بسم اللّه الرحمن الرحیم

آموزشگاه تحلیل داده

تخصصی ترین مرکز برنامه نویسی و دیتابیس در ایران

دکمه های ضرب، تفریق و تقسیم در ماشین حساب

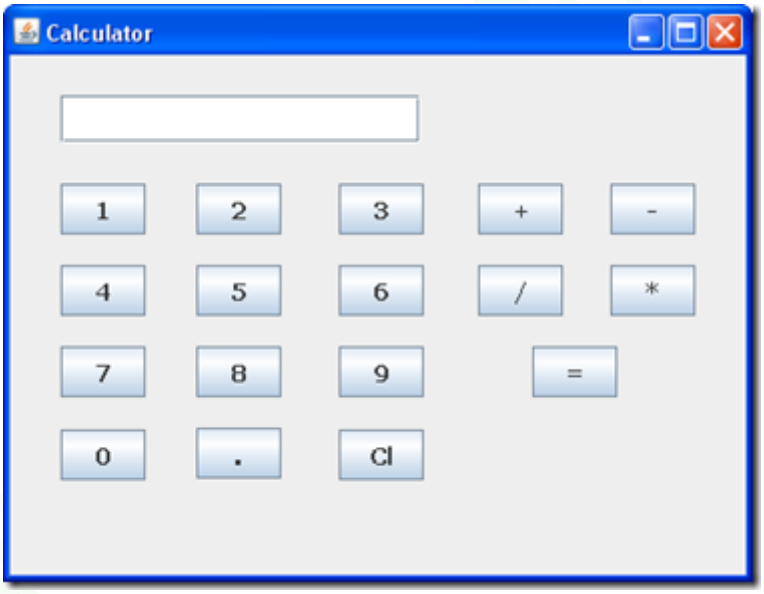
مدرس: مهندس افشین رفوآ

آدرس آموزشگاه. تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 - واحد 7
88146323 - 88446780 - 88146330

دکمه های ضرب، تفریق و تقسیم در ماشین حساب

اکنون که دکمه ی جمع در ماشین حساب جاوا کار می کند، می توانیم دکمه های تفریق، تقسیم و ضرب را اضافه کنیم. درست مانند دکمه ی جمع، این دکمه ها نیز کاری انجام نمی دهند: هنوز دکمه ی تساوی تمام کارها را انجام خواهد داد. تنها کاری که اپراتورهای دکمه ها انجام می دهند، ثبت دکمه ای است که کلیک می شود: جمع، تقسیم، تفریق و یا ضرب.

اولین کاری که باید انجام دهید، قرار دادن چند دکمه در فرم می باشد. در تصویر زیر، دکمه ی Clear را جابه جا کرده ایم و تمام اپراتورهای مربوط به دکمه ها را در سمت راست panel قرار می دهیم. به راحتی می توانید طرح خود را داشته باشید.



زمانی که دکمه های جدید را اضافه کرده اید، متغیرهای پیش فرض را به btnSubtract، btnDivide و btnMultiply تغییر نام دهید. (روش دیگری برای تغییر نام متغیر کلیک راست کردن روی دکمه می باشد. سپس منویی ظاهر خواهد شد. "Change Variable Name" را انتخاب کنید.)

تکنیکی که برای دریافت دکمه ی کلیک شده استفاده خواهیم کرد، ذخیره کردن متن دکمه در field variable می باشد. سپس می توانیم از یک عبارت switch برای امتحان کردن کاراکتر در field variable استفاده کنیم.

اگر این کاراکتر علامت + باشد، می توانیم عمل جمع را انجام دهیم. اگر علامت ؟ باشد می توانیم تفریق انجام دهیم و اگر علامت / باشد، تقسیم خواهیم کرد و اگر علامت * باشد، ضرب خواهیم کرد.

اکنون روی دکمه ی Source کلیک کنید تا به کد خود بازگردید. متغیر فیلد زیر را در بالا و درست زیر دو متغیر دیگر اضافه کنید.

```
private char math_operator;
```

بالای کد شما باید مانند تصویر زیر باشد.

```
public class JavaCalculator extends javax.swing.JFrame {

    private double total1 = 0.0;
    private double total2 = 0.0;
    private char math_operator;

    public JavaCalculator() {
        initComponents();
    }
```

برای دریافت کاراکتر روی دکمه ای که کلیک شده بود، می توانیم متودی را تنظیم کنیم. متود زیر را به کد خود اضافه کنید.

```
private void getOperator(String btnText) {
  math_operator = btnText.charAt(0);
  total1 = total1 + Double.parseDouble(txtDisplay.getText());
  txtDisplay.setText("");
}
```

تا زمانیکه متود بالا بین پرانتزهای Class و نه بین کروشه های متود دیگر، می باشد، می توانید آن را در هر جایی در کد خود قرار دهید.

ما متود getOperator را فرا خوانده ایم. این یک متود void می باشد، بنابراین هیچ مقداری گزارش نمی دهد– این متود تنها در اجرای کد همراهی می کند. بین پرانتزهای تیتر متود یک متغیر String به نام btnText داریم. این متغیر مشخصا متن مربوط به دکمه ای است که کلیک شده است.

