения

## Графично приложение „Суматор за числа“

Напишете **графично (GUI) приложение**, което изчислява **сумата на две числа**:

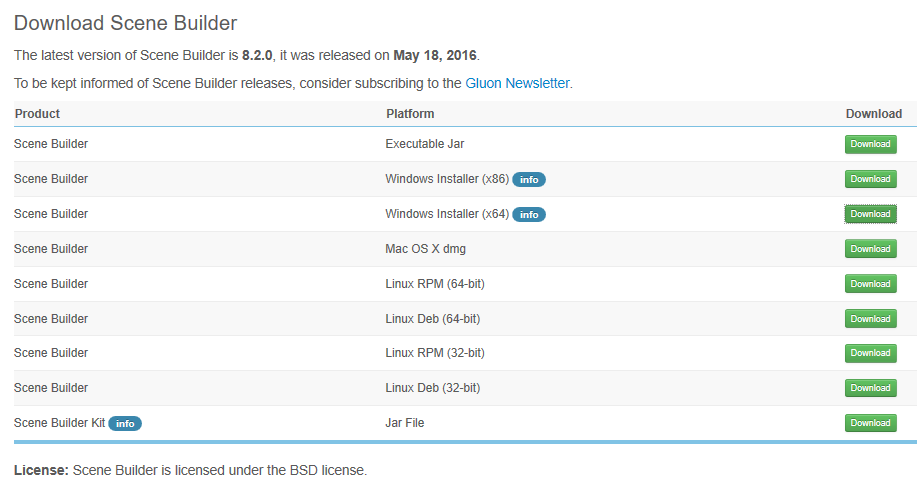


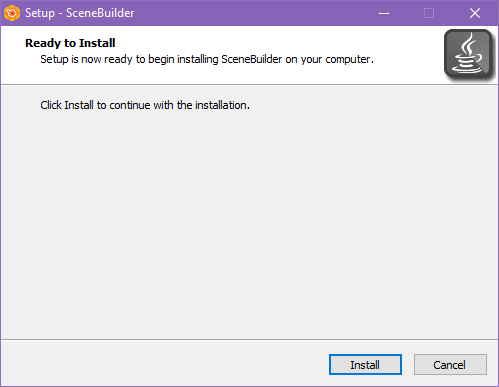
При въвеждане на две числа в първите две текстови полета и натискане на бутона [Calculate] се изчислява тяхната сума и резултатът се показва в третото текстово поле.

За разлика от конзолните приложения, които четат и пишат данните си във вид на текст на конзолата, **графичните (GUI) приложения** имат визуален потребителски интерфейс. Графичните приложения (настолни приложения, desktop apps) се състоят от един от няколко графични прозореца, в които има контроли: текстови полета, бутони, картинки, таблици и други.

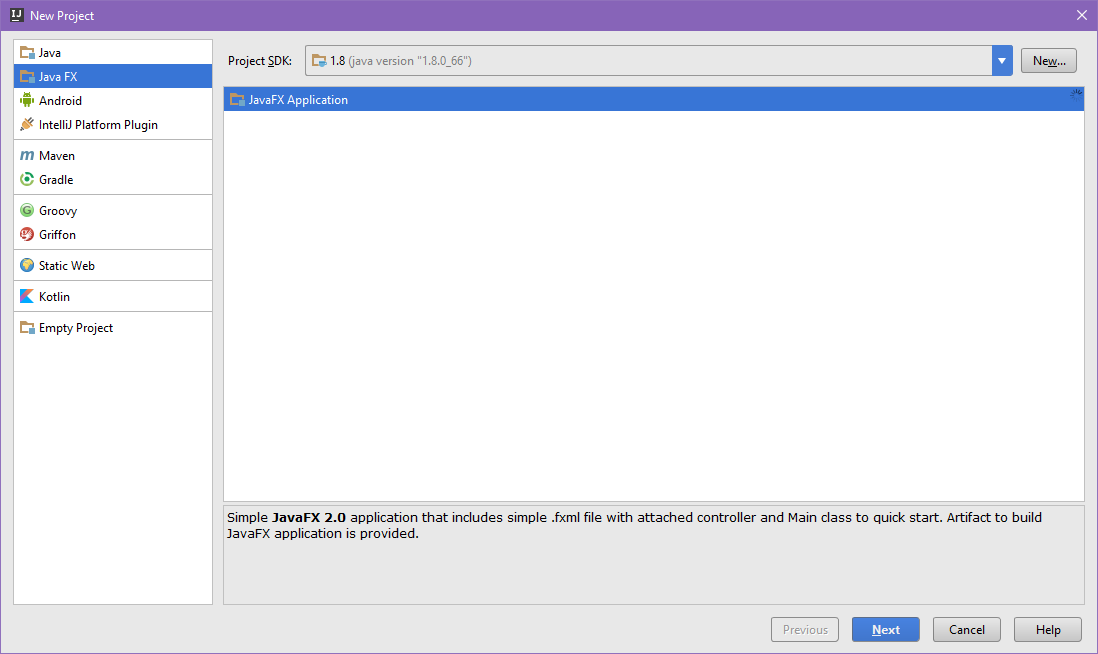
За нашето приложение ще използваме технологията **JavaFX**, която позволява създаване на графични приложения за всички платформи с езика за програмиране **Java**. За среда на разработка ще ползваме програмата **IntelliJ IDEA.**

За да направим по-лесно създаването на графични приложения, ще ползваме програмата **SceneBuilder**, която ще изтеглим от [**тук**](http://gluonhq.com/labs/scene-builder/#download):

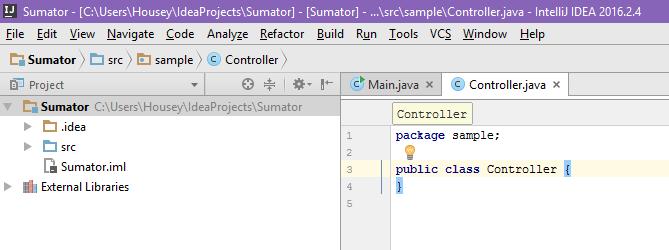




1. В IntelliJ IDEA създайте нов Java проект от тип „**JavaFX Application**“:



1. При създаването на JavaFX приложение ще се появи прозорец, който изглежда така:



**NB:** Ако по някаква причина не виждате прозореца Project, можете да го отворите от **View -> Tool Windows -> Project**.

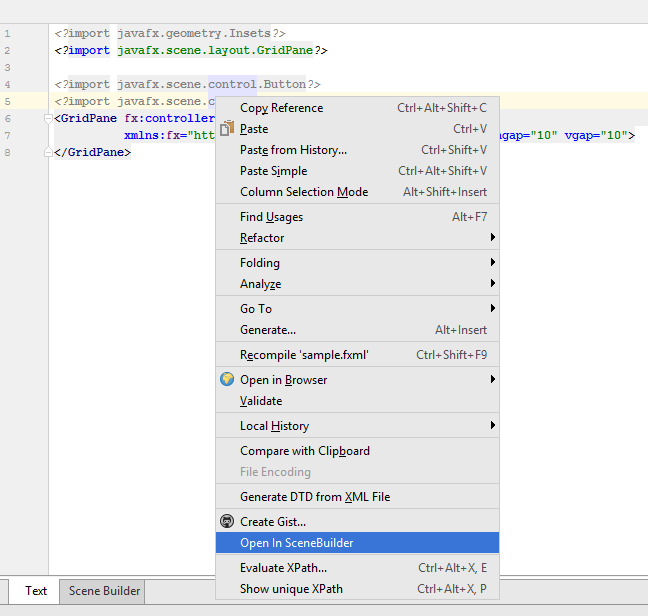
1. Файлът, в който се намира изгледа на нашето приложение се намира в пътя **src/sample/sample.fxml**.

Нека да отидем там и да го отворим:

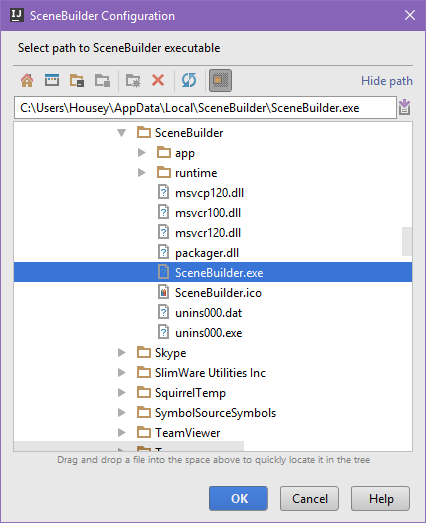


Файлът изглежда по този начин. Няма да работим директно с него, а ще ползваме горепосочения **SceneBuilder**, който ще генерира кода вместо нас.

1. Кликваме с десен бутон на файла и избираме **„Open in SceneBuilder”**:

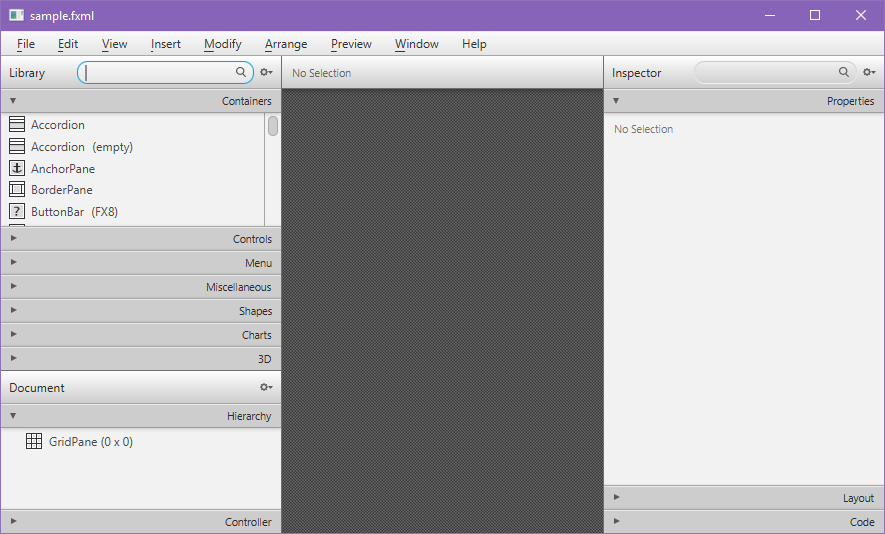


При избирането на тази опция **за първи път**, IntelliJ ще поиска да посочим пътеката към **SceneBuilder.exe**:

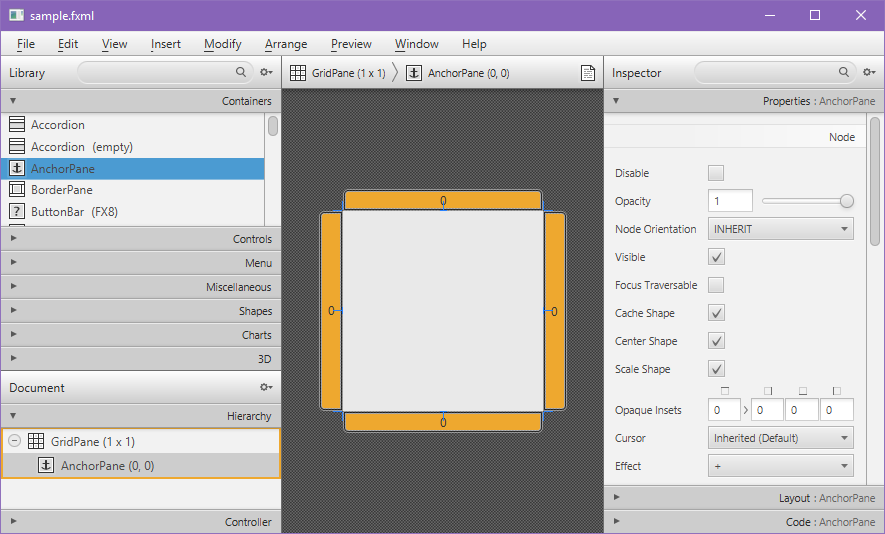


При обикновена инсталация на Windows, **SceneBuilder** се намира в **C:\Users\*[username]*\AppData\Local\SceneBuilder**.

Отваря се SceneBuilder, при което виждаме този екран:



Отляво имаме видовете контроли, които можем да добавяме, така че ще намерим **AnchorPane** прозореца и ще го добавим като го завлечем в средата, където се намира нашият дизайн:



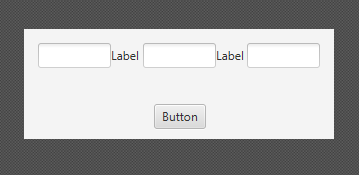
Това е нашият прозорец, в който вече можем да започнем да добавяме контроли, които можем да добавим от същото място, от което добавихме **AnchorPane** **(в менюто Containers)**. За нашият интерфейс ни трябват:

3x текстови полета: **TextField**,

2x етикета между текстовите полета (за *+* и *=*): **Label**

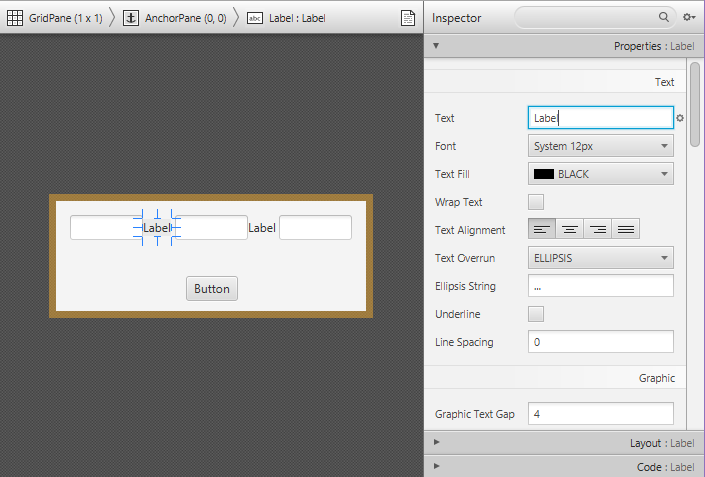
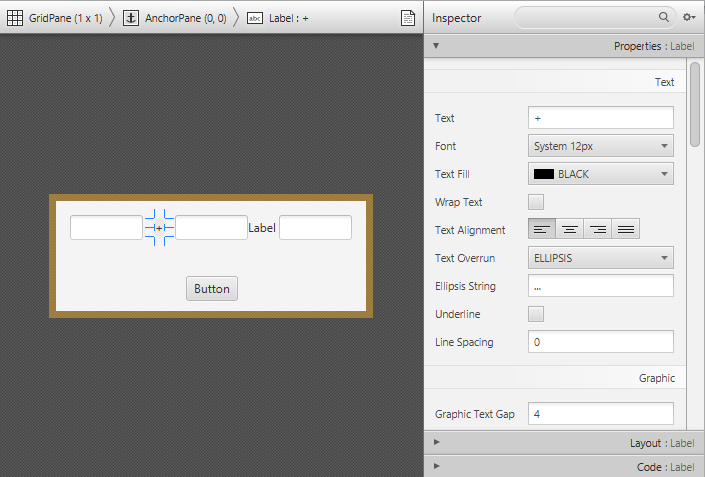
1x бутон за пресмятане на резултата: **Button**

1. След като ги добавим, нашето приложение трябва да изглежда така (все още ни предстои да сменим текста в контролите):

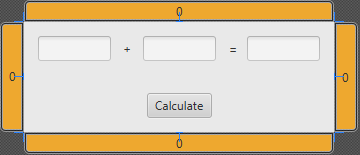


1. Сега ще променим имената на контролите, за да изглежда правилно. Това се осъществява като кликнем на контрола, който искаме да променим и отидем в дясно на неговите свойства (Properties).

За начало ще променим свойството **Text** на етикетите:

 => 

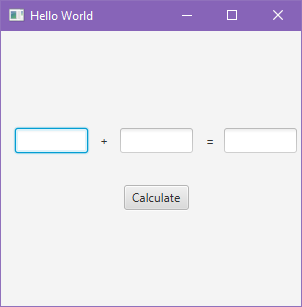
Променете също текста на **другия етикет** и на **бутона**, за да изглежда така:



1. Запазете промените към дизайна с **Ctrl+S**:



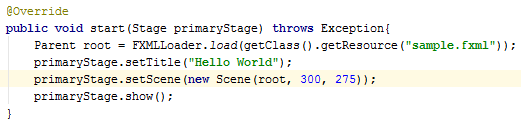
1. Върнете се обратно в **IntelliJ** и **Стартирайте** **приложението** с [Shift+F10]:



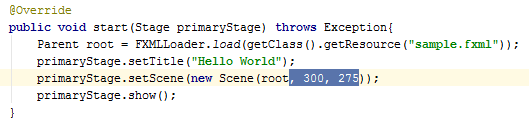
Вероятно приложението изглежда по-голямо или по-малко, отколкото очаквахме. Защо?

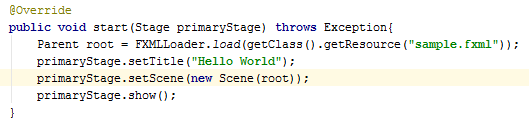
Причината е, че в кода на нашето приложение има една част, която казва изрично какви да са неговите размери. Нека оправим този проблем:

1. Отидете в класа Main и намерете това парче код:

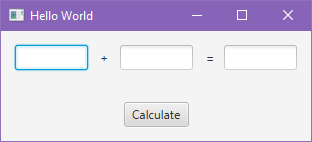


1. Изтрийте кода, който казва на нашата програма колко ще е голям прозореца, за да се използва оразмеряването, което ние избрахме в SceneBuilder:



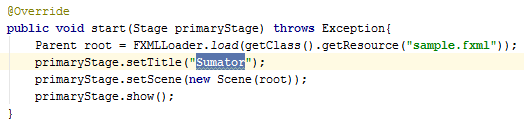
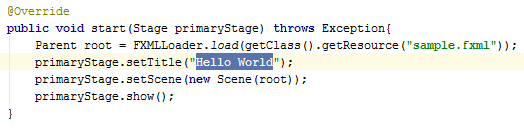


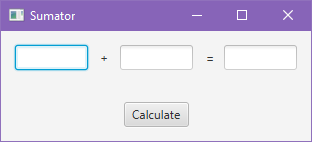
1. Пуснете програмата пак с [Shift+F10] и проверете дали оразмеряването е правилно:



Изглежда правилно!

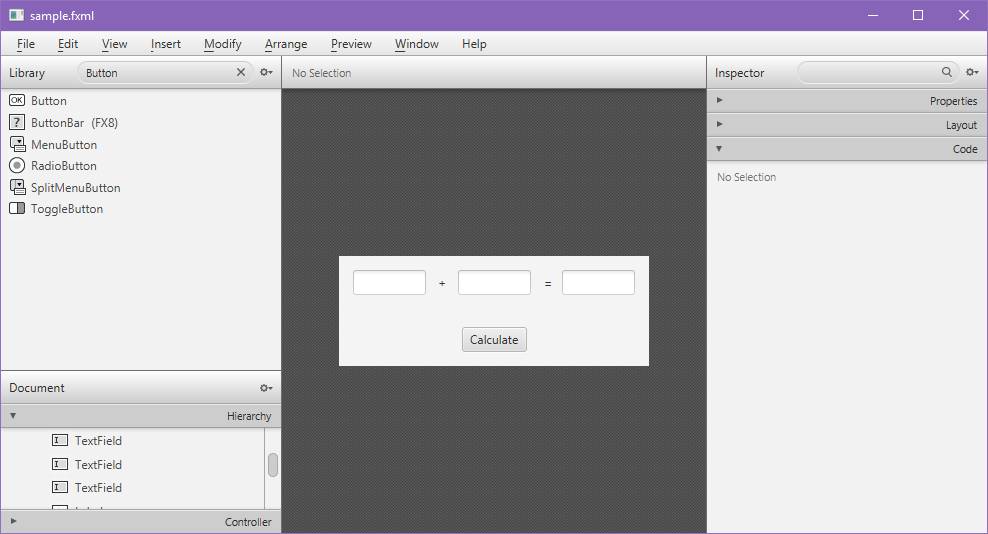
1. Като последна стъпка за завършването на нашия дизайн, ще се върнем в нашия код и ще променим **заглавието** на приложението, което се намира на този ред:



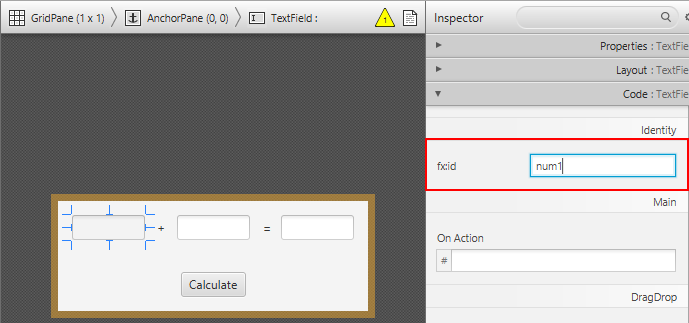


Дизайна изглежда готов. Сега е време за нещо доста по-интересно - програмната логика!

