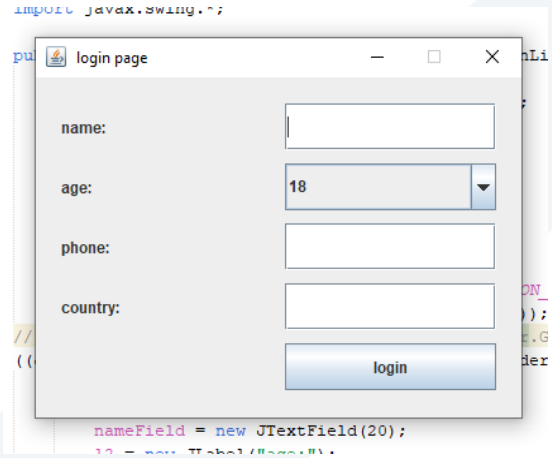
الجلسة السابعة والثامنة - برمجة 3

الغاية من الجلسة: مقدمة عن مفهوم واجهات المستخدم الرسومية GUI

Swing هي مكتبة GUI ضمن Java تسمح بإنشاء نوافذ ،أزرار ،حقول إدخال وغيرها

سنوضح مفهوم بناء واجهة مستخدم رسومية بسيطة باستخدام Swing من خلال مثال تطبيقي على نموذج تسجيل دخول بسيط.



سنقوم بانشاء project يحوي على كلاسين :

1. **كلاس يحوي على دالة التنفيذ الرئيسي main فقط لأجل الاستدعاء**

```
public class Test {

   public static void main(String[] args) {

      new Frame();

} }
```

2. **وكلاس Frame يحوي على كود انشاء الواجهة الرسومية**

أولاً يجب استيراد المكتبات وتضمينها ضمن Frame حتى استطيع استخدام الكلاسات والتوابع المتاحة ضمنها.

```
import javax.swing.*;

import java.awt.*;

import java.awt.event.*;
```

1 الوراثة من **JFrame لجعل الصف واجهة رسومية.**
2 تعيين عنوان النافذة في الشريط العلوي.
3 تحديد حجم النافذة (عرض و ارتفاع).
4 منع المستخدم من تغيير حجم النافذة.
5 إنهاء البرنامج عند إغلاق النافذة بإشارة ✖
6 لجعل النافذة تظهر في وسط الشاشة.
7 عرض النافذة فعليًا على الشاشة أي لجعلها مرئية، في الحالة الافتراضية تكون غير مرئية.

```
import javax.swing.*;
                    1
public class Frame extends JFrame {

public Frame() {

2   this.setTitle(" login page ");
3   this.setSize(400, 300);
4   this.setResizable(false);
5   this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
6   this.setLocationRelativeTo(null);
7   this.setVisible(true);

    } }
```

- **ملاحظة** : يجب دائمًا وضع هذه التعليمة setVisible(true) آخر الباني لأنها التعليمة التي تُظهر كل ما بُني في الواجهة و يجب أن تُنفذ بعد إعداد كل العناصر وترتيبها، وليس قبلها.

بعد انشاء الواجهة سنقوم بإنشاء العناصر من حقول دخل و ازرار ..، سنعرفها ك attribute ضمن الكلاس.

- JLabel : مكون (Component) يُستخدم لعرض نص داخل نافذة Swing ، ولا يمكن للمستخدم التفاعل معه، بل هو فقط للعرض.
- JTextField : حقل إدخال يستخدم لكتابة نص من طرف المستخدم في نافذة Swing مثلاً: إدخال اسم، بريد إلكتروني، رقم هاتف، ...
- JButton : هو زر يمكن للمستخدم الضغط عليه لتنفيذ إجراء معين (مثل إرسال النموذج، فتح صفحة جديدة، ...).
- JComboBox : يُستخدم لعرض قائمة خيارات يمكن للمستخدم اختيار واحدة منها فقط.

```
public class Frame extends JFrame {

 JLabel l1, l2, l3, l4;
JTextField nameField, phoneField, countryField;
JButton b1;
JComboBox<Integer> ageCombo;

public Frame() {
this.setTitle(" login page ");
this.setSize(400, 300);
this.setResizable(false);
this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
this.setLocationRelativeTo(null);
this.setVisible(true);
} }
```

```
public class Frame extends JFrame {

 JLabel l1, l2, l3, l4;
JTextField nameField, phoneField, countryField;
JButton b1;
JComboBox<Integer> ageCombo;

public Frame() {
this.setTitle(" login page ");
this.setSize(400, 300);
this.setResizable(false);
this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
this.setLocationRelativeTo(null);

l1 = new JLabel("name:");
nameField = new JTextField(20);
 l2 = new JLabel("age:");
ageCombo = new JComboBox<>();
for (int i = 18; i <= 60; i++) {
      ageCombo.addItem(i);
  }
l3 = new JLabel("phone:");
phoneField = new JTextField(20);
l4 = new JLabel("country:");
countryField = new JTextField(20);
b1 = new JButton("login");

this.setVisible(true);
} }
```