متن مربوط به دکمه یک رشته می باشد. به هرحال عبارت های switch در جاوا نمی توانند رشته ها را کنترل کنند، بنابراین لازم است که رشته را به یک کاراکتر تغییر دهیم. این کار را می توان به وسیله ی خط زیر انجام داد.

```
math_operator = btnText.charAt(0);
```

متود charAt یک کاراکتر از یک رشته را دریافت می کند. کاراکتر مورد نظر شما بین پرانتزهای charAt قرار می گیرد. نماد ریاضی دکمه های ما در رشته همیشه در کاراکتر 0 قرار می گیرد. سپس در فیلد متغیر char که در بالای کد تنظیم کرده ایم، ذخیره می شود.

به دو خط دیگر در کد دقت کنید. این دو خط دقیقا مشابه خطوط دکمه ی به علاوه هستند و دقیقا همان کار را نیز انجام می دهند – ذخیره سازی اولین عدد در متغیری به نام total1. اپراتور هر دکمه لازم است که این کار را انجام دهد، بنابراین وجود این دو خط در متود به جای کد مربوط به اپراتور دکمه، معناددار می باشد.

بنابراین کد btnPlus را جایگذاری کرده و دو خط زیر را از آن حذف کنید.

```
total1 = total1 + Double.parseDouble(txtDisplay.getText( ) );
        txtDisplay.setText("");
```

این دو خط را جایگزین آنها نمایید.

```
String button_text = btnPlus.getText();
        getOperator(button_text);
```

دو خط اول متن را از دکمه ی به علاوه دریافت کرده و آن را در یک رشته متغیر ذخیره می سازند. این کار به متود getOperator نیز واگذار می شود.

همان دو خط می توانند به اپراتور دیگر دکمه ها اضافه شوند و تنها نام دکمه را تغییر دهند.

به ویو design بازگردید و روی دکمه ی منها دابل کلیک کنید. برای code stub خطوط زیر را اضافه کنید.

```
String button_text = btnMinus.getText();
        getOperator(button_text);
```

(گرچه برای متغیر String از همان نام استفاده کرده ایم، به هرحال جاوا گیج نمی شود، چرا که متن هر دکمه ای برای کد مخصوص دکمه ی خود داخلی می باشد)

روی دکمه ی تقسیم خود دابل کلیک کرده و خطوط زیر را اضافه کنید.

```
String button_text = btnDivide.getText();
        getOperator(button_text);
```

در اینجا نیز کد مربوط به دکمه ی ضرب را مشاهده می کنید.

```
String button_text = btnMultiply.getText();
        getOperator(button_text);
```

اکنون که کد مربوط به اپراتور هر چهار دکمه را داریم، می توانیم با دکمه ی تساوی تطبیق دهیم.

برای دکمه ی تساوی می توان یک عبارت switch تنظیم کرد تا درک کنیم که متغیر math_operator چیست.

```
switch ( math_operator ) {
        case '+':
        break;
        case '-':
        break;
        case '/':
        break;
        case '*':
        break;
        }
```

عبارت switch برای هر کدام از اپراتورهای ریاضی یک مورد دارد: +، -، /، و *. هنوز هیچ کدی اضافه نکرده ایم. اما به کد مربوط به دکمه ی تساوی، نگاهی داشته باشید.

```
total2 = total1 + Double.parseDouble( txtDisplay.getText( ) ) ;
        txtDisplay.setText( Double.toString(total2) );
        total1 = 0;
```

دو خط آخر نیازی به تغییر ندارند. به هرحال خط اول می تواند در عبارت switch استفاده شود. به یاد داشته باشید که این خط، خطی است که جمع می کند.

این خط می تواند به عنوان کد مربوط به + استفاده شود.

```
case '+':
        total2 = total1 + Double.parseDouble(txtDisplay.getText(
) );
        break;
```

اگر دکمه ی منها کلیک شده باشد، می توان به سادگی در خط بالا جمع را به منها تغییر داد.

```
case '-':
        total2 = total1 - Double.parseDouble(txtDisplay.getText(
) );
        break;
```

کد مربوط به تقسیم نیز به شکل زیر می باشد.

```
case '/':
        total2 = total1 / Double.parseDouble(txtDisplay.getText(
) );
        break;
```

کد مربوط به کاراکتر ضرب هم مانند زیر است.

```
case '*':
        total2 = total1 * Double.parseDouble(txtDisplay.getText(
) );
        break;
```

در اینجا تمام کد مربوط به دکمه ی تساوی را مشاهده می کنید.

```
switch ( math_operator ) {
  case '+':
    total2 = total1 + Double.parseDouble(txtDisplay.getText());
    break;
  case '-':
    total2 = total1 - Double.parseDouble(txtDisplay.getText());
    break;
  case '/':
    total2 = total1 / Double.parseDouble(txtDisplay.getText());
    break;
  case '*':
    total2 = total1 * Double.parseDouble(txtDisplay.getText());
    break;
}

txtDisplay.setText(Double.toString(total2));
total1 = 0;
```

زمانی که کد جدید را به دکمه ی تساوی اضافه کرده اید، ماشین حساب را اجرا کرده و آن را امتحان کنید.

مورد زیر را امتحان کنید تا ببیند کار می کند یا نه.

```
58. 6 + 37.5 (answer should be 96.1)
    78 - 25.5 (answer should be 52.5)
    68 / 8 (answer should be 8.5)
    56.3 * 6.8 (answer should be 382.84)
```

اکنون یک ماشین حساب ساده دارید که می تواند عملکردهای جمع، تفریق، ضرب و تقسیم را انجام دهد.

حالا باید تمرین هایی با آبجکت های فرم داشته باشید، اجازه بدهید برنامه ی جدیدی ایجاد کنیم که از کنترل های متداول تری که روی یک فرم یافت می کنید، استفاده می کند.