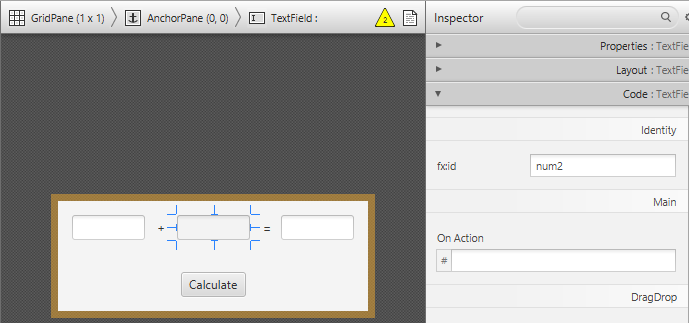
1. Време е **да напишете кода, който сумира числата** от първите две полета и показва резултата в третото поле. За целта трябва да се върнем обратно в **SceneBuilder (3-та точка)** и да дадем имена на текстовите полета, за да можем да ги достъпваме в кода:

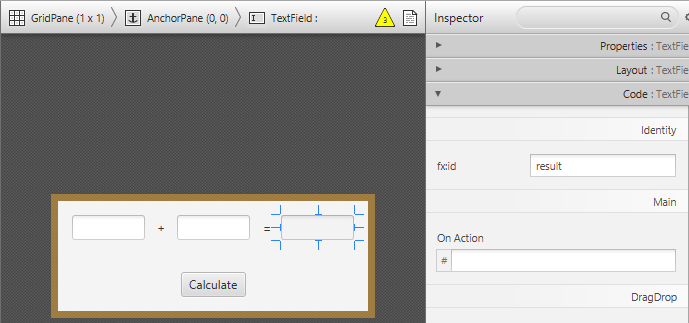


Кликваме първото текстово поле с мишката и отиваме в секцията **Code** отдясно, където ще дадем името **num1** в полето **fx:id**:

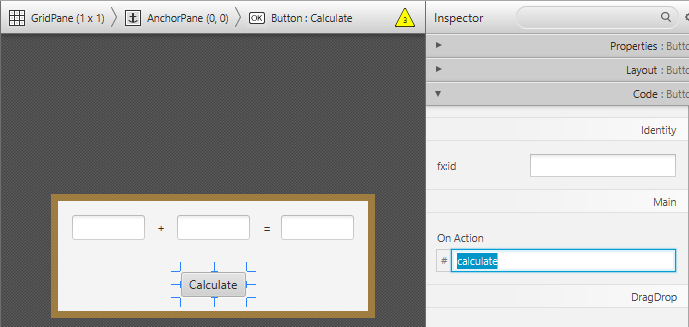


1. Ще направим същото и за останалите полета:





1. Сега ще кажем на бутона коя функция да изпълнява, когато той бъде кликнат. Отиваме в **On Action** полето и му подаваме името на функцията **calculate**, която след малко ще създадем:



1. Запишете промените в SceneBuilder с [Ctrl+S] и се върнете обратно в **sample.fxml** файла:

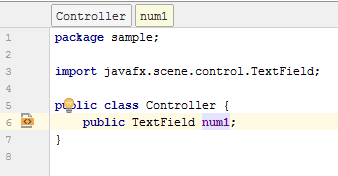


Виждаме, че нашите полета си имат имена, а нашият бутон си има функция, която да се изпълнява, когато той бъде кликнат.

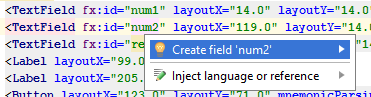
До тук добре, но тези полета и функцията на бутона все още не съществуват в нашия код. За да ги генерираме, ще отидем на всеки от тях и ще натиснем **[Alt+Enter]->Create field *[име на полето]***:

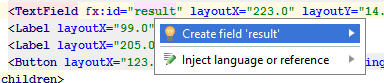


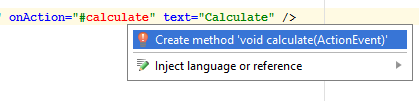
След като създадем едно от полетата, ще бъдем пренасочени към файла **Controller.java**, където ще се създаде полето в кода по следния начин:



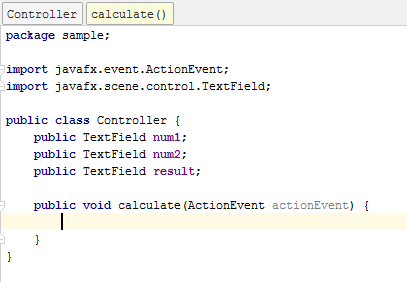
1. Добавете по същия начин полетата **num2**, **result** и функцията **calculate**:



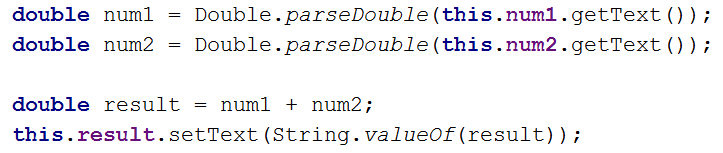




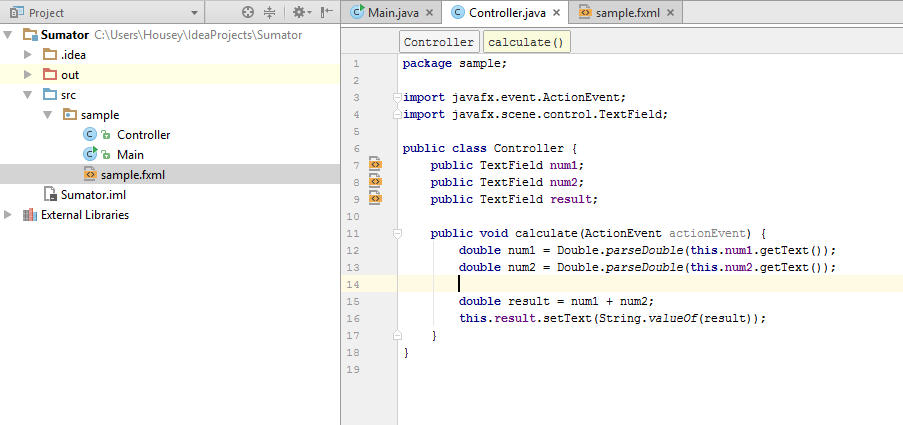
1. След като добавим нашите полета и функцията calculate, кода в файла Controller.java би трябвало да изглежда така:



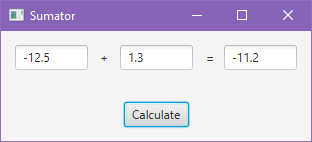
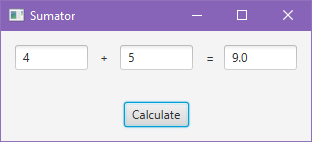
1. Напишете следния Java код между отварящата и затварящата скоба { }, където е курсорът:



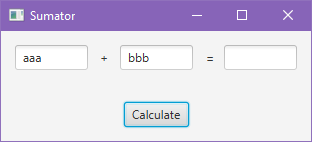
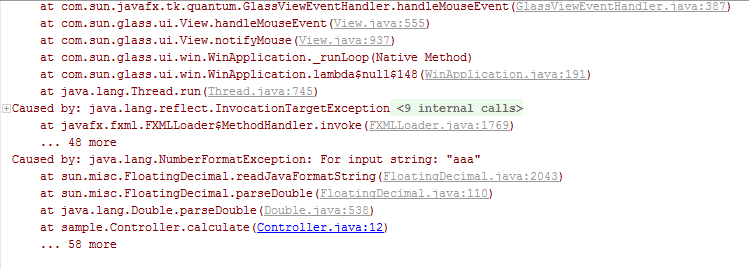
Този код взима първото число от полето num1 в променлива num1, след това второто число от полето num2 в променлива num2, след това ги сумира num1 + num2 в променлива result и накрая извежда текстовата стойност на result в полето result.



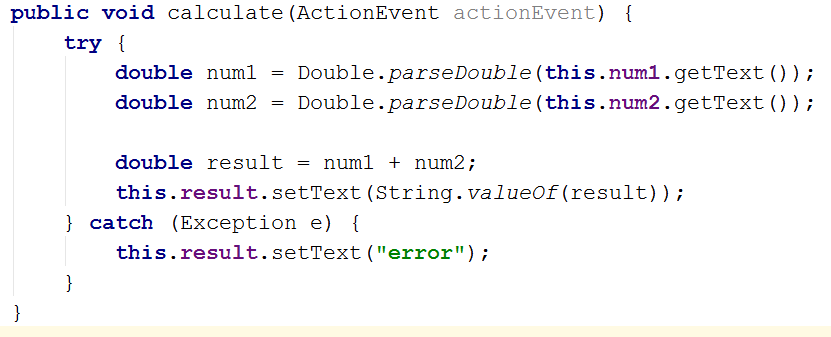
1. **Стартирайте отново** програмата с [Shift+F10] и я **пробвайте дали работи**. Пробвайте да сметнете 4 + 5. След това пробвайте да сметнете -12.5 + 1.3:



1. Пробвайте с **невалиден вход**, примерно “**aaa**” и “**bbb**”. Изглежда има проблем:

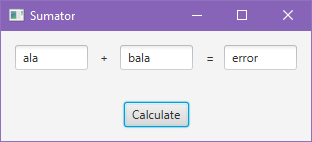
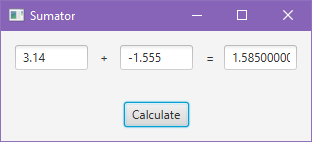
 

1. Проблемът идва от прехвърлянето на текстово поле в число. Ако стойността в полето не е число, програмата се чупи и **дава грешка**. Можете да поправите кода, за да решите проблема така:



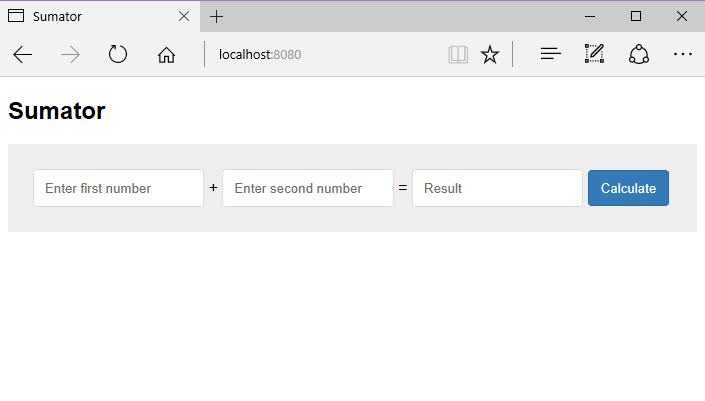
Горният код прихваща грешките при работа с числа (**хваща изключенията**) и в случай на грешка извежда стойност “**error**” в полето с резултата.

1. Стартирайте отново програмата с [Shift+F10] и я **пробвайте дали работи**. Този път при грешно число резултатът е “**error**” и програмата не се чупи.



## Уеб приложение „Суматор за числа“

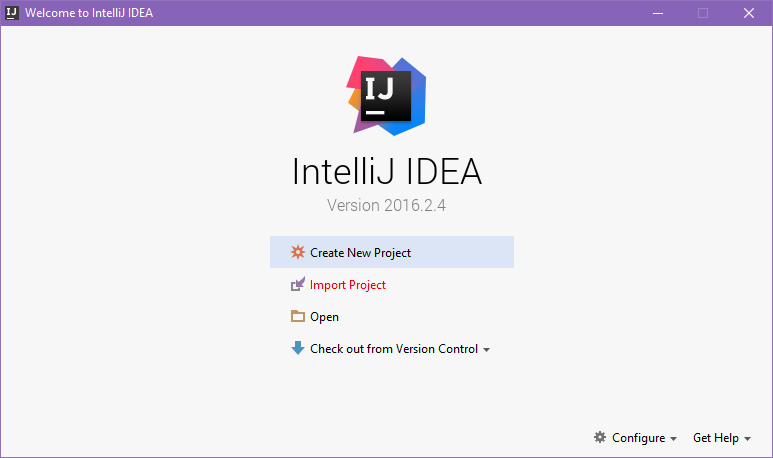
Напишете **уеб приложение**, което изчислява **сумата на две числа**. При въвеждане на две числа в първите две текстови полета и натискане на бутона [Calculate] се изчислява тяхната сума и резултатът се показва в третото текстово поле. Уеб приложението би могло да изглежда по следния начин:



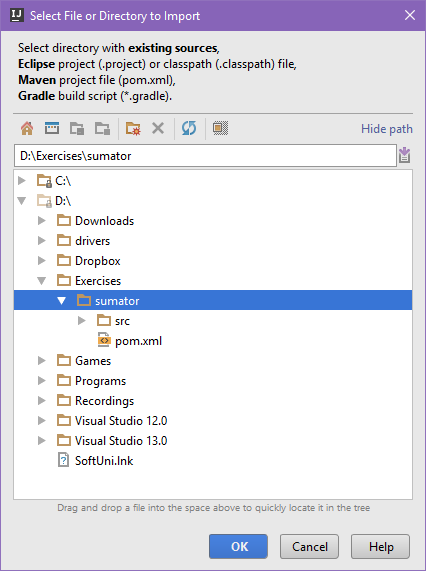
За разлика от конзолните приложения, които четат и пишат данните си във вид на текст на конзолата, **уеб приложения** имат **уеб базиран потребителски интерфейс**. Уеб приложенията се зареждат от някакъв Интернет адрес (URL) чрез стандартен **уеб браузър**. Потребителите пишат входните данни в страница, визуализирана от уеб приложението, данните се обработват на уеб сървъра и резултатите се показват отново в страницата в уеб браузъра.

За нашето уеб приложение ще използваме технологията **Spring MVC**, която позволява създаване на уеб приложения с езика за програмиране **Java** в средата за разработка **IntelliJ IDEA**.

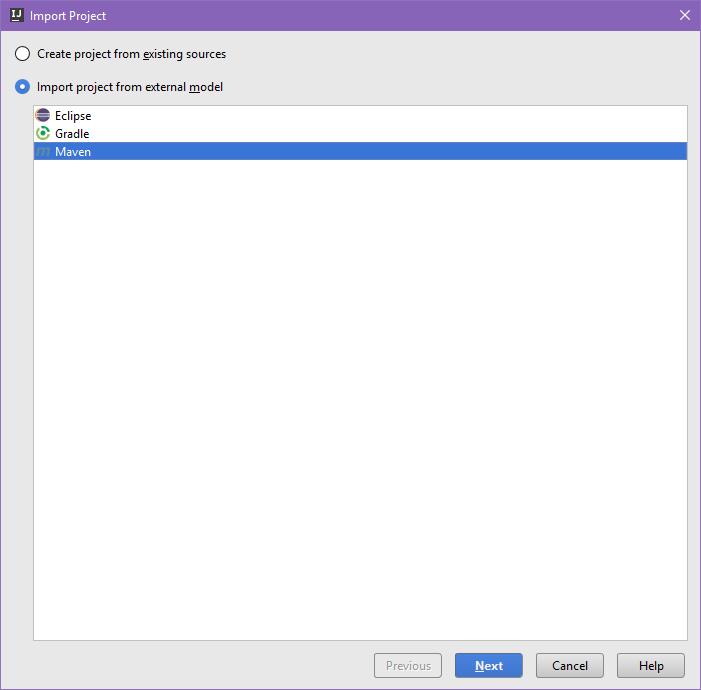
1. В IntelliJ, вкарайте проекта от [скелета](https://softuni.bg/downloads/svn/programming-basics/2016-12/Java/1.%20First-Steps-in-Coding-Exercises-Web-Skeleton.zip), чрез **Import Project**:

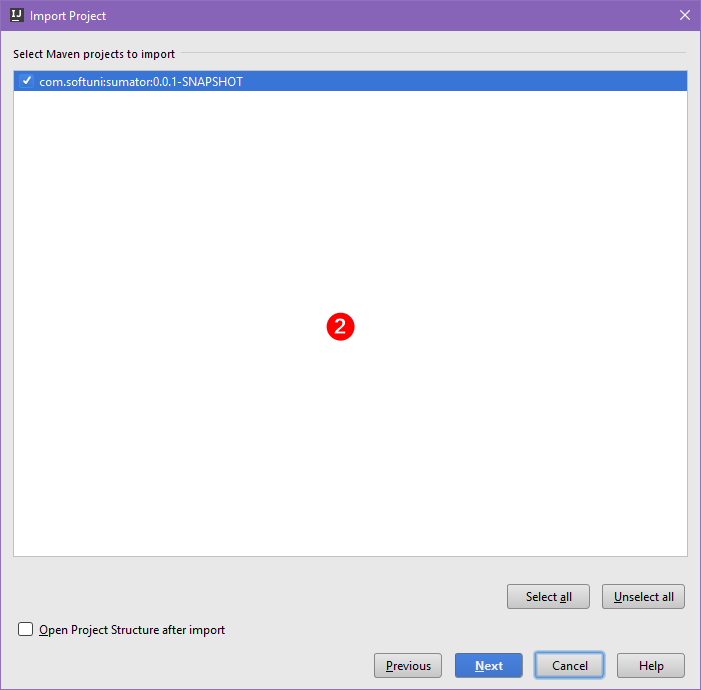
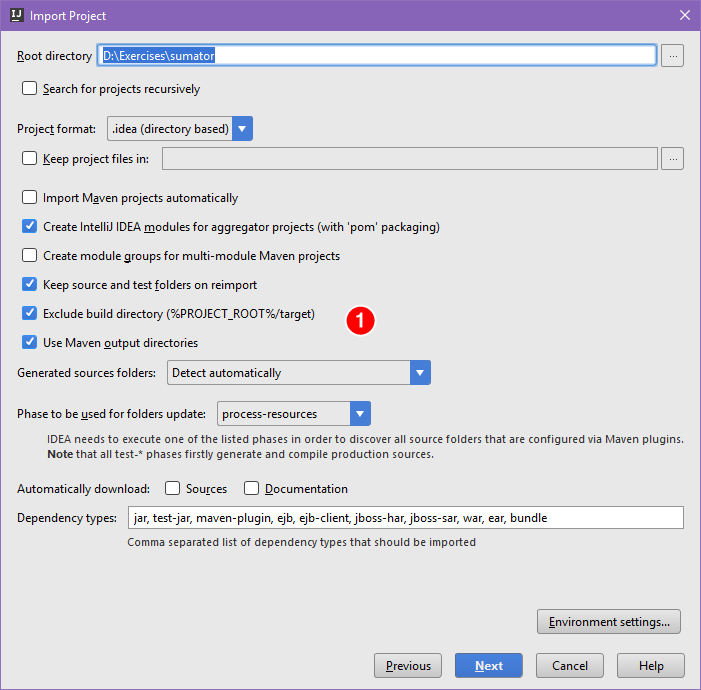


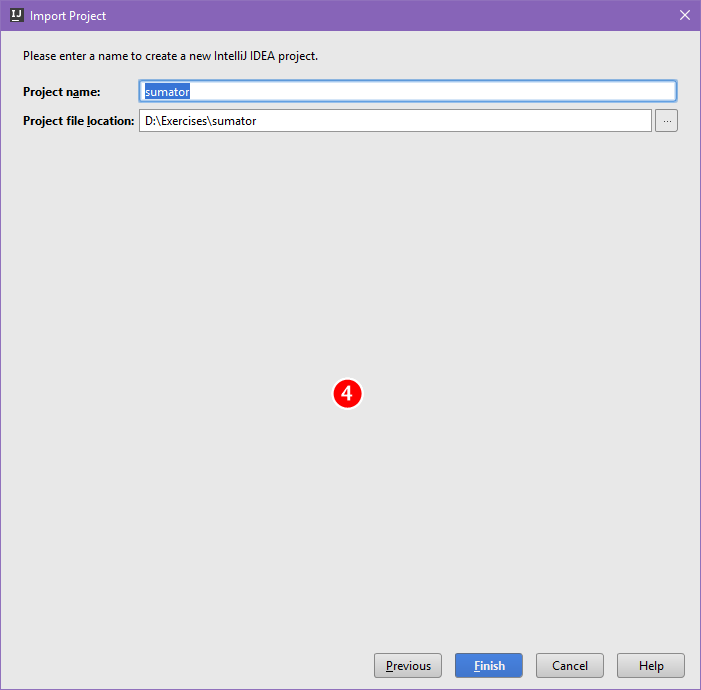
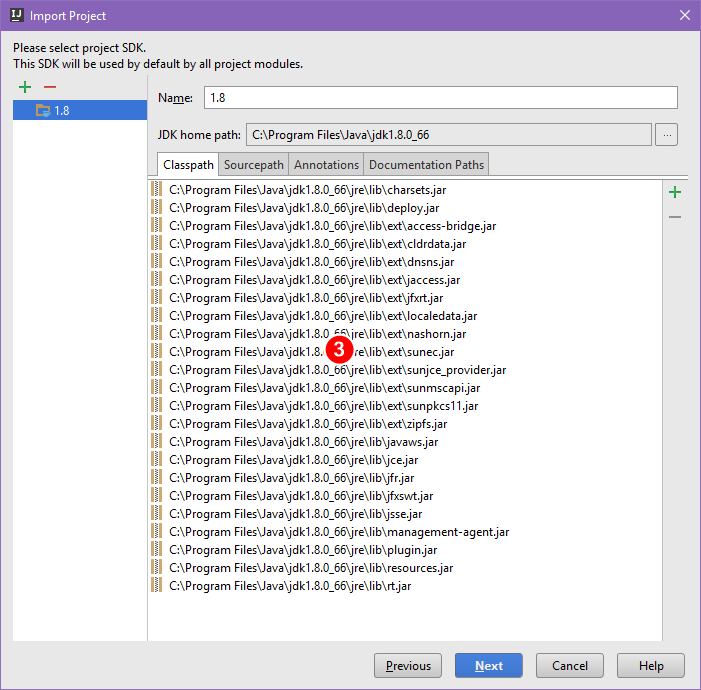
1. Намерете папката, където се намира скелета:



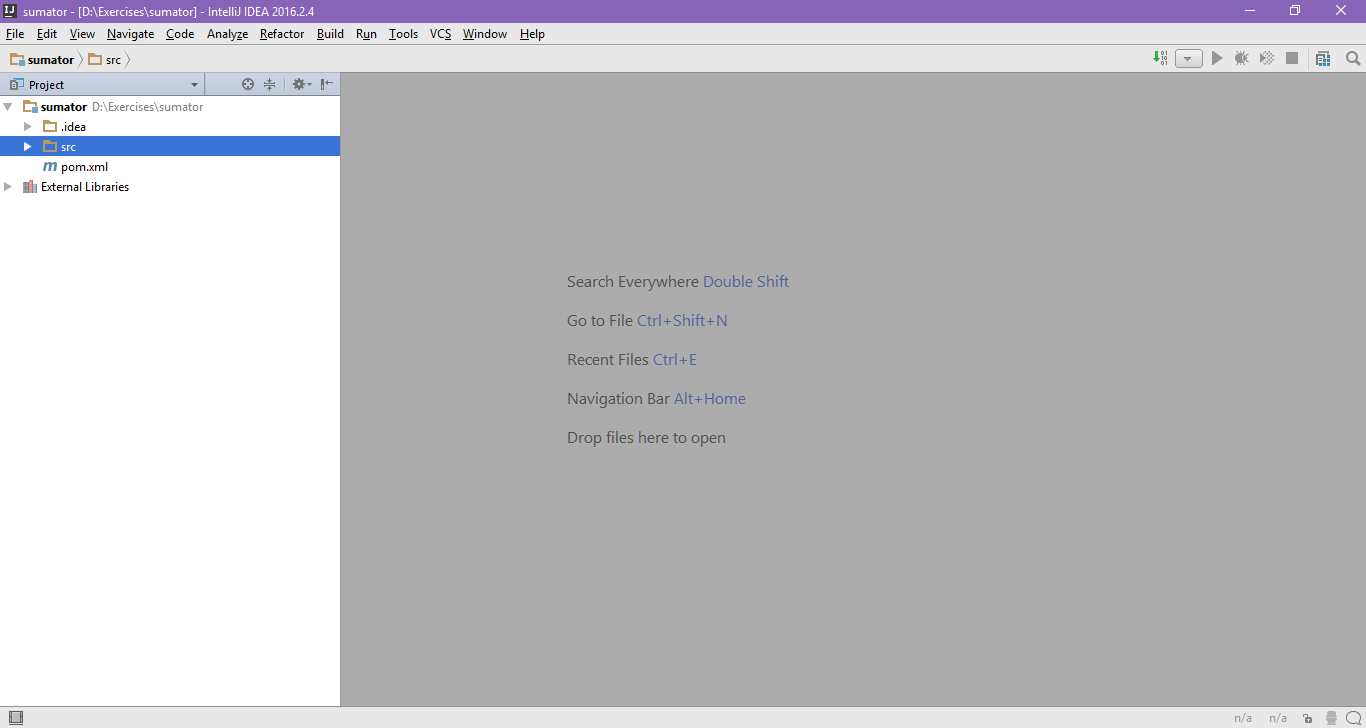
1. Изберете „**Import project from external model**” и след това „**Maven**”:



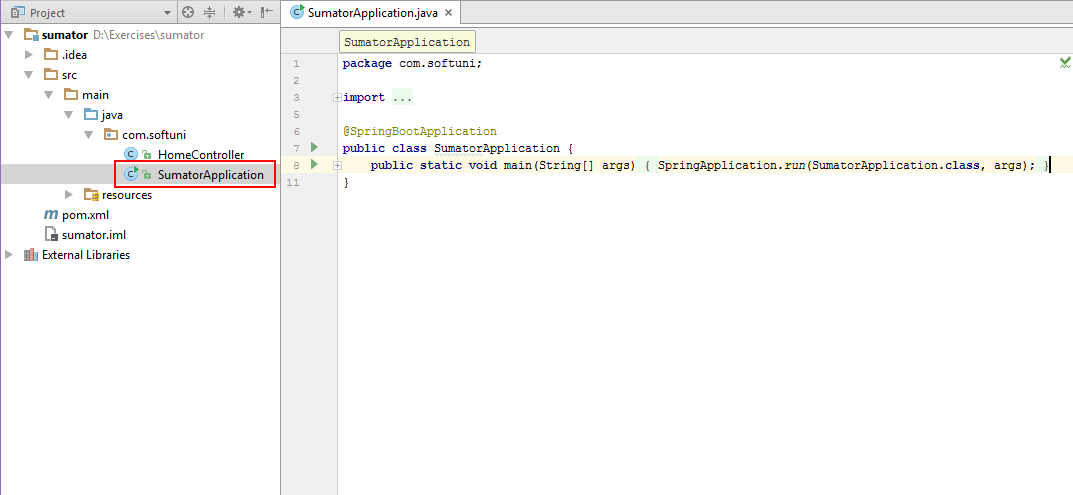




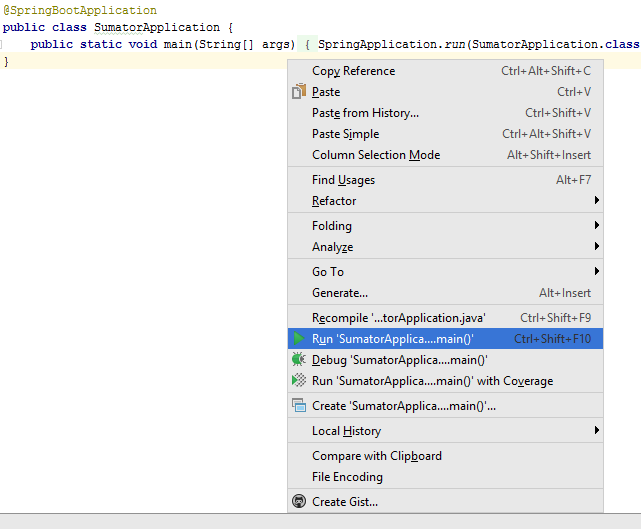
1. След като вкараме проекта в IntelliJ, той би трябвало да изглежда така:



1. Нека пуснем проекта, за да видим дали тръгва. За да осъществим това, ще отидем в папката “**src/main/java/com.softuni”** и отворим файла **SumatorApplication**:



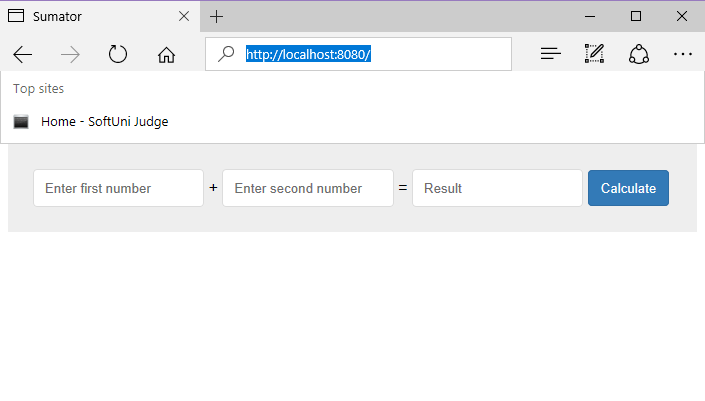
За да пуснем програмата отиваме в редактора и натискаме **[Ctrl+Shift+F10]**, или натискаме **[Run ‘SumatorApplication.main()’]** в контекстното меню:



1. След като уеб приложението зареди, би трябвало да видим това съобщение **най-отдолу** в конзолата, която се отваря:

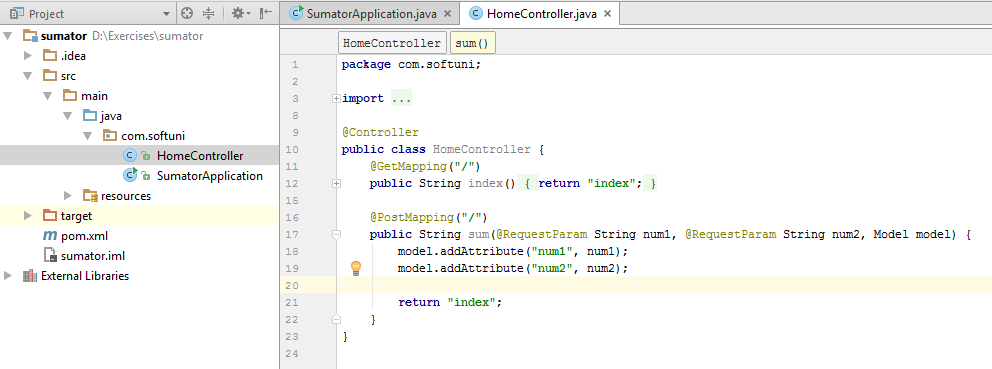


1. Можем да пуснем приложението и да проверим дали работи като отидем в нашия уеб браузър и напишем „**localhost:8080**“:

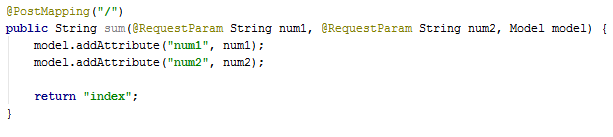


До тук добре, само че когато въведем две числа и натиснем “Calculate”, не става нищо. Нека да напишем логиката, която ще направи суматора да работи

1. Отваряме файла „**HomeController**“ в същата папка и би трябвало да видим следното:



Частта, която ни интересува е функцията **“sum”:**



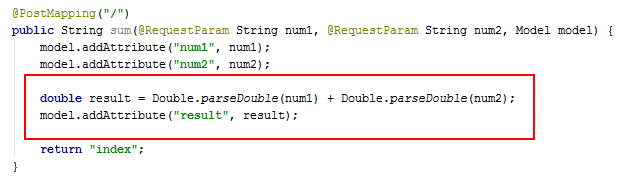
В момента, тази функция приема два текста **num1** и **num2**, подава ги на сървъра и връща файла “index” на потребителя. Както може би се досещате, вътре няма код, който пресмята числата в двете текстови полета и ги подава на третото поле.

Нека напишем логиката, която ще осъществи това.

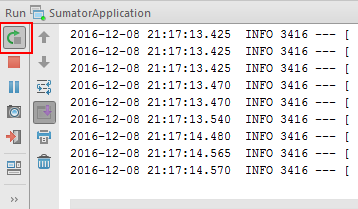
1. Отиваме между къдравите скоби и написваме следното:



1. Ето как трябва да изглежда файлът HomeController след промяната:



1. Преди да се върнем обратно в приложението, трябва да приложим нашите промени. Ще направим това като отидем в прозореца на конзолата и натиснем бутона „Rerun application”:



1. Приложението е готово. Ако въведем две числа в текстовите полета и натиснем бутона “**Calculate**”, би трябвало резултата да излезе в третото текстово поле:

