

# Лекция 9с

Създаване на многократно използваеми  
JavaFX компоненти с графичен визуален  
редактор.

## 4.9 Създаване на многократно използвани FXML графични компоненти

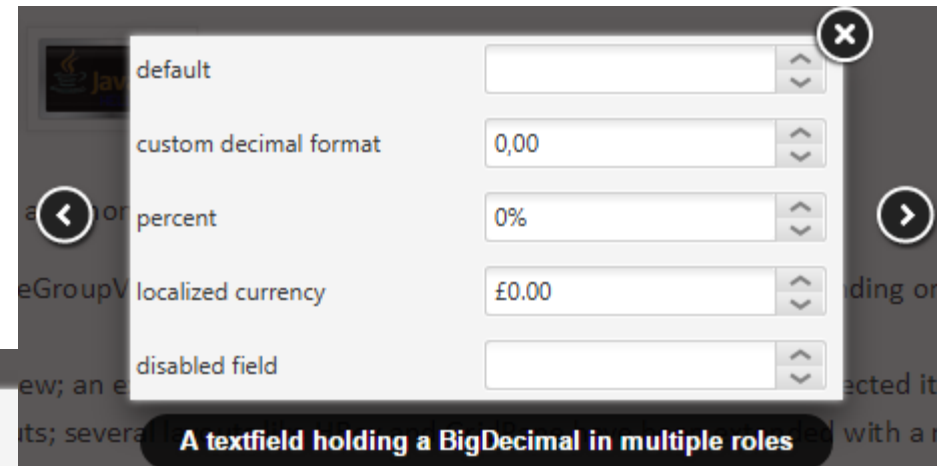
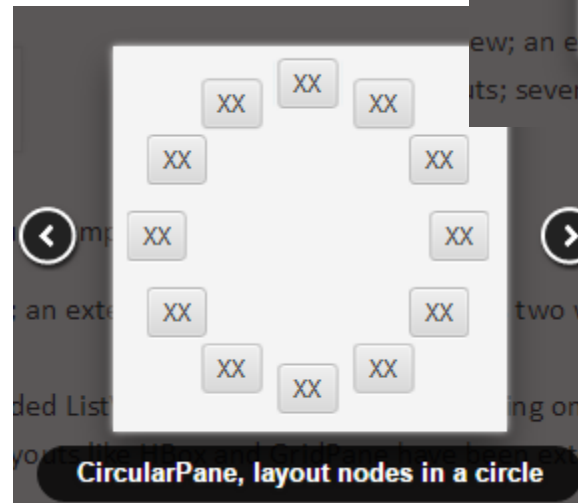
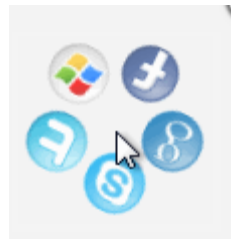
## 9 Създаване многократно използваеми компоненти в JavaFX

Създаването на потребителски дефинирани компоненти с JavaFX позволява многократното използване на тези компоненти в различни проекти. На практика JavaFX предоставя неограничени възможности за създаване на такива компоненти. Тези компоненти могат да се разпространяват с техния **FXML** код или като **JAR** файл. В последният случай това позволява да се разпространяват компоненти, които обединяват графичен модел, заедно със съпътстващ ги сорс код на Java за описание на методи за обработка на събития или удовлетворяване на други функционални изисквания на потребителя.

# 9 Създаване многократно използваеми компоненти в JavaFX

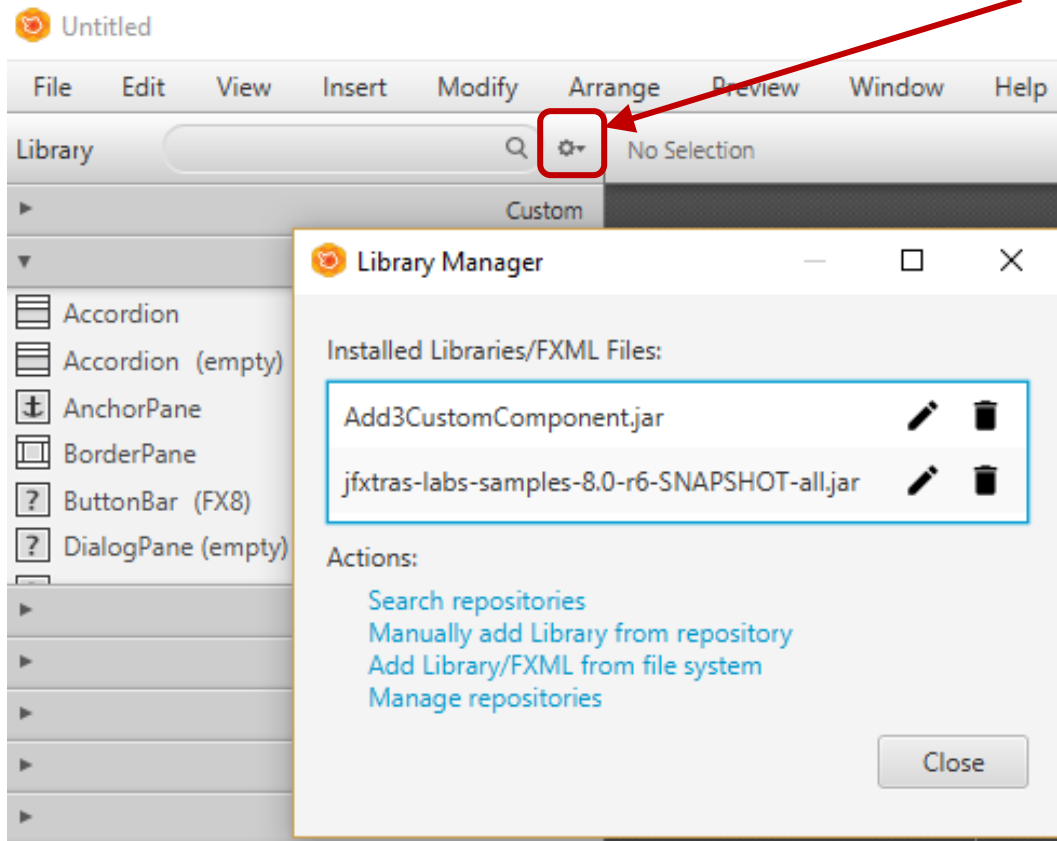
Примери на такива компоненти са компоненти с отворен сорс код, които са достъпни в Internet

- JFXtra.



# 9 Създаване многократно използваеми компоненти в JavaFX

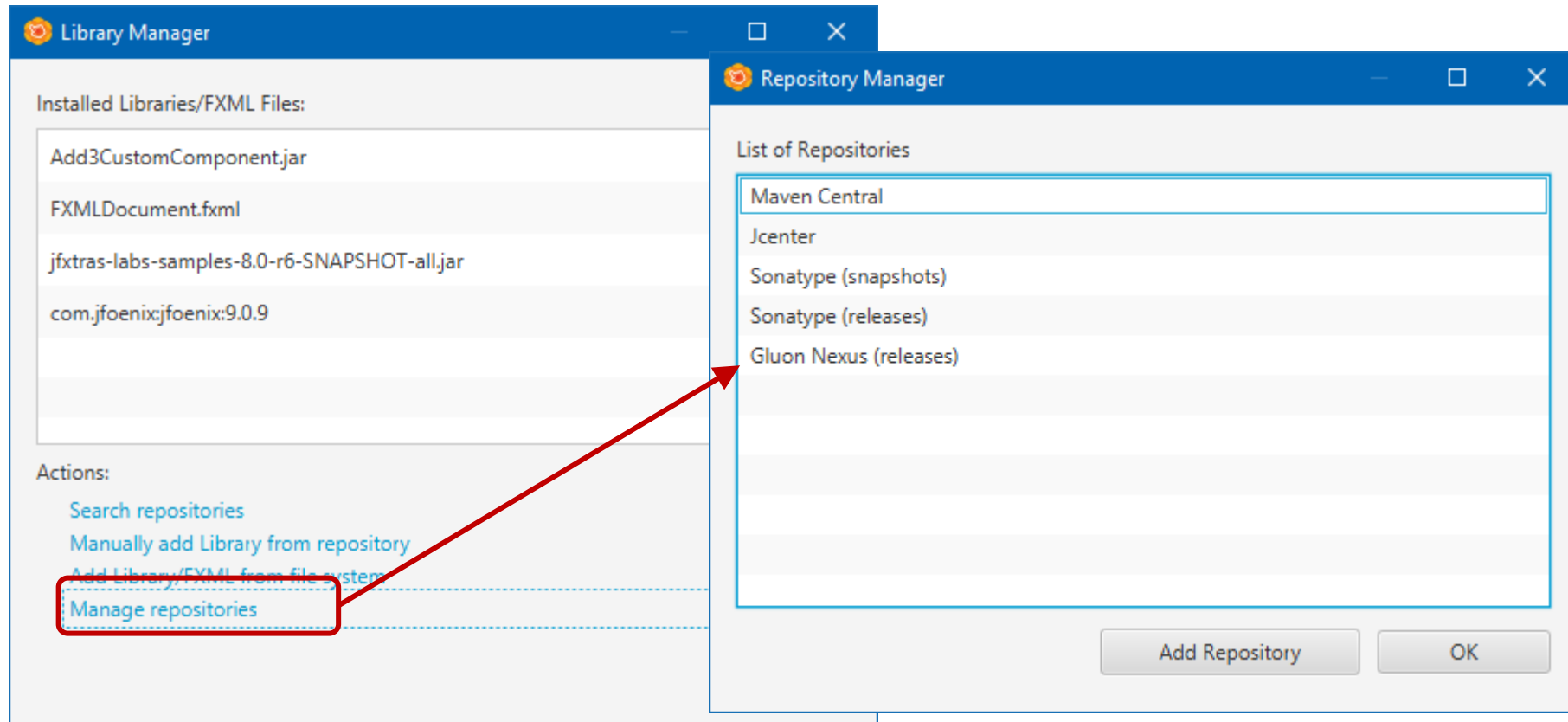
Тези компоненти могат да се интегрират с другите графични компоненти, налични по стандарт в **SceneBuilder** като се използва **Library Manager**



**Забележка:** Не всички JFXtra компоненти са съвместими с версия на Scene Builder по-висока от 8.

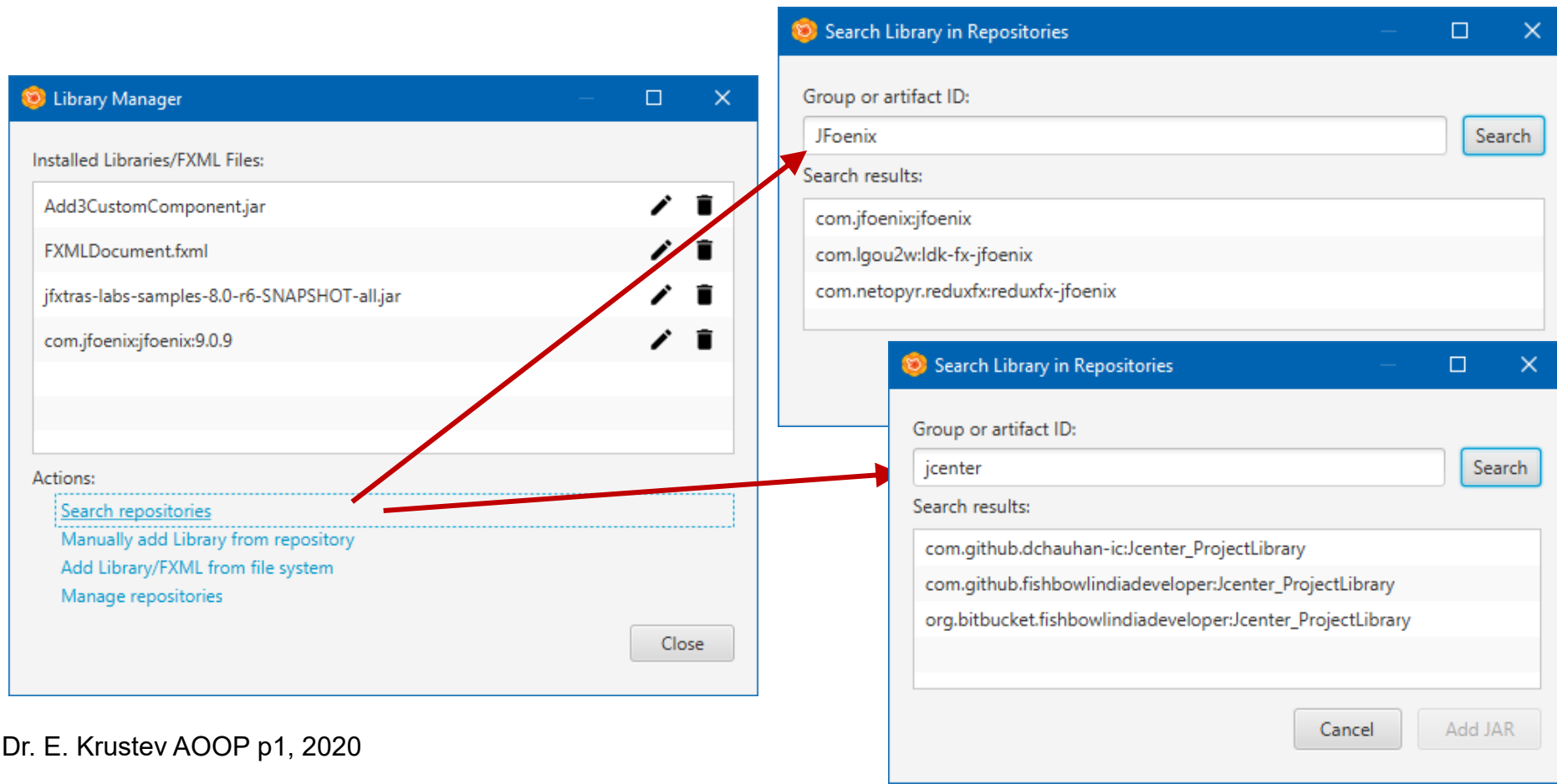
# 9 Създаване многократно използваеми компоненти в JavaFX

По подразбиране **SceneBuilder** поддържа няколко доставчици на свободни графични компоненти. като се използва **Library Manager**



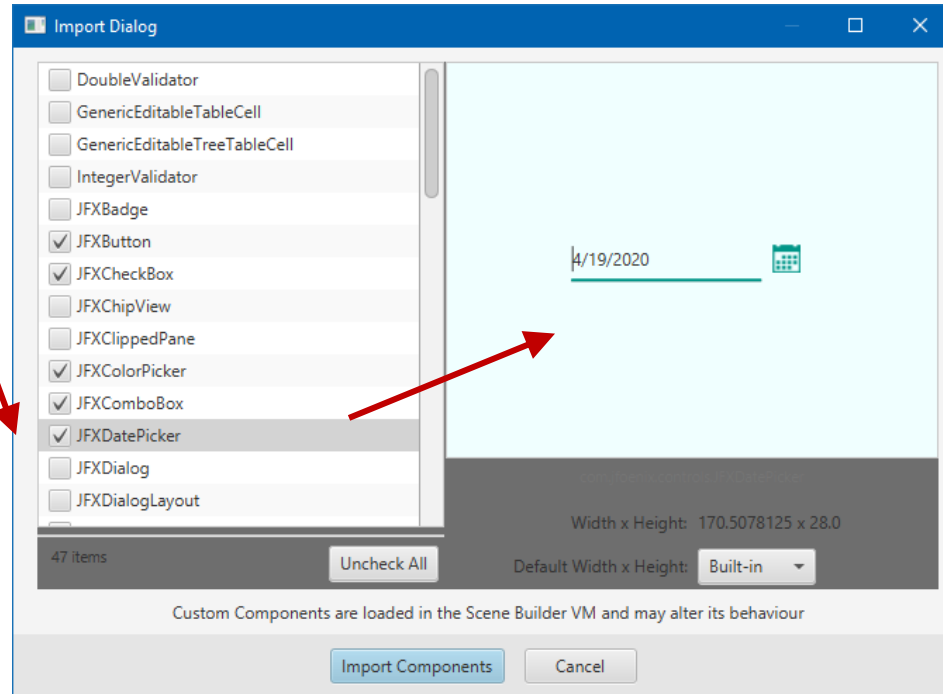
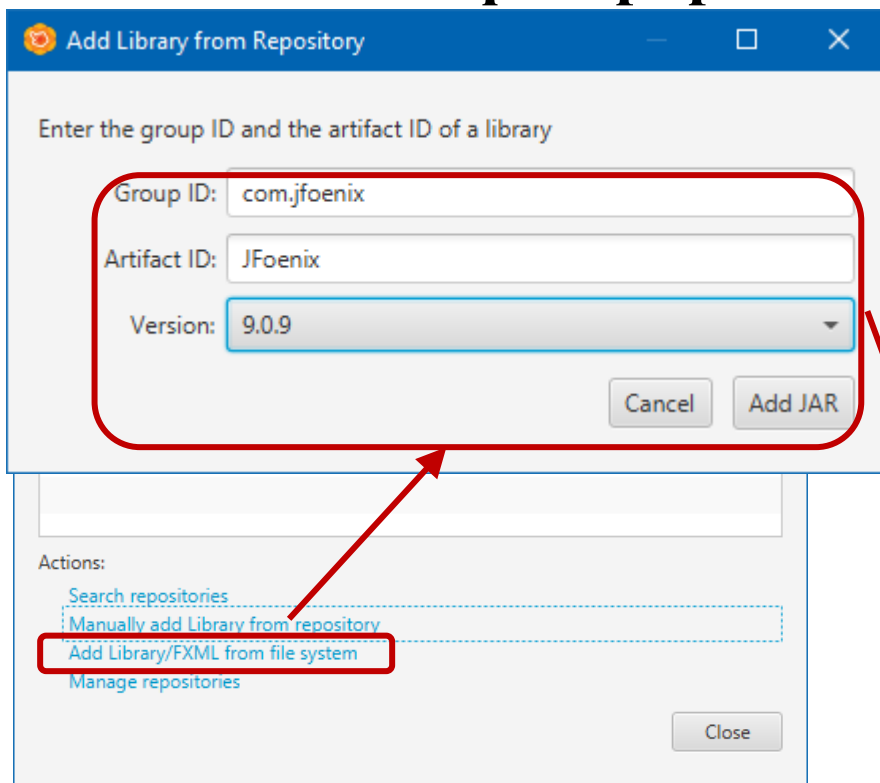
# 9 Създаване многократно използваеми компоненти в JavaFX

Добавяне в `SceneBuilder` на потребителски компоненти от тези доставчици се извършва като се изпълни `Search` по името на доставчика.



## 9 Създаване многократно използваеми компоненти в JavaFX

Малка част от тези доставчици предлагат компоненти, които са съвместими с версии на **SceneBuilder** по-високи от версия 8. Освен **JFXtra**, може да добавите компоненти от наличните доставчици с филтриране по версия на компонентите



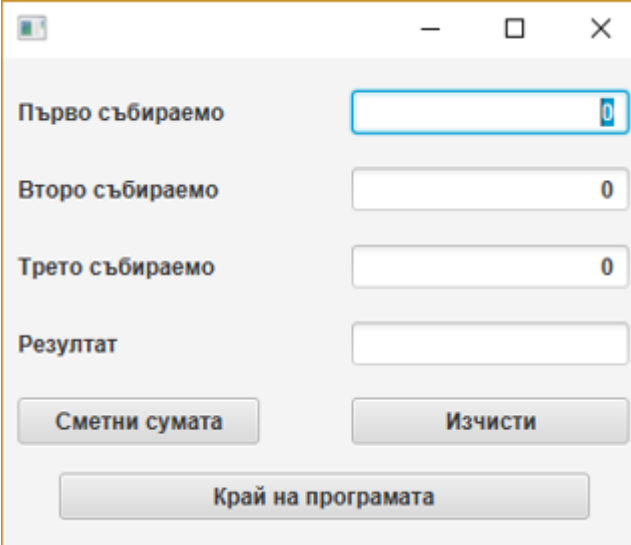


## 9.1 Създаване на проекта

Създаването на потребителски дефинирани компоненти с **JavaFX** не се отличава съществено от създаването на **Java FXML Application**.

Тук ще демонстрираме създаването на графична компонента, която предоставя същия графичен интерфейс, както **JavaFX** приложението, разгледано в **лекция 4**, а именно

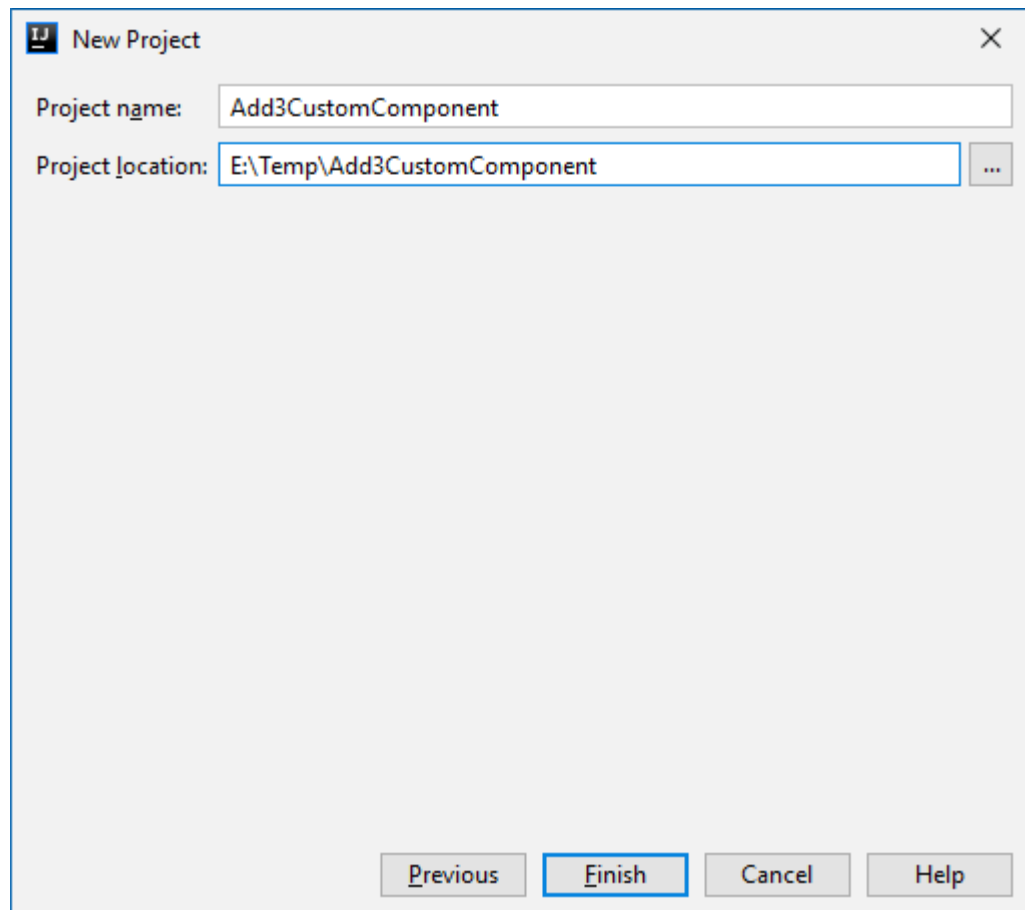
За целта изпълняваме следните действия:



The screenshot shows a JavaFX application window with a light gray background and standard window controls (minimize, maximize, close) in the top right corner. The interface contains four text input fields stacked vertically, each with a label to its left: 'Първо събираемо' (First addend), 'Второ събираемо' (Second addend), 'Трето събираемо' (Third addend), and 'Резултат' (Result). The first three input fields have a blue border and a small '0' button on the right side. The 'Резултат' field is empty. Below the input fields are two buttons: 'Сметни сумата' (Calculate sum) and 'Изчисти' (Clear). At the bottom of the window is a wide button labeled 'Край на програмата' (End of program).

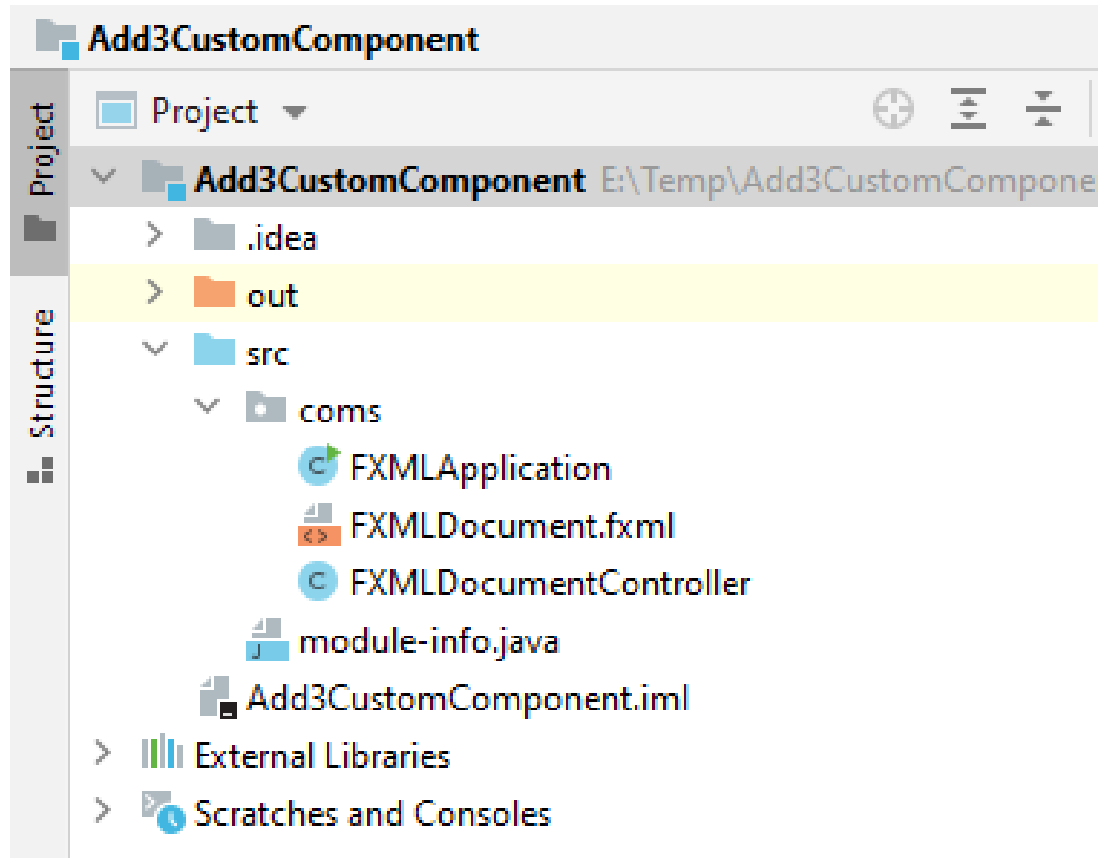
# 9.1 Създаване на проекта

## 1. Създаваме JavaFX FXML Application Add3CustomComponent



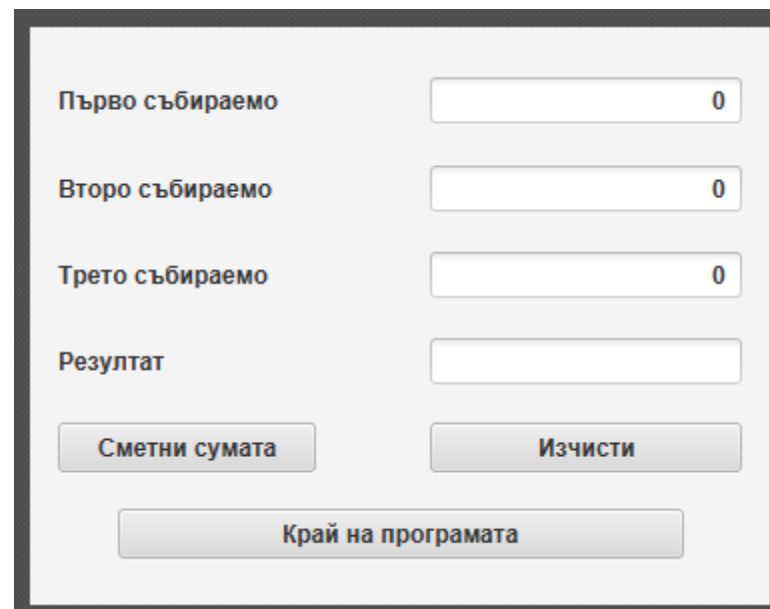
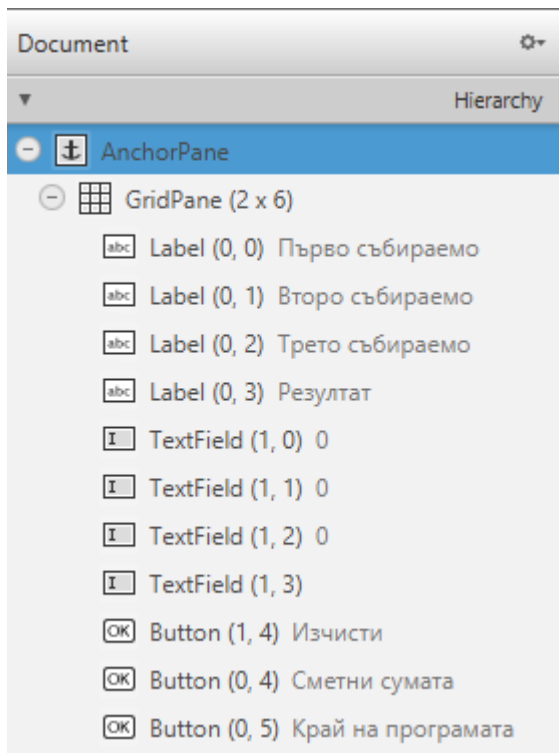
# 9.1 Създаване на проекта

## 2. Създаваме **JavaFX FXML Application** **Add3CustomComponent**



## 9.1 Създаване на проекта

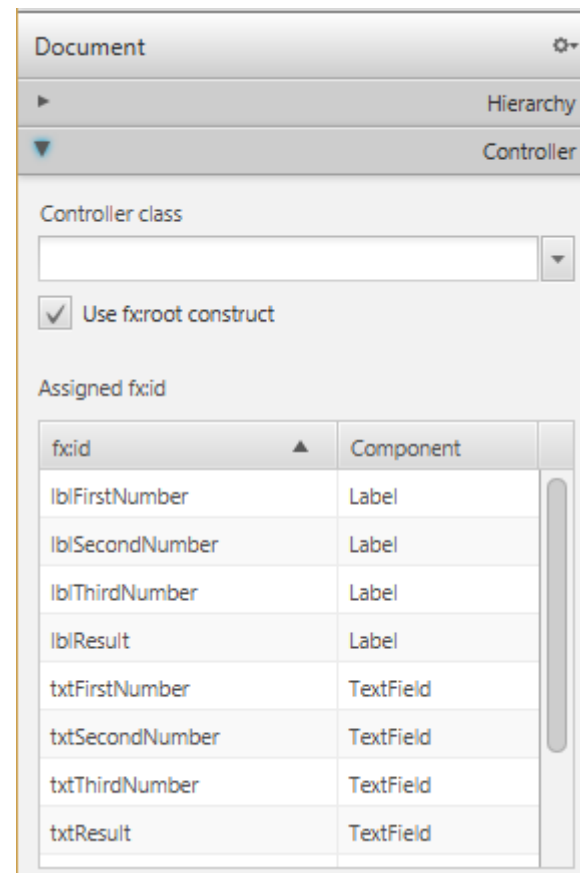
Създаваме дървото от възли на Графичния интерфейс съвсем по същия начин, описан в предходната лекция като използваме **SceneBuilder**



## 9.1 Създаване на проекта

3. **Важна особеност в случая** е ,че редактираме изчистваме текстовото поле, идентифициращо Контролера в раздела **Controller** и избираме **Use fx:root construct**

4. Запазваме **FXML** файла в **SceneBuilder**



## 9.1 Създаване на проекта

- 5 Копираме, както преди, **пълния скелет** на Контролера, създаден в SceneBuilder, във файла на Контролера, наличен в NetBeans проекта
6. Проверяваме, че FXML файла с дървото с възли на графичния интерфейс действително не идентифицира съответния му Контролер т.е **корена** `<fx:root ...>` **не съдържа** елемента `fx:controller` (в противен случай, изтрийте този елемент). За да разгледате FXML файла кликнете върху него с десен бутон и изберете **Edit** от помощното меню
7. Напишете методите за обработка на събитията в Контролера(в този случай може да препишете тези методи от Контролера, използван в предходното JavaFX приложение със същия графичен интерфейс)

## 9.1 Създаване на проекта

8. Направете Java класа на Контролера да бъде произведен на компонентата в корена на дървото на възлите, която опакова останалите възли. В случая тази компонента е `AnchorPane`, а класа на Контролера е `FXMLDocumentController`. Така получаваме

```
public class FXMLDocumentController extends AnchorPane {  
  
    @FXML  
    private ResourceBundle resources;  
  
    @FXML  
    private URL location;  
  
    @FXML  
    private Label lblFirstNumber;
```

# 9.1 Създаване на проекта

## 9. Добавяме конструктор в Контролера

```
public FXMLDocumentController() {  
    FXMLLoader fxmlLoader = new FXMLLoader(  
        getClass().getResource("/coms/FXMLDocument.fxml"));  
  
    fxmlLoader.setRoot(this);  
    fxmlLoader.setController(this);  
  
    try {  
        fxmlLoader.load();  
    } catch (IOException exception) {  
        throw new RuntimeException(exception);  
    }  
}
```

където **coms** е пакетът, в който е сорс кода на FXML файла.  
**Видно е, че този конструктор ще се различава единствено по използвания FXML файл в други такива компоненти.**



## 9.1 Създаване на проекта

**10. Остава да редактираме класа на приложението по следния начин**

```
public class FXMLApplication extends Application {  
  
    @Override  
    public void start(Stage primaryStage) {  
    }  
  
    public static void main(String[] args) {  
        launch(args);  
    }  
}
```

**11. Изпълняваме Clean and Build на проекта, с което създаваме JAR файла на компонентата**

## 9.1 Създаване на проекта

В обобщение:

1. Създаваме Сцената (FXML file ) и съответния ѝ Контролер Java class, който наследява избрания контейнер за базов възел (root )
2. Редактираме Сцената (FXML file ) да използва `fx:root` за базов елемент. (без това не става)
3. Изтриваме `fx:controller` атрибута (без това не става)
4. **Добавяме конструктор в Контролера (без това не става)**
5. Компилираме и добавяме jar към Scene Builder

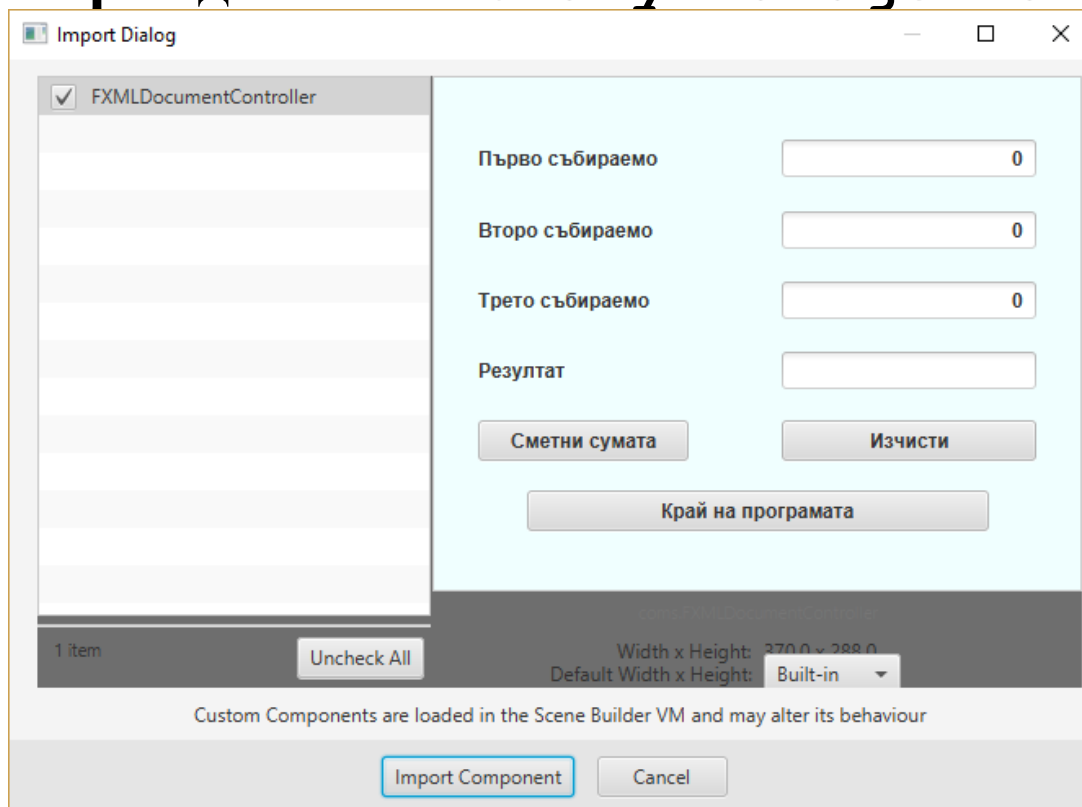
**Важно:**

**Задължително е версията на JavaFX да съвпада с версията на Scene Builder (Ако компилирате с JavaFX 11, то ползвайте Scene Builder 11)**

## 9.2 Добавяне на компонента

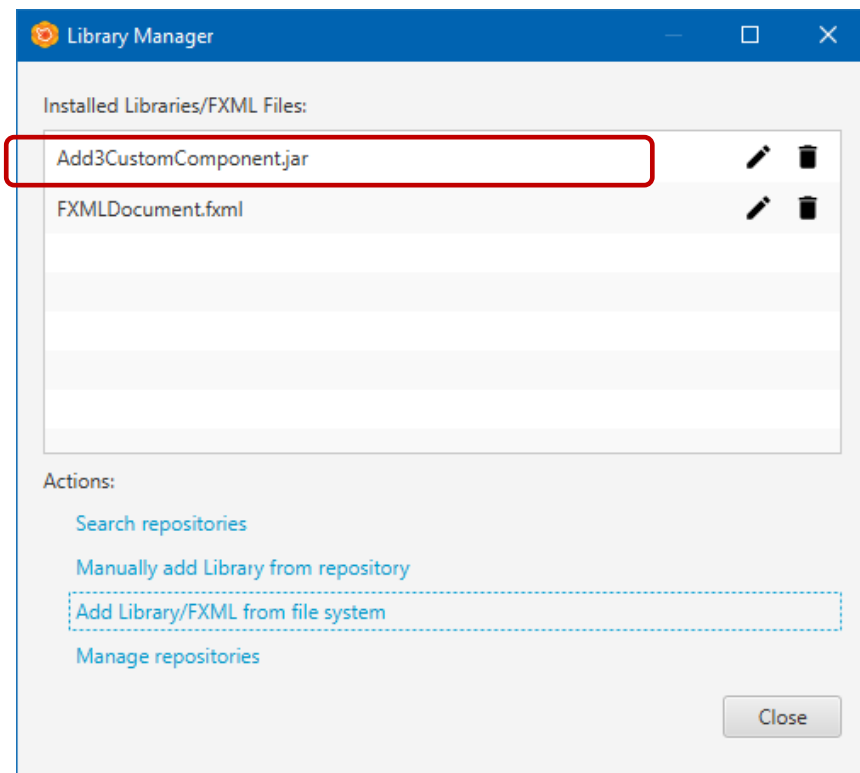
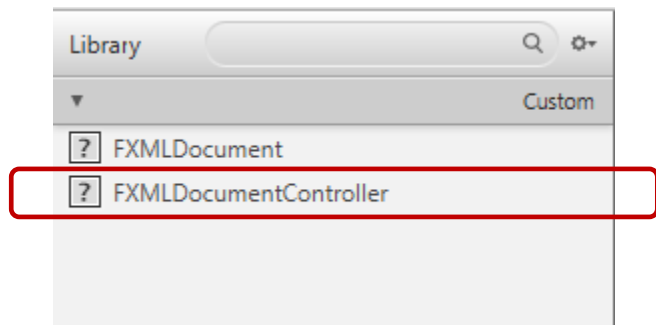
Следващата стъпка е да добавим JavaFX компонентата в SceneBuilder.

С помощта на Library Manager-а на SceneBuilder намираме JAR файла на компонентата във файловата система и я зареждаме в Library Manager-а



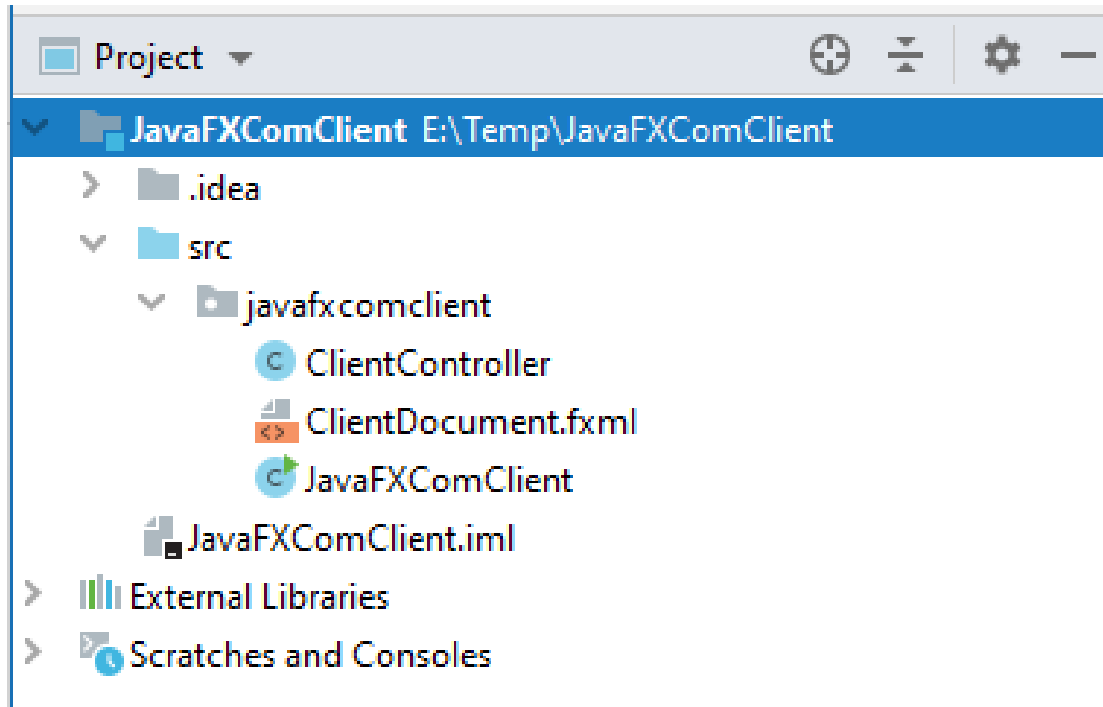
## 9.2 Добавяне на компонента

По този начин тази компонента става достъпна за изграждане на графичен интерфейс в раздела **Custom** на SceneBuilder по същия начин, както останалите стандартни компоненти на JavaFX. Името на компонентата в Library съвпада с името на Контролера в JAR файла



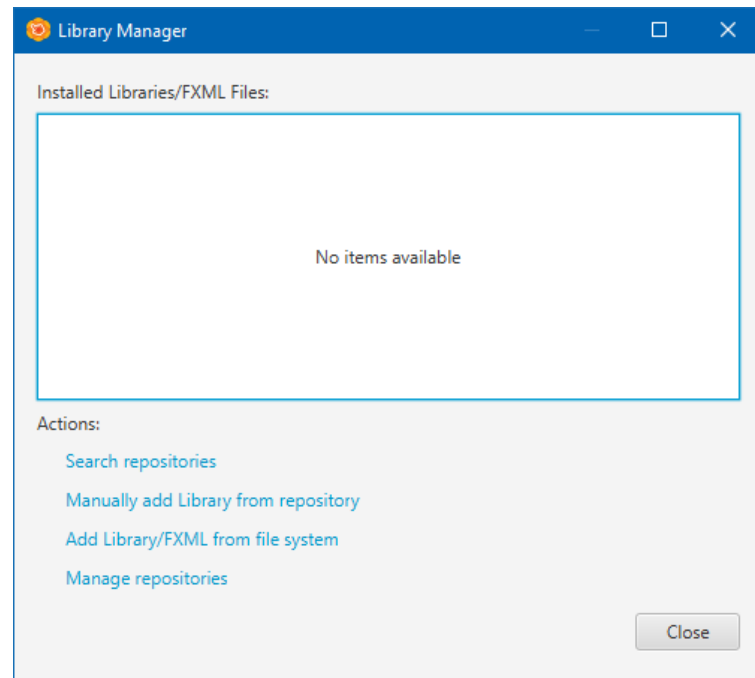
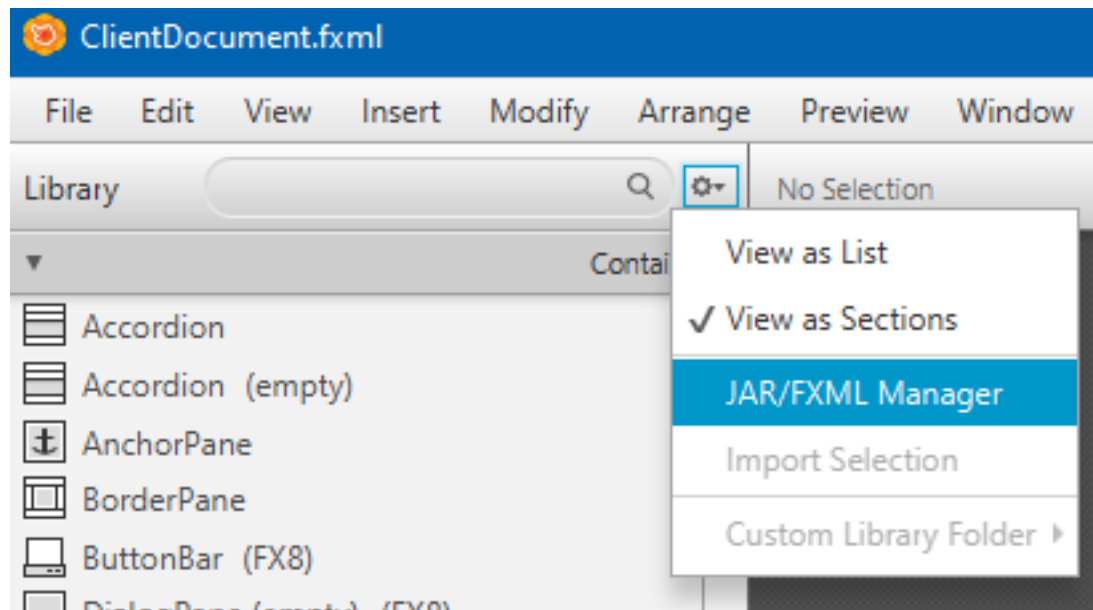
## 9.3 Използване на потребителска компонента

1. Създаваме ново Java FXML Application приложение на JavaFX, например, `JavaFXComClient`.
2. Отваряме `ClientDocument.fxml` в Scene Builder



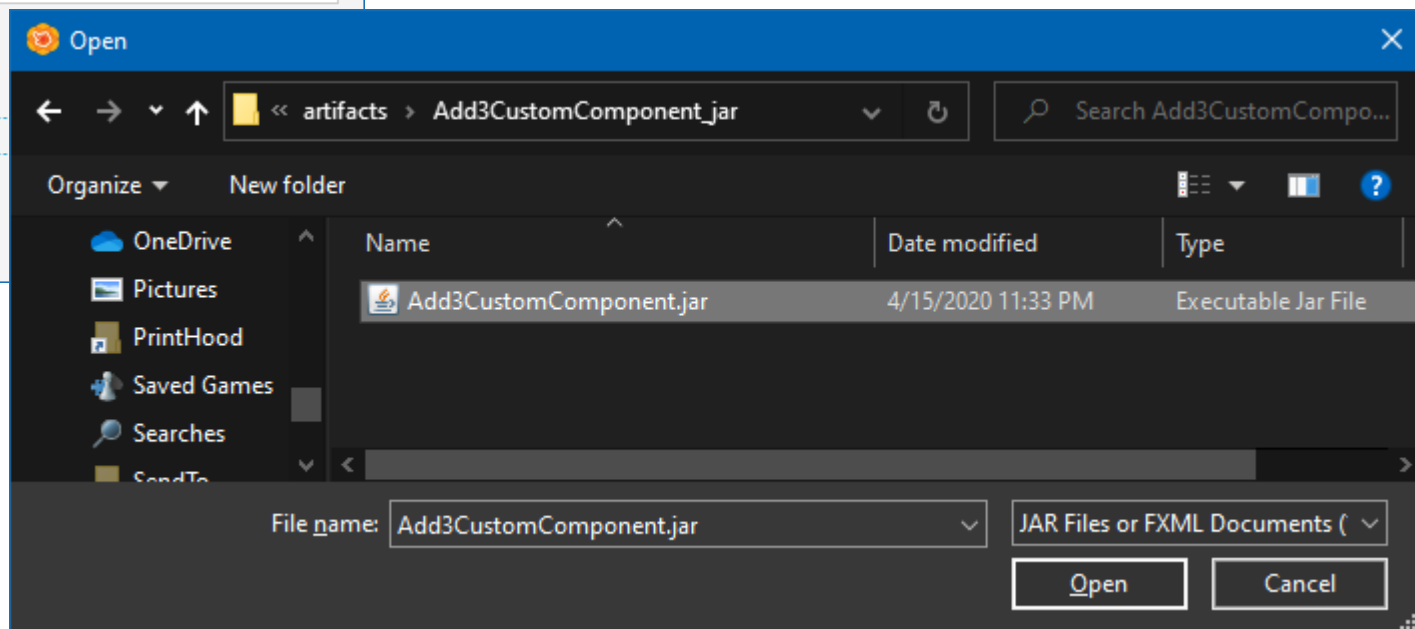
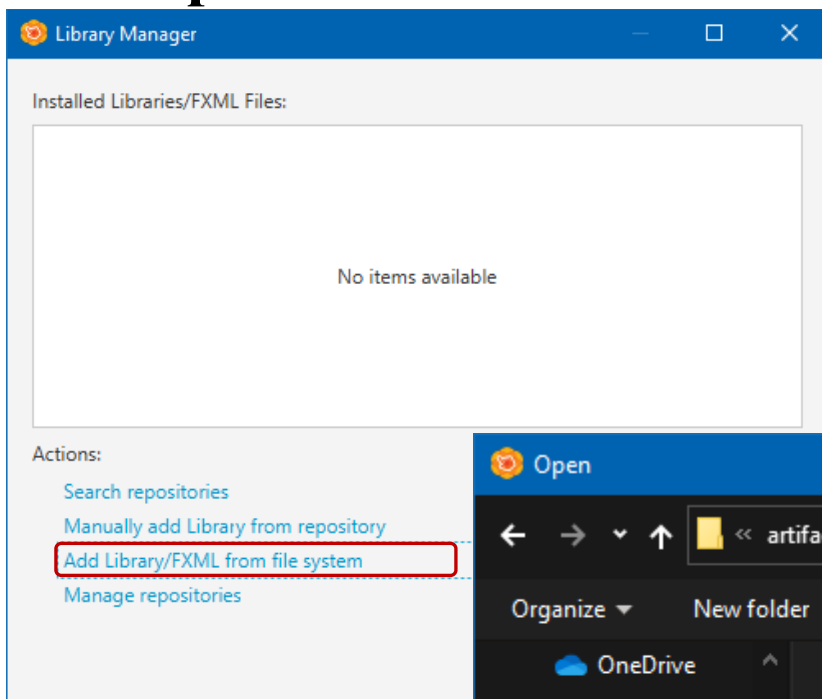
## 9.3 Използване на потребителска компонента

