آموزش زبان برنامه نویسی جاوا

گرافیک در جاوا – پکیج Swing

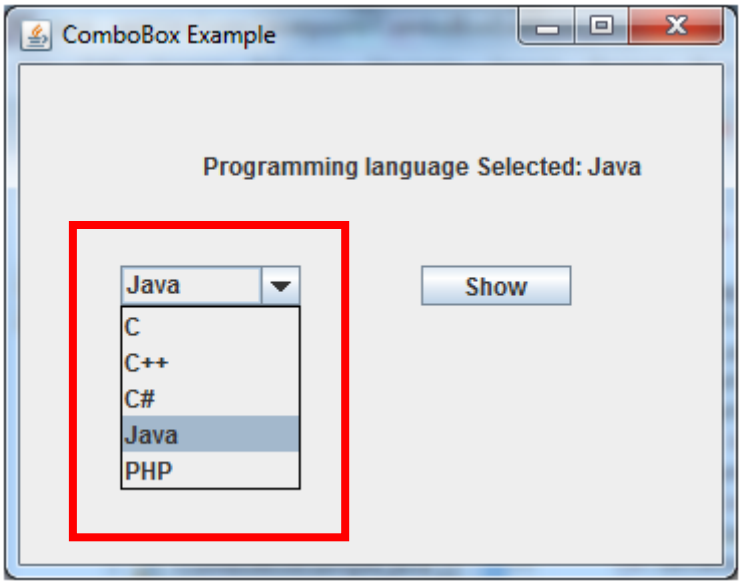
جلسه نهم

کلاس JComboBox

نویسنده : رحمان زارعی

جاوا را ساده،آسان و شیرین بنوشید!!!!

کلاس JComboBox جزیی از پکیج Swing در جاوا است که یک لیست کشویی انتخابی را ارائه می کند و به کاربر اجازه می دهید یک آیتم از میان لیست موجود را انتخاب کند. برای درک بهتر شکل ظاهری این اجزای گرافیکی تصویر(۱) را مشاهده کنید:



تصویر(۱)

- در تصویر(۱) بخش قرمز رنگ یک ComboBox را نشان می دهد.همان طور که در تصویر(۱) مشاهده می کنید یک ComboBox یک لیست کشویی است که از چند آیتم تشکیل شده است و ما تنها می توانیم یک آیتم را از میان لیست انتخاب کنیم.

سازنده های پرکاربرد کلاس JComboBox:

کاربرد	سازنده
ایجاد یک JComboBox با مدل داده ای پیش فرض	`JComboBox()`
ایجاد یک JComboBox که لیست آن را عناصر مشخصی از یک آرایه تشکیل می دهد.	`JComboBox(Object[] items)`

متدهای پرکاربرد کلاس JComboBox:

متد	کاربرد
`void addItem(Object anObject)`	این متد یک آیتم را به لیستی از آیتم های JComboBox اضاف می کند.
`void removeItem(Object anObject)`	این متد یک آیتم را از لیستی از آیتم های JComboBox حذف می کند
`void removeAllItems()`	این متد همه آیتم ها را از لیست حذف می کند
`void setEditable(boolean b)`	این متد تعیین می کند که ایا لیست کشویی (JComboBox) ایجاد شده قابل ویرایش باشد یا خیر. اگر پارامتر متد `true` باشد قابل ویرایش است در غیر این صورت اگر `false` بود غیر قابل ویرایش می باشد.
`void addActionListener(ActionListener a)`	برای افزودن یک ActionListener به لیست کشویی ما کاربر دارد. استفاده از این متد و ActionListener ها برای نسبت عمل و اکشنی خاص به آیتم های لیست کشویی می باشد. این متد برای سایر اجزای گرافیکی در جاوا که قصد داریم هنگام کلیک کردن یا انتخاب آنها یک عمل خاصی را انجام دهد استفاده می شود.
`void addItemListener(ItemListener i)`	برای افزودن یک ItemListener به لیست کشویی کاربرد دارد. این متد هم برای نسبت دادن عملی خاص در صورت رخ دادن رویداد مربوطه استفاده می شود.

خب روش پیاده سازی کلاس JComboBox در جاوا را بررسی می کنیم:

- ابتدا یک کلاس با نام ComboBoxExample تعریف می کنیم:

```
public class ComboBoxExample {


}
```

- قصد داریم اجزای گرافیکی خود را در سازنده کلاس ایجاد کنیم برای این کار سازنده ای در بدنه کلاس خود پیاده سازی می کنیم:

```
public class ComboBoxExample {

      ComboBoxExample() {

      }

}
```

- چون که قصد داریم از اجزای گرافیکی در برنامه خود استفاده کنیم پکیج Swing را در برنامه خود import می کنیم:

```
import javax.swing.*;

public class ComboBoxExample {

    ComboBoxExample() {

      }

}
```

- کاراکتر * یعنی این که در برنامه خود می توانیم از همه کلاس های موجود در پکیج Swing استفاده کنیم.
- حال در بدنه سازنده کلاس اجزای گرافیکی خود را تعریف می کنیم. همان طور که برای ساختن یک خانه ابتدا اسکلت خانه را می سازند و بعد سایر مصالح را به آن اضاف می کنند! در ساخت برنامه کاربردی در جاوا نیز ابتدا یک فریم که حکم اسکلت برنامه دارد را ایجاد می کنیم:

```
import javax.swing.*;

public class ComboBoxExample {

	ComboBoxExample() {
		JFrame f = new JFrame("ComboBox Example");

	}

}
```

- یک ارایه تعریف و مقداردهی اولیه می کنیم ، عناصر این آرایه آیتم های لیست JComboBox ما را تشکیل خواهد داد:

```
import javax.swing.*;

public class ComboBoxExample {

	ComboBoxExample() {
		JFrame f = new JFrame("ComboBox Example");
		 String city[]={"Borazjan","Bushehr","Shiraz","Tabriz","Tehran"};
	}

}
```

- حال یک شی از کلاس JComboBox ایجاد می کنیم:

```
import javax.swing.*;

public class ComboBoxExample {

	ComboBoxExample() {
		JFrame f = new JFrame("ComboBox Example");
		 String city[]={"Borazjan","Bushehr","Shiraz","Tabriz","Tehran"};
		 JComboBox cb=new JComboBox(city);
	}

}
```

- حالا مختصات و ابعاد لیست کشویی خود را در فریم تعیین می کنیم:

```
import javax.swing.*;

public class ComboBoxExample {
```

```
	ComboBoxExample() {
		JFrame f = new JFrame("ComboBox Example");
		 String city[]={"Borazjan","Bushehr","Shiraz","Tabriz","Tehran"};
		 JComboBox cb=new JComboBox(city);
		 cb.setBounds(50, 50,90,20);
	}

}
```

- در این مرحله شی cb را با صدا زدن متد add به frame خود اضافه می کنیم و چون قصد نداریم از طرح بندی خاصی برای اجزای گرافیکی خود استفاده کنیم بعد از صدا زدن متد setLayout مقدار پارامتر آن را null قرار می دهیم. :

```
import javax.swing.*;

public class ComboBoxExample {

	ComboBoxExample() {
		JFrame f = new JFrame("ComboBox Example");
		 String city[]={"Borazjan","Bushehr","Shiraz","Tabriz","Tehran"};
		 JComboBox cb=new JComboBox(city);
		 cb.setBounds(50, 50,90,20);
		 f.add(cb);
	    f.setLayout(null);
	}

}
```

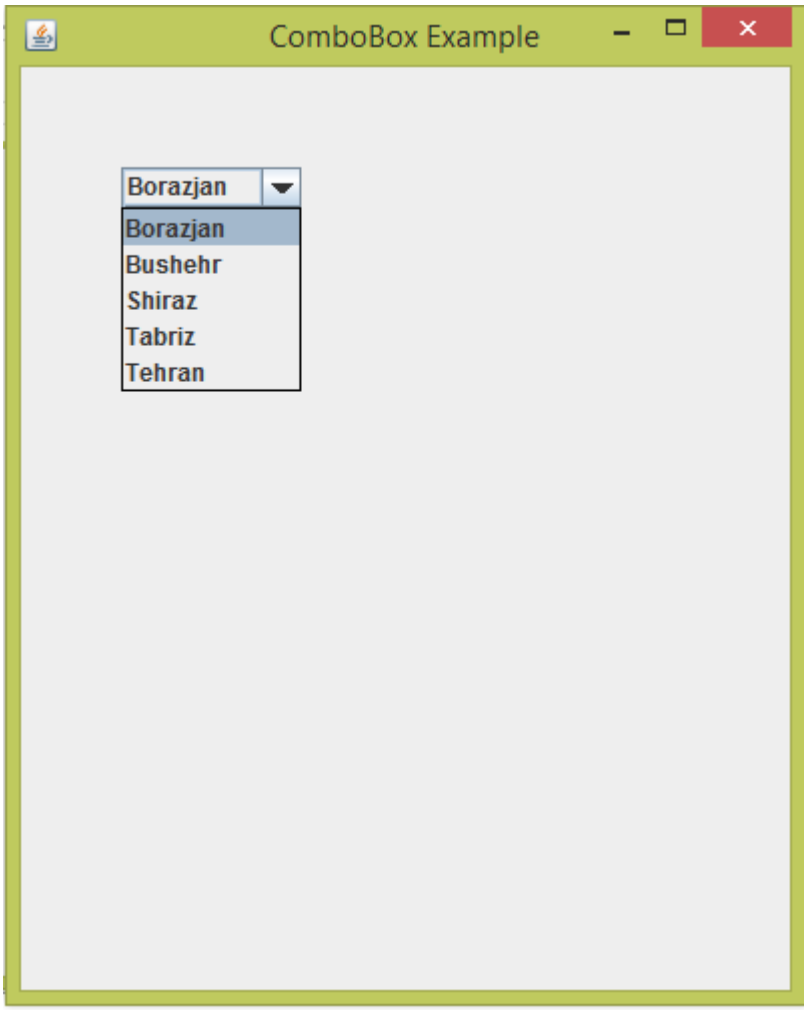
- تعیین سایز فریم برنامه و نمایش فریم و تمام اجزای گرافیکی :

```
import javax.swing.*;
public class ComboBoxExample {

	ComboBoxExample() {
		JFrame f = new JFrame("ComboBox Example");
		 String city[]={"Borazjan","Bushehr","Shiraz","Tabriz","Tehran"};
		 JComboBox cb=new JComboBox(city);
		 cb.setBounds(50, 50,90,20);
		 f.add(cb);
		 f.setSize(400,500);
		    f.setVisible(true);
	}

}
```

در گام اخر برای اجرای برنامه متد **main** را در بدنه کلاس پیاده سازی کرده و از کلاس خود شی ایجاد می کنیم و برنامه را کامپایل و اجرا می کنیم:

```
package swing_javalike;
import javax.swing.*;
public class ComboBoxExample {
JFrame f;
ComboBoxExample(){
    f=new JFrame("ComboBox Example");
    String city[]={"Borazjan","Bushehr","Shiraz","Tabriz","Tehran"};
    JComboBox cb=new JComboBox(city);
    cb.setBounds(50, 50,90,20);
    f.add(cb);
    f.setLayout(null);
    f.setSize(400,500);
    f.setVisible(true);
}
public static void main(String[] args) {
    new ComboBoxExample();
}
}
```

خروجی برنامه: تصویر(۲)



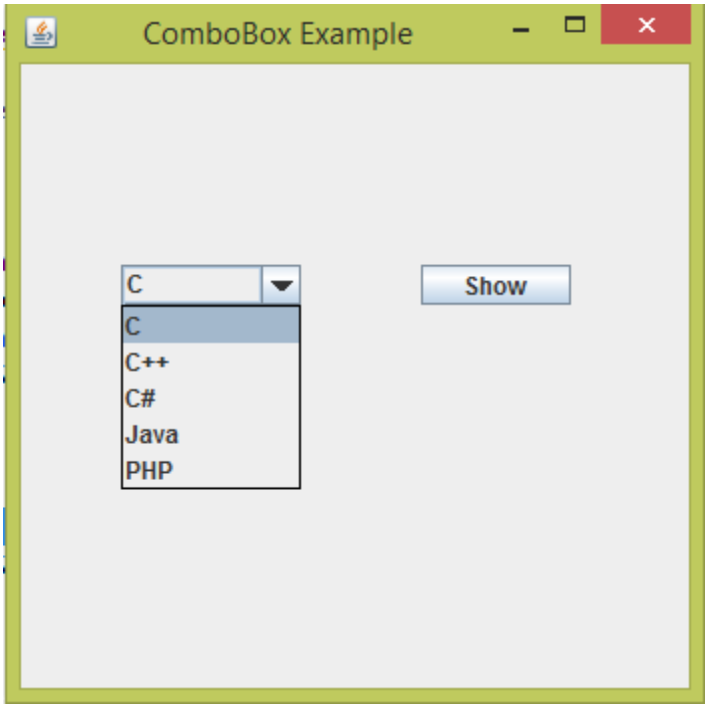
تصویر(۲)

- همان طور که در تصویر(۲) مشاهده می کنید یک لیست کشویی ایجاد کردیم که آیتم های آن عناصر یک ارایه می باشند.
- در این برنامه هیچ اکشن و عملی را برای لیست کشویی خود تعریف نکردیم.

مثال:

