Ministerul Educaţiei al Republicii Moldova

Universitatea Tehnică a Moldovei

Catedra Tehnologii Informaționale

RAPORT

Lucrarea de laborator#3

la Medii Interactive de Dezvoltare a Produselor Soft

A efectuat: Profir Andrei

st.gr. TI – 143

A verificat: Cojanu Irina

lect.asist.

Chişinău 2016

Tema: Version Control Systems si modul de setare a unui server

Scopul lucrării:

- Realizeaza un simplu GUI Calculator
- Operatiile simple: +,-,*,/,putere,radical,InversareSemn(+/-),operatii cu numere zecimale.
- Divizare proiectului in doua module - Interfata grafica(Modul GUI) si Modulul de baza(Core Module).

Formularea conditiei problemei:

- *Basic Level* (nota 5 || 6):
    - Realizeaza un simplu GUI calculator care suporta functiile de baza: +, -, /, *.
- *Normal Level* (nota 7 || 8):
    - Realizeaza un simplu GUI calculator care suporta urmatoare functii: +, -, /, *, putere, radical, InversareSemn(+/-).
- *Advanced Level* (nota 9 || 10):
    - Realizeaza un simplu GUI calculator care suporta urmatoare functii: +, -, /, *, putere, radical, InversareSemn(+/-), operatii cu numere zecimale.
    - Divizare proiectului in doua module - Interfata grafica(Modul GUI) si Modulul de baza(Core Module).

Implementare task-uri:

- IDE: Intellij Idea 15.0.4, JavaFX Scene Builder Tool
- Limbajul: Java 8
- Tehnologii: JavaFX(Pattern MVC)

In acest laborator a fost folosit Patternul MVC ceea ce presupune diviziunea proectului in 3 parti. Aceste 3 parti fiind: Model – sau mai simplu spus datele si lucrul cu ele, Controller – ceea ce alcatuieste busines logica aplicatiei si in final dar nu mai putin important View – vizualizarea ceea ce vedem cu ochii si ceea cu ce interactioneaza nemijlocit utilizatorul aplicatiei asa ca butoane, liste, radio butoane si altele.

Pentru a crea partea vizuala a aplicatiei a fost folosit instrumentul Scene Builder. Acest instrument ne permite rapid si usor sa creem interfata grafica a aplicatiei. Toate elementele create se salveaza intr-un fisier ".fxml". Acest fisier reprezinta un fisier text care este asemanator "xml" fisiere sau "html".

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.text.*?>
<?import javafx.scene.control.*?>
<?import javafx.geometry.*?>
<?import javafx.scene.shape.*?>
<?import java.lang.*?>
<?import javafx.scene.layout.*?>

<VBox fx:id="wtf" alignment="CENTER" focusTraversable="true" onKeyPressed="#keyPressed" onKeyRelea
   <children>
      <StackPane maxHeight="-Infinity" maxWidth="-Infinity" prefHeight="115.0" prefWidth="400.0" s
      <HBox alignment="CENTER" prefHeight="290.0" prefWidth="400.0">
         <children>
            <VBox prefHeight="290.0" prefWidth="73.0" spacing="5.0"...>
            <GridPane fx:id="grid" hgap="5.0" prefHeight="290.0" prefWidth="307.0" vgap="5.0"...>
         </children>
      </HBox>
   </children>
   <padding>
      <Insets bottom="15.0" left="15.0" right="15.0" top="15.0" />
   </padding>
</VBox>
```

O secventa a fisierului “fxml”.

Dupa ce am creat interfata grafica in Scene Builder si structura aceste interfete a fost salvata in fisier, trebuie acest fisier sal conectam la proectul nostru intr-o clasa care este creata prin extinderea clasei “Application”. In baza acestei clase sunt create toate aplicatiile JavaFx. La crearea clasei noastre cind exindem clasa “Application” trebuie sa redefinim metoda “start()” altfel programul va genera eroare si nu va rula.

```
package calculator;

import ...

public class Main extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception{
        Parent root = FXMLLoader.load(getClass().getResource("sample.fxml"));

        root.setCursor(new ImageCursor(new Image("file:resources/images/cursor.png")));
        primaryStage.setTitle("Calculator");
        primaryStage.setResizable(false);
        primaryStage.getIcons().add(new Image("file:resources/images/icon.png"));
        Scene scene = new Scene(root);
        primaryStage.setScene(scene);
        primaryStage.show();
        scene.getRoot().requestFocus();
    }

    public static void main(String[] args) { launch(args); }
}
```

Fisierul “Main.java”

A doua clasa creata a fost “Model”, aceasta clasa raspunde de operatii cu numere intregi sau cu virgula. In aceasta clasa are un singur cimp de tip “BigDecimal” aceasta clasa de numere permite efectuarea operatiilor cu o precizie foarte mare, si pastrarea numerelor fara rotungire.

Clasa “Model” are 4 metode, 2 metode se refera la cimpul privat declarat in aceasta clasa. O metoda este “setter” care permite sa setam valoarea cimpului privat, iar alta metoda este “getter” care permite returnarea referintei catre cimpul privat.

Celelalte doua metode raspund nemarginit de efectuarea operatiilor, fiecare metoda primeste cite 2 parametri de tip “String”, unul din parametri este operatia care urmeaza sa fie efectuata si cel dea-l doilea parametru este un numar cu care se va face operatia, acest numar este pastrat intr-un sir de caractere pentru a nu se pierde precizia in primul rind si in al doilea rind la creare un numar “BigDecimal” este mai bine de initializat cu un sir de caractere.

Metodele au fost numite “calculations” si “additionalCalculations”. Prima metoda efectueaza 4 operatii de baza: +, -, *, /. Cea dea doua metoda efectueaza operatii adaugatoare si anume: radical, ridicarea la puterea a doua, negarea unui numar, procentul si reciproca. Fiecare din aceste metode intoarce ca rezultat un numar “BigDecimal”.

```
package calculator;

import java.math.BigDecimal;

/**
 * Created by Profir Andrei on 06.03.2016.
 */
public class Model {
    private BigDecimal mX;

    public BigDecimal calculations(String number, String operator) throws ArithmeticException {...}

    public BigDecimal additionalCalculations(String number, String operator) {...}

    public void setX(BigDecimal value) { mX = value; }

    public BigDecimal getX() { return mX; }
}
```

Structura fisierului “Model.java”

De busines logica a proiectului raspunde clasa “Controller”, care se afla in fisierul “Controller.java”. Aceasta clasa contine mai multe multe metode dar cele mai importante sunt metodele care prelucreaza mesajele sau actiunile care provin de la butoane sau taste.

Metoda “number” prelucreaza evenimentele cind este efectuat click pe unul din butonul care reprezinta un numar sau punctul care permite introducerea numerelor zecimale.

Metodele “operatorProc si aditionalOperator” prelucreaza evenimentele provenite de la butoanele care reprezinta operatiile atit cele standarte +, -, *, / cit si radicalul, puterea etc.

Metoda “control” primeste evenimentele si le prelucreaza de la butoanele de control sau asa butoane ca “Clear”, “Clear Entered”, “Delete”.

In final 2 metode care primesc si prelucreaza evenimentele generate de tastatura, aceste metode au denumirea de “keyPressed” si “keyReleased”. Ele primesc evenimentulverifica ce tasta a fost apasata si indeplinesc o anumita actiune in baza informatiei primite.

La fel au fost create niste metode optionale care ajuta la afisarea numerelor in forma dorita si afisarea istoriei operatiilor.

```java
package calculator;

import ...


public class Controller {
    @FXML
    private Text outputNumber;
    @FXML
    private Text historyOperations;
    @FXML
    private GridPane grid;

    private boolean isStart = true;
    private String mInput = "0";
    private String operator = "";
    private boolean isInput = true;
    private boolean isError = false;
    private boolean isReset = false;
    private boolean isAdditionalOperator = false;
    private String lastInHistory = "";
    private String history = "";

    private Button currentButton;

    private Model model = new Model();

    @FXML
    public void number(ActionEvent ae) {...}

    @FXML
    public void operatorProc(ActionEvent ae) {...}

    @FXML
    public void additionalOperator(ActionEvent ae) {...}
```

```java
    @FXML
    public void control(ActionEvent ae) {...}

    private void changeOperatorInHistory(String op) {...}

    private void outputHistory(String number, String operator) {...}

    private void showNumber(String output) {...}

    private void inputSymbol(String symbol) {...}

    private String processingInputNumber(String inputNumber) {...}
```

```
    private void resetAll() {...}

    private String formatNumberInHistory(String number) {...}

    private List<Button> keysPressed = new ArrayList<>();

    public void keyPressed(KeyEvent event) {...}
    public void keyReleased(KeyEvent event) {...}
}
```

Structura fisierului “Controller.java”

Pentru a modifica interfata grafica a fost folosit tabelul de stiluri “css”.

```
.root {
    -fx-background-color: #1D1D1D;
}

.textOutContainer {
    -fx-background-color: #0A640A;
}

.rectangle {
    -fx-fill: #2E2E2E;
}

.textOut {
    -fx-fill: #FFFFFF;
    -fx-font-family: "Segoe UI";
}

.textOutBig {
    -fx-font-size: 35px;
}
```

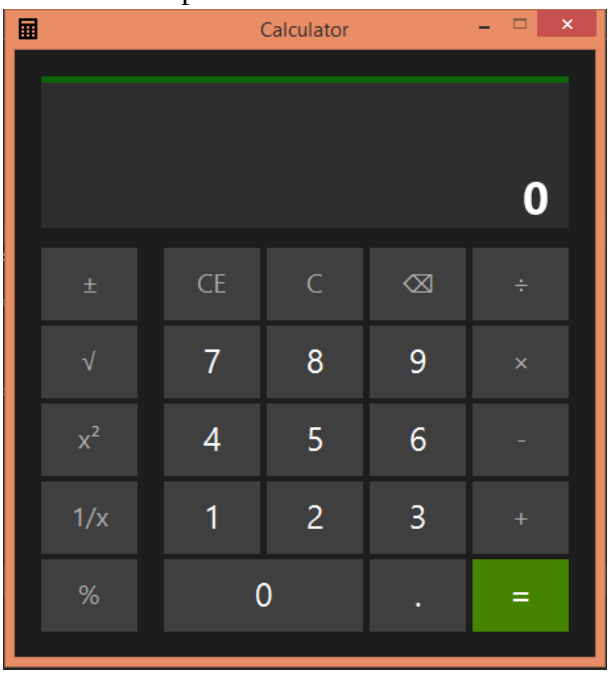
```
.textOutSmall {
    -fx-font-size: 15px;
    -fx-fill: #B2B2B2;
}

.button {
    -fx-focus-traversable: false;
    -fx-background-color: #3F3F3F;
    -fx-text-fill: #B2B2B2;
    -fx-border-radius: 0;
    -fx-background-radius: 0;
    -fx-font-size: 20px;
    -fx-font-family: "Segoe UI";
}

.button:hover {
    -fx-background-color: #585858;
}
```

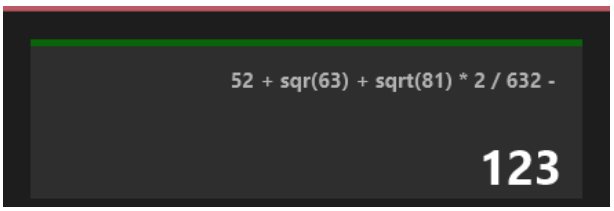
```
.button:pressed {
    -fx-background-color: #FFFFFF;
    -fx-text-fill: black;
}

.btnNumber {
    -fx-text-fill: #FFFFFF;
    -fx-font-size: 25px;
}

.btnEqual {
    -fx-background-color: #458400;
}

.btnEqual:hover {
    -fx-background-color: #5D9521;
}
```

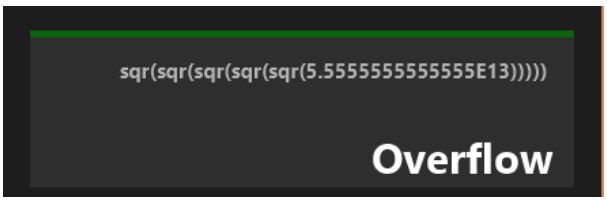
```
.btnEqual:pressed {
    -fx-background-color: #FFFFFF;
    -fx-text-fill: black;
}

.click {
    -fx-background-color: #FFFFFF;
    -fx-text-fill: black;
}
```

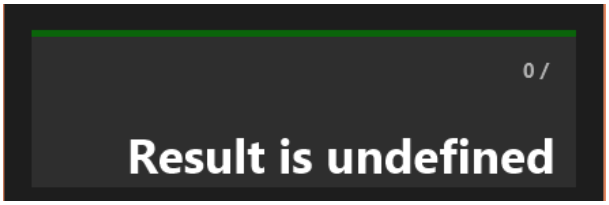
Fisierul “stylesheet.css” – reprezinta tabelul de stiluri al aplicatiei.

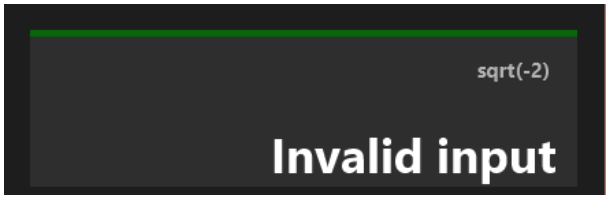
Screen-urile aplicatiei:

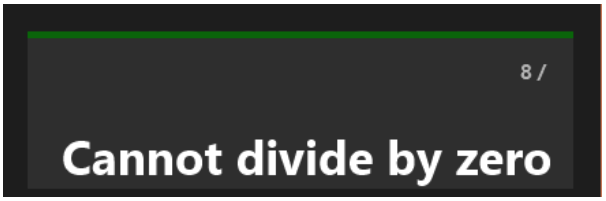












Concluzie:

In aceasta lucrare de laborator am capatat deprinderi practice in lucrul cu fisierele “fxml”, cu adaugarea stilurilor css la aplicatia noastra, ne-am invatat sa creem rapid si comod interfata grafica in Scene Builder. Am divizat proiectul in trei parti fiecare parte avind functiile si importanta proprie.

La finisarea laboratorului au fost atinse toate scopurile a fost realizat un calculator care permite efectuarea atit operatiilor de baza cu numere zecimale si intregi cit si a unor operatii adaugatoare.

Si datorita folosirii pattern-ului MVC sau ceva apropiat de acest pattern, aceasta a permis sa separam Modulul GUI si Modulul Core.