- إنشاء كل من حقول الإدخال والعرض والزر بالإضافة إلى القائمة المنسدلة ، وتهيئتها بالشكل المناسب

```
public class Frame extends JFrame {

 JLabel l1, l2, l3, l4;
JTextField nameField, phoneField, countryField;
JButton b1;
JComboBox<Integer> ageCombo;

public Frame() {
this.setTitle(" login page ");
this.setSize(400, 300);
this.setResizable(false);
this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
this.setLocationRelativeTo(null);
l1 = new JLabel("name:");
nameField = new JTextField(20);
 l2 = new JLabel("age:");
```

```
ageCombo = new JComboBox<>();
for (int i = 18; i <= 60; i++) {
      ageCombo.addItem(i);
  }
l3 = new JLabel("phone:");
phoneField = new JTextField(20);
l4 = new JLabel("country:");
countryField = new JTextField(20);
b1 = new JButton("login");

this.add(l1); this.add(nameField);
this.add(l2); this.add(ageCombo);
  this.add(l3); this.add(phoneField);
this.add(l4); this.add(countryField);
this.add(new JLabel()); // فراغ
this.add(b1);

this.setVisible(true);
} }
```

- إضافة المكونات إلى النافذة باستخدام this.add(...)
- يجب الانتباه ان ترتيب الإضافة يحدد ترتيب عرض العناصر في الواجهة الرسومية Java Swing

ملاحظة :

إذا قمنا بتنفيذ الكود دون استخدام نظام تخطيط (Layout Manager) مناسب: لن تظهر الواجهة بالشكل المطلوب. و ستُوضع الحقول والعناصر فوق بعضها البعض. و ستمتد المكونات لتملأ كامل النافذة بشكل غير منظم.

GridLayout هو نظام ترتيب (Layout Manager)
يُقسّم النافذة إلى شبكة (جدول) تتكون من:
عدد من الصفوف(rows)
عدد من الأعمدة(columns)
مسافة أفقية وعمودية
بحيث يتم توزيع كل عنصر في خانة واحدة من هذه الشبكة، بترتيب من اليسار إلى اليمين، ومن الأعلى إلى الأسفل.

```
public class Frame extends JFrame {
JLabel l1, l2, l3, l4;
JTextField nameField, phoneField, countryField;
JButton b1;
JComboBox<Integer> ageCombo;
public Frame() {
this.setTitle(" login page ");
this.setSize(400, 300);
this.setResizable(false);
this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
this.setLocationRelativeTo(null);

this.setLayout(new GridLayout(5, 2, 10, 10));

l1 = new JLabel("name:");
nameField = new JTextField(20);
 l2 = new JLabel("age:");
ageCombo = new JComboBox<>();
for (int i = 18; i <= 60; i++) {
      ageCombo.addItem(i);
  }
l3 = new JLabel("phone:");
phoneField = new JTextField(20);
l4 = new JLabel("country:");
countryField = new JTextField(20);
b1 = new JButton("login");
this.add(l1); this.add(nameField);
this.add(l2); this.add(ageCombo);
this.add(l3); this.add(phoneField);
this.add(l4); this.add(countryField);
this.add(new JLabel()); // فراغ
this.add(b1);
this.setVisible(true);
} }
```