### 3. Избираме Jar/FXML Manager, за да стартираме Library Manager на Scene Builder



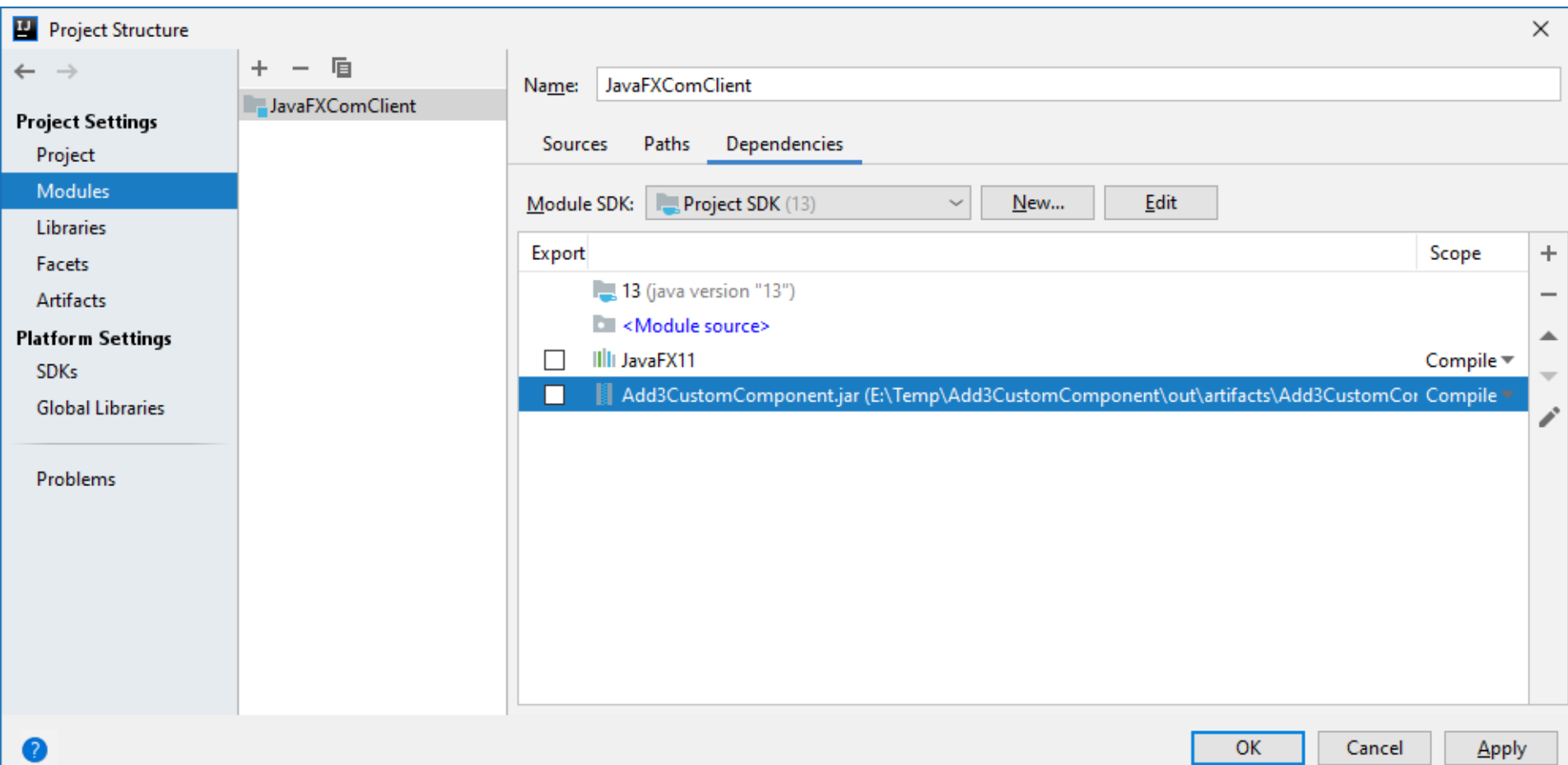
## 9.3 Използване на потребителска компонента

### 4. Добавяме към библиотеките на проекта jar файла на потребителската компонента (Project-> Properties)



## 9.3 Използване на потребителска компонента

5. Добавяме към библиотеките на проекта jar файла на потребителската компонента (**Modules>Dependencies**)

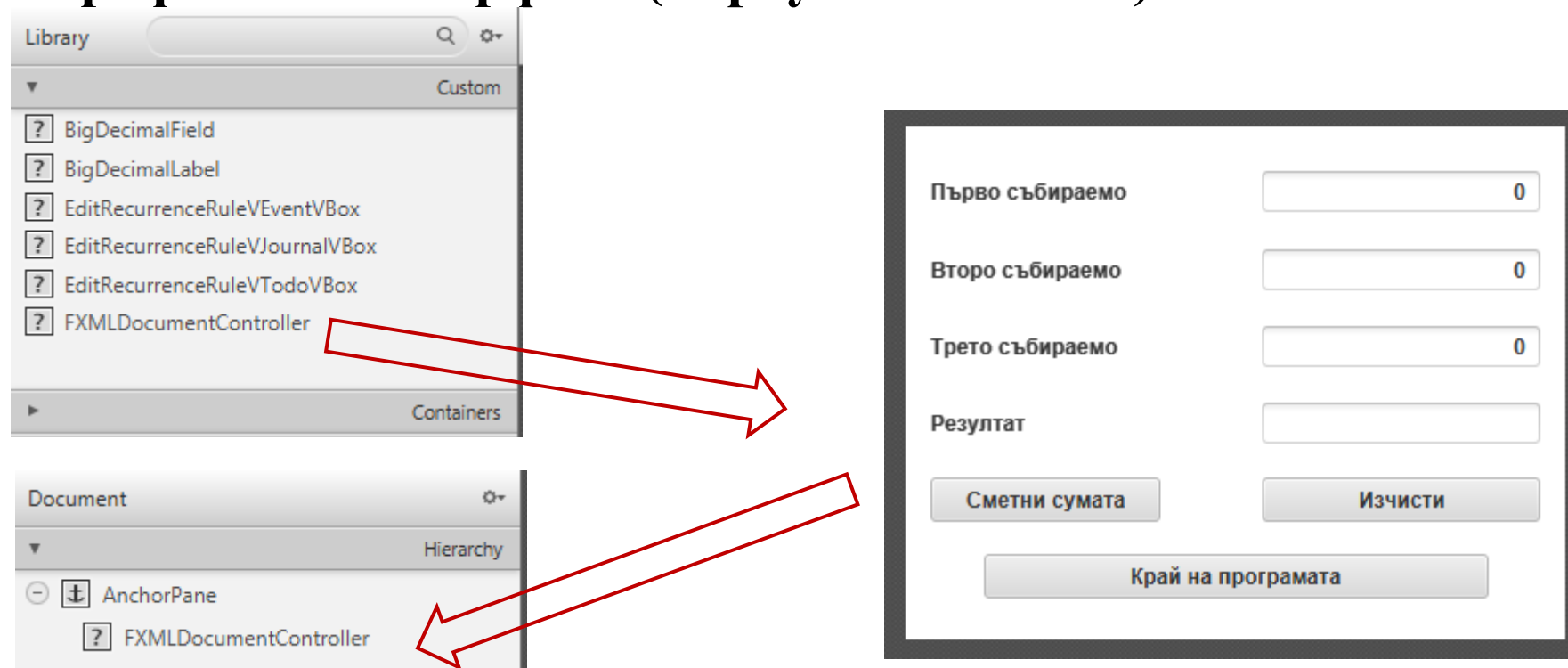




## 9.3 Използване на потребителска компонента

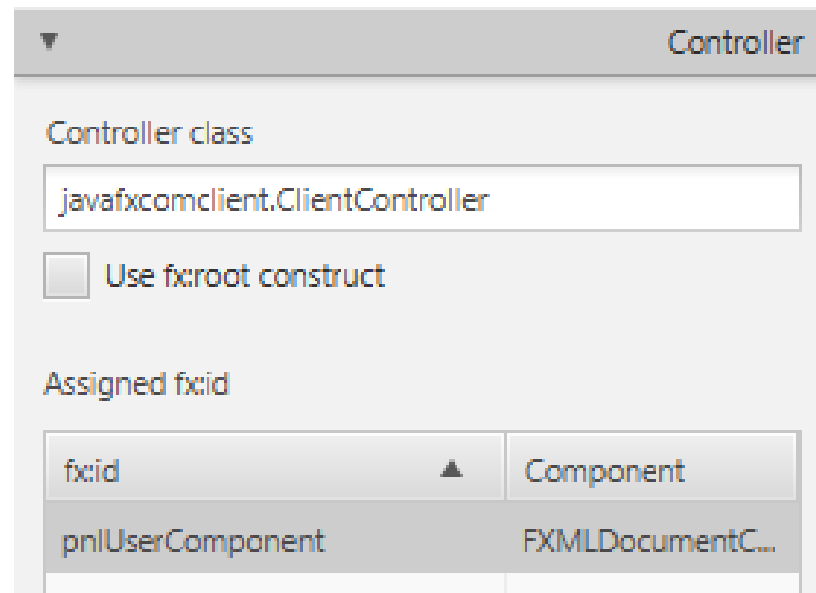
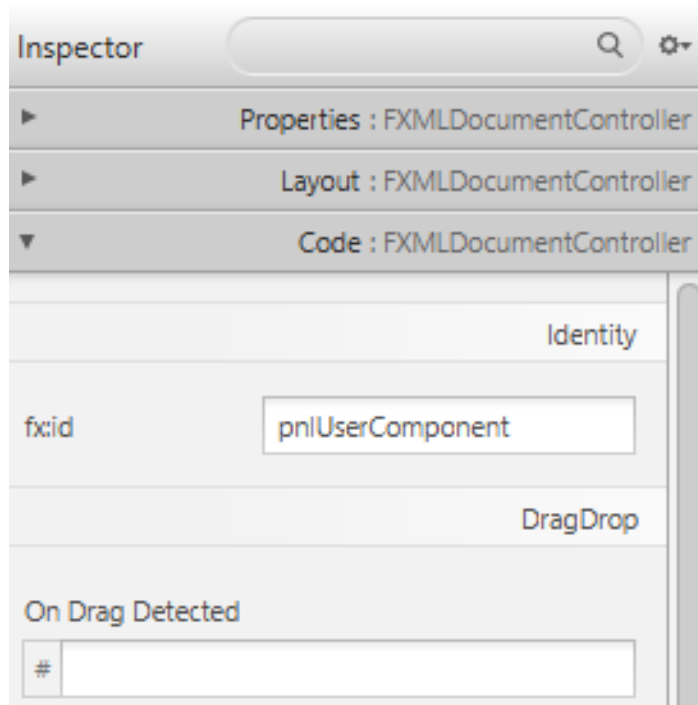
6. С двойно кликване върху FXML файла стартираме Scene Builder

7. Редактираме дървото на възлите на Сцената като добавяме („дърпаме“) потребителската компонента към графичния интерфейс (върху AnchorPane)



## 9.3 Използване на потребителска компонента

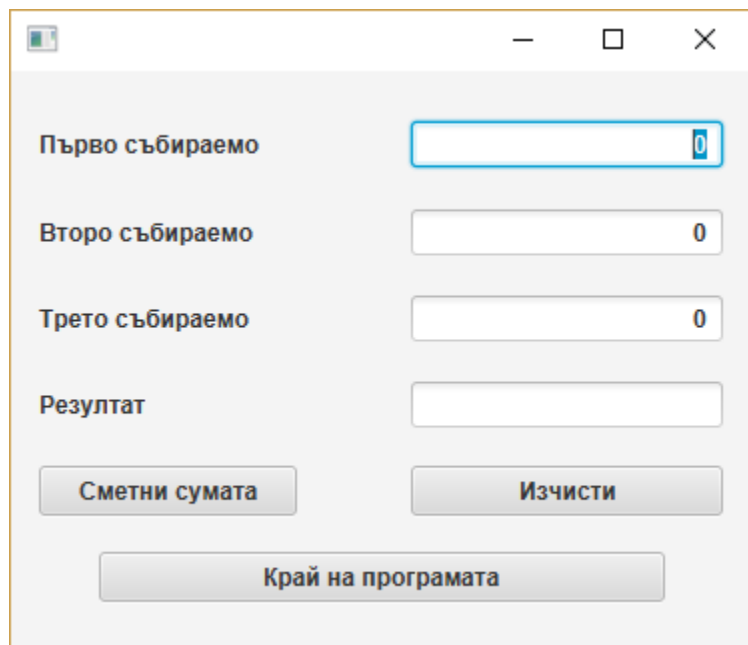
### 8. Въвеждаме `fx:id` идентификатор на потребителската компонента



## 9.3 Използване на потребителска компонента

9. Задаваме името на Контролера в секцията `Document->Controller class` в `SceneBuilder`. Това име трябва да съвпада точно с името на класа на Контролера в проекта на **IntelliJ**, предхождано от наименованието на пакета, където се намира този клас.
10. Запазваме `FXML` файла в `SceneBuilder` и копираме, както преди, **пълния скелет** на Контролера, създаден в `Scene Builder`, във файла на Контролера, наличен в **IntelliJ** проекта. Там коригираме с десен бутон и `Fix Imports` евентуални липсващи `import` дефиниции.
11. Изпълняваме проекта като стартираме `FXML` приложението с потребителската компонента

## 9.3 Използване на потребителска компонента



Първо събираемо 0

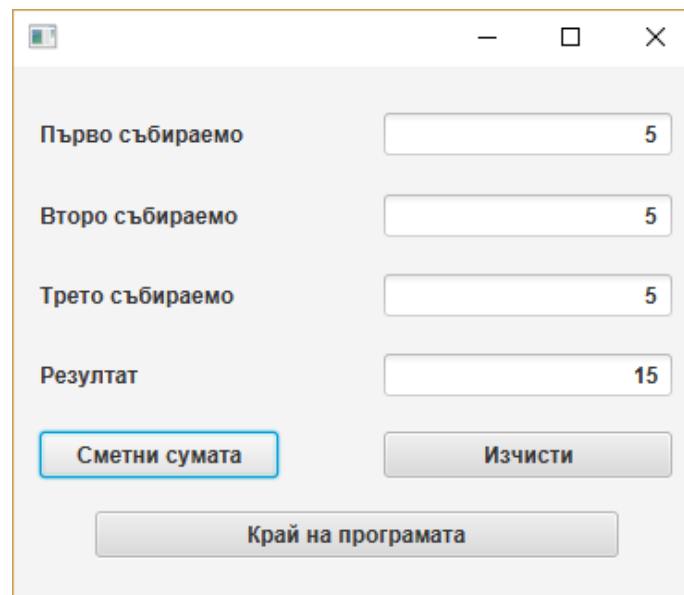
Второ събираемо 0

Трето събираемо 0

Резултат

Сметни сумата Изчисти

Край на програмата



Първо събираемо 5

Второ събираемо 5

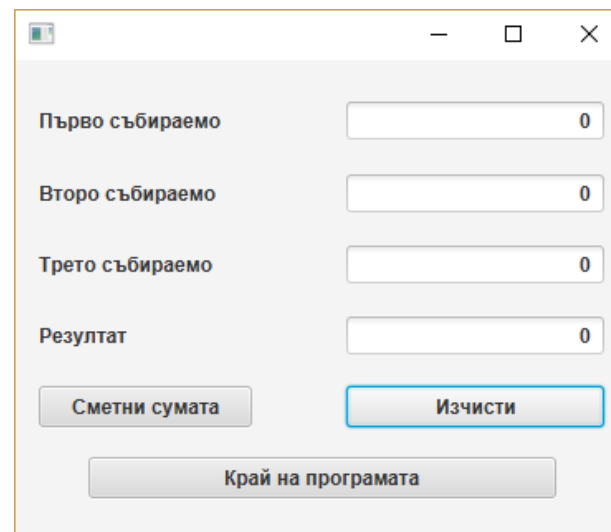
Трето събираемо 5

Резултат 15

Сметни сумата Изчисти

Край на програмата

Непосредствено се убеждаваме, че така полученото приложение притежава всички елементи от графичния интерфейс и функционалност, съдържащи се в потребителската компонента.



Първо събираемо 0

Второ събираемо 0

Трето събираемо 0

Резултат 0

Сметни сумата Изчисти

Край на програмата

# Задачи

1. Напишете JavaFXML графично приложение , което използва *JavaFX* компоненти за решаване на следната задача.
  - Нека с подходящ етикет и текстово поле се въвежда парична сума като число с плаваща запетая (представляваща ресто при покупка на дадена стока)
  - Нека има бутони *Calculate* и *Exit*. Бутонът *Exit* да прекратява работата на приложението.
  - Бутонът *Calculate* да пресмята броят видове монети, които трябва да се върнат като ресто. Рестото да се извежда в серия от етикет и текстово поле (не редатируеми) като приемаме, че разполагаме с монети по: 1, 2, 5, 10, 20 и 50 стотинки.
  - Например, ако се въведе ресто 23.32 то програмата трябва да разпредели (и изведе в съответните текстови полета) рестото като *23 лева, 0 монети по 50 стотинки, 1 монета от 20 стотинки, 1 монета от 10 стотинки, 0 монети от 5 стотинки, 1 монета от 2 стотинки и 0 монети от 1 стотинка*

# Задачи

2. **Напишете многократно използвана графична компонента, която позволява да се пресметне и изведе месечната вноска по сключен договор за кредит, при което се въвежда сумата на кредита, годишната лихва и брой години за изплащане на кредита. Демонстрирайте приложението на графичната компонента в JavaFXML приложение**
3. **Напишете многократно използвана графична компонента, която представя калкулатор. Демонстрирайте приложението на графичната компонента в JavaFXML приложение**