ActionListenerهو واجهة(Interface) تُستخدم لمعالجة أحداث التفاعل مثل الضغط على زر . حيث أن الكلاس Frame أصبح يستطيع الاستماع ومعالجة الأحداث من نوع ضغط زر. عند الضغط على الزر b1، سينفّذ الكود الموجود في الدالة actionPerformed داخل هذا الكائن
(هذه هي الدالة التي سيتم استدعاؤها تلقائيًا عند الضغط على الزر.)

```
public class Frame extends JFrame implements
ActionListener {

 JLabel l1, l2, l3, l4;
JTextField nameField, phoneField, countryField;
JButton b1;
JComboBox<Integer> ageCombo;

public Frame() {
this.setTitle(" login page ");
this.setSize(400, 300);
this.setResizable(false);
this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
this.setLocationRelativeTo(null);
l1 = new JLabel("name:");
nameField = new JTextField(20);
 l2 = new JLabel("age:");
ageCombo = new JComboBox<>();
for (int i = 18; i <= 60; i++) {
      ageCombo.addItem(i);
  }
l3 = new JLabel("phone:");
phoneField = new JTextField(20);
l4 = new JLabel("country:");
countryField = new JTextField(20);
b1 = new JButton("login");

this.add(l1); this.add(nameField);
this.add(l2); this.add(ageCombo);
  this.add(l3); this.add(phoneField);
this.add(l4); this.add(countryField);
فراغthis.add(new JLabel()); //
this.add(b1);
 b1.addActionListener(this);
this.setVisible(true);
}
public void actionPerformed(ActionEvent arg0)
{
 String name = nameField.getText();
JOptionPane.showMessageDialog(this, "hello " + name + "!");
}}
```

في برمجة Java الموجهة للأحداث (Event-Driven Programming) ، تتمثل الأساسيات في مفاهيم الـ Source (المصدر) والـ Event (الحدث) والـ Listener (المستمع):

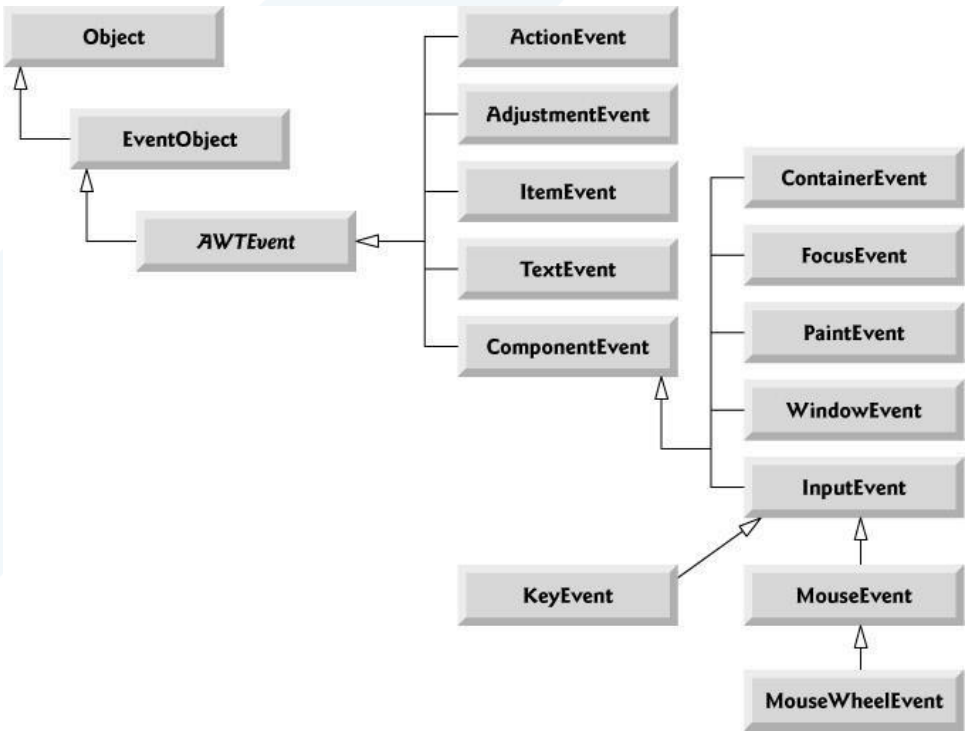
1. Source

- المصدر هو العنصر الذي يُثير الحدث عندما يحدث تفاعل معه، مثل زر الفأرة أو زر على واجهة المستخدم أو أي عنصر آخر يتفاعل مع المستخدم.
- عندما يحدث تفاعل مع المصدر، ينشئ الـ Source (المصدر) كائن ا من نوع الحدث المناسب ويُعلن الحدث.

2. Event

- الحدث هو النتيجة التي تحدث نتيجة لتفاعل المصدر، مثل النقر على زر الفأرة أو الضغط على زر على واجهة المستخدم.
- في Java، الحدث هو كائن من نوع معين يمثل الحدث الذي وقع، مثل ActionEvent أو MouseEvent.
- يحتوي الحدث عادة على معلومات إضافية حول الحدث الذي وقع، مثل الموقع (في حالة الفأرة) أو النص الذي تم كتابته (في حالة حدث النص).

ولدينا أنواع للحدث مثل الـ Mouse Event ويتضمن أي تفاعل يقوم به المستخدم مع الفأرة مثل النقر والتحريك و .. الخ، وكذلك الـ Key Event تنتج من التفاعل مع الـ keyboard والـ Event Action للنقر على الزر، أنواع أخرى من الـ Events واضحة في الشكل الآتي:

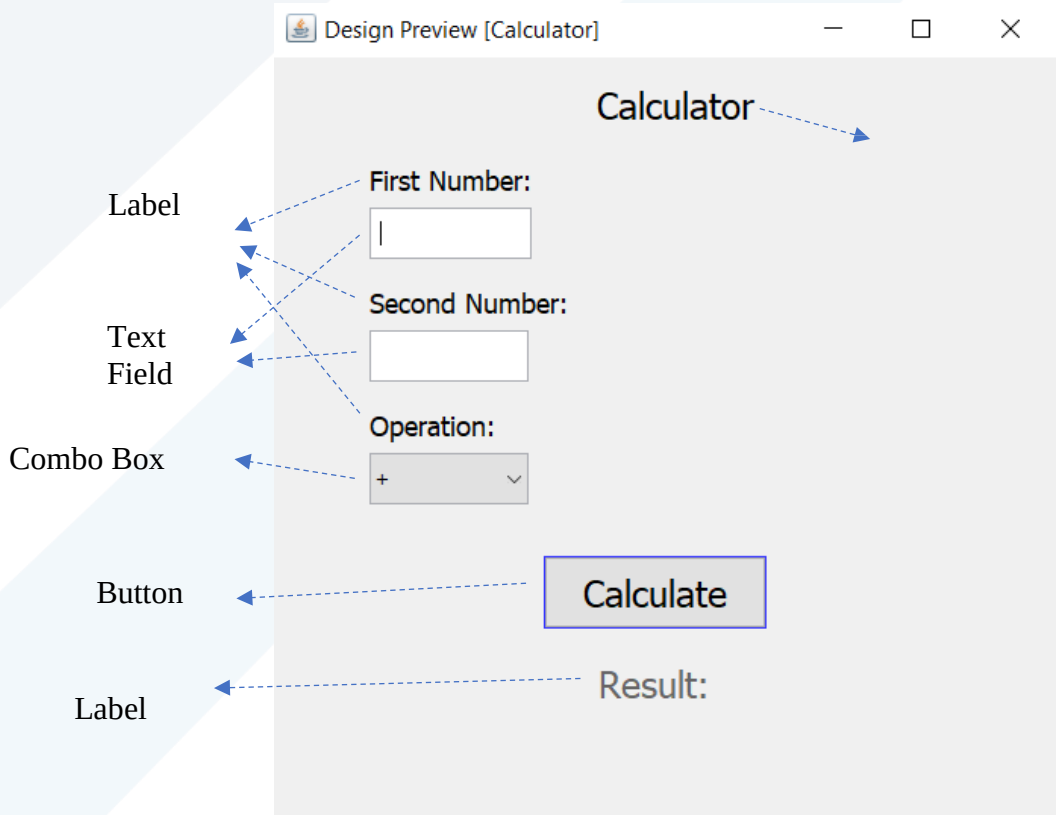


3. listener

- المستمع هو الكائن الذي يتم تعليقه على المصدر للاستماع إلى الأحداث التي يثيرها.
- في Java، يحقق المستمع واجهة معينة تحتوي على طريقة (أو أكثر) يتم استدعائها عند حدوث الحدث المرتبط بالمستمع.
- عاد ة ما يتم إنشاء مستمع مخصص لكل نوع من الأحداث التي يمكن أن يثيرها المصدر، على سبيل المثال ActionListener
للاستماع إلى حدث النقر على زر.

هذه الآلية تمكنك من بناء تطبيقات تفاعلية في Java باستخدام الـ Java Swing وهي مكتبة غنية تم ِكن المستخدمين من التفاعل مع العناصر على واجهة المستخدم ويستجيب التطبيق بنا ء على هذه الإدخالات

الآن نريد إنشاء واجهة بطريقة أخرى
نريد بناء تطبيق بسيط لآلة حاسبة بحيث تحتوي على حقل لإدخال العدد الأول وحقل لإدخال العدد الثاني وقائمة منسدلة يختار المستخدم منها العملية التي يريدها، وزر calculate، وعبارة من أجل الخرج، الشكل الآتي يوضح شكل الآلة الحاسبة:



ننشئ Java project ثم ننقر بالزر اليميني على الـ package ونختار JFrame Form. سوف نستخدم مجموعة من العناصر مثل الـ label والـ TextField والـ Button والـ ComboBox.

أين نكتب الكود الخاص بعملية الجمع؟

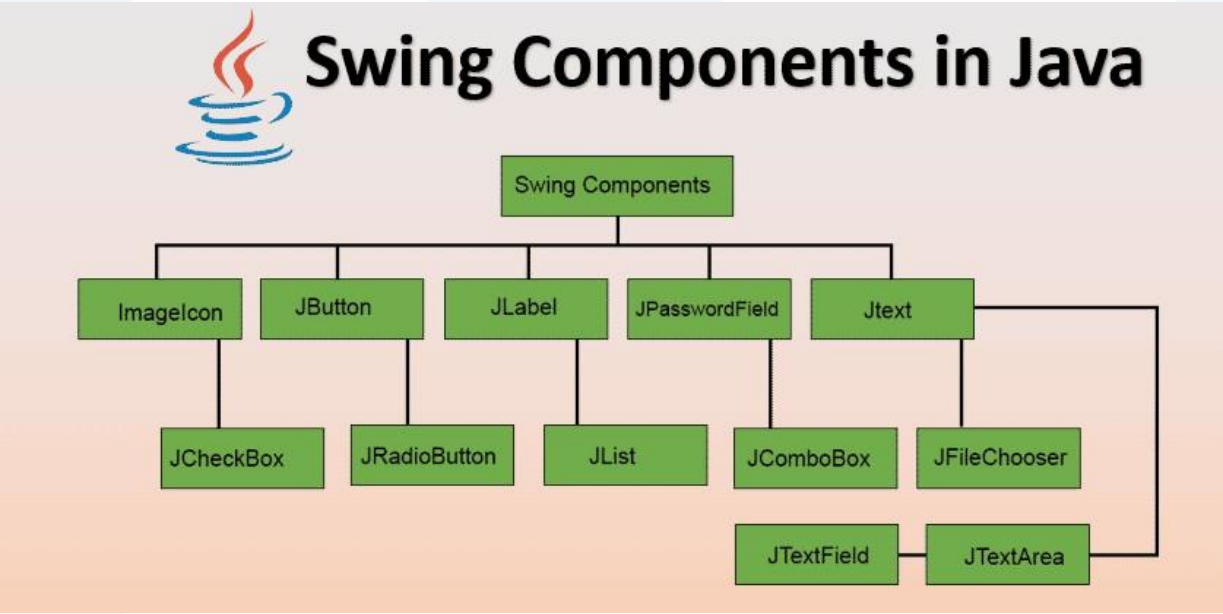
ننقر نقرتين على زر الـ Calculate فنذهب إلى الطريقة
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) :

الاسم jButton1ActionPerformed يعبر عن اسم الطريقة، وعادة ما يكون هو نفس اسم العنصر (component) الذي يثير الحدث (في هذه الحالة، الزر Calculate يُسمى jButton1) وبعد ذلك تأتي كلمة "ActionPerformed" لتشير إلى أن هذه الطريقة تتعامل مع حدث ينشأ عندما يتم تنفيذ إجراء (action) معين.

الوسيط (java.awt.event.ActionEvent evt) يحتوي على معلومات حول الحدث الذي حدث مثل المصدر وما إلى ذلك.

هذه الطريقة هي الـ Listener لأنه ضمنها تتم معالجة الحدث الذي هو النقر على زر الـ Calculate. لاحظ أن الوسيط الممرر للطريقة السابقة هو ActionEvent (الذي هو النقر على الزر).

فيما يلي مجموعة من العناصر في Java Swing:



العناصر السابقة هي الـ Sources والتي عندما نتفاعل معها تنقدح أنواع مختلفة من الأحداث Events والتي تنصت إليها الـ Listeners. في البداية يجب أن نعطي اسم (متحول) لكل عنصر في النافذة من

خلال النقر بالزر اليميني على العنصر ثم change variable name
ونعطي كل منها اسم معين مثلا نسمي الحقل الأول x والثاني y وقائمة العمليات الاسم op وعبارة النتيجة الاسم res، ثم نكتب في الطريقة السابقة الكود الآتي:

```
double xx = Double.parseDouble(x.getText());
double yy = Double.parseDouble(y.getText());

String selectedOperator = (String) op.getSelectedItem();
switch (selectedOperator) {
    case "+": res.setText("Result: "+ (xx+yy)); break;
    case "-": res.setText("Result: "+ (xx-yy)); break;
    case "*": res.setText("Result: "+ (xx*yy)); break;
    case "/": res.setText("Result: "+ (xx/yy)); break;
}
```

حيث أن xx هي من أجل تحويل القيمة في الحقل الأول (حقل إدخال العدد الأول) من String إلى double ونفس الأمر بالنسبة للـ yy.

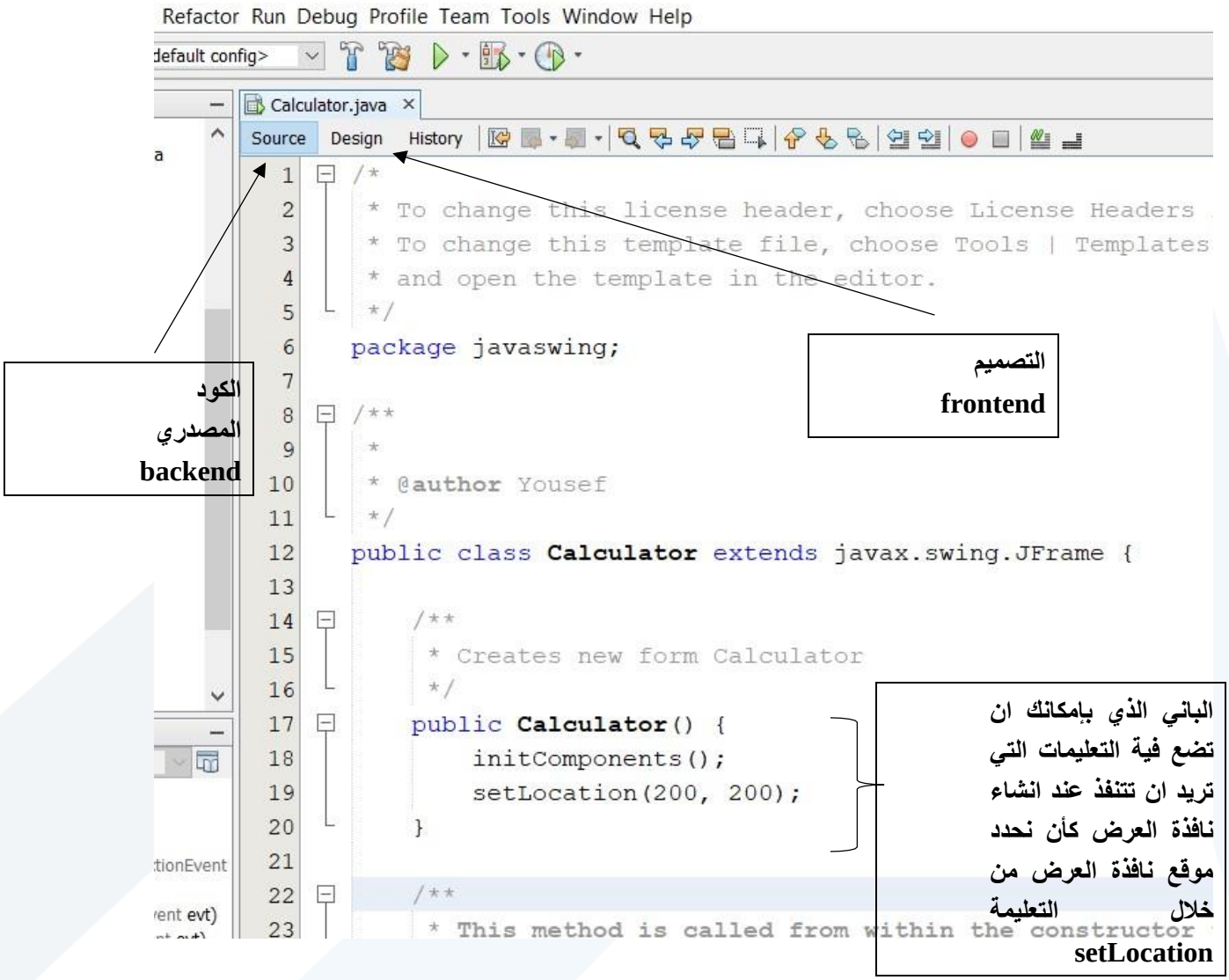
حيث أن parseDouble تحول الـ String المدخل (الوسيط) إلى double.

المتحول selectedOperator هو للحصول على العملية التي تم تحديدها (+ أو – أو * أو /) إلى String.

ثم في عبارة switch نختبر، فإذا كانت العملية هي "+" بالتالي أسند إلى متحول res (متحول الخرج) كلمة result متبوعة بجمع القيمتين المدخلتين xx+yy، ونفس الأمر بالنسبة لباقي العمليات.

من أجل اختبار البرنامج ننقر على run أو بالزر اليميني نختار run File ونختبر الآلة الحاسبة.

لا تنس: أنت تتعامل مع قسم يعبر عن الواجهة أو الـ Frontend وقسم يتعامل مع الكود في الكواليس أو Backend.



أيضاً من المهم أن تعرف أن جميع العناصر التي تعرفها في الواجهة من أزرار وحقول إدخال وقوائم وغيرها هي في الحقيقة أغراض من صفوف يتم تعريفها في الـ Backend بشكل تلقائي بمجرد أن تضعها في الواجهة Frontend:

```
210
211         // Variables declaration - do not modify
212         private javax.swing.JButton jButton1;
213         private javax.swing.JLabel jLabel1;
214         private javax.swing.JLabel jLabel2;
215         private javax.swing.JLabel jLabel3;
216         private javax.swing.JLabel jLabel4;
217         private javax.swing.JComboBox<String> op;
218         private javax.swing.JLabel res;
219         private javax.swing.JTextField x;
220         private javax.swing.JTextField y;
221         // End of variables declaration
222     }
```

أما باقي الطرق الموجودة في الـ Backend هي طرق معالجة الأحداث Listeners لمعالجة جميع التفاعلات مع العناصر التي وضعتها في الواجهة من أزرار وحقول إدخال وقوائم وغيرها، بالإضافة إلى طريقة الـ main التي يبدأ التنفيذ منها حيث يتم فيها أخذ غرض من الواجهة التي عرفتها التي هي Calculator وجعلها مرئية setVisible(true)

```
/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    Look and feel setting code (optional)

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new Calculator().setVisible(true);
        }
    });
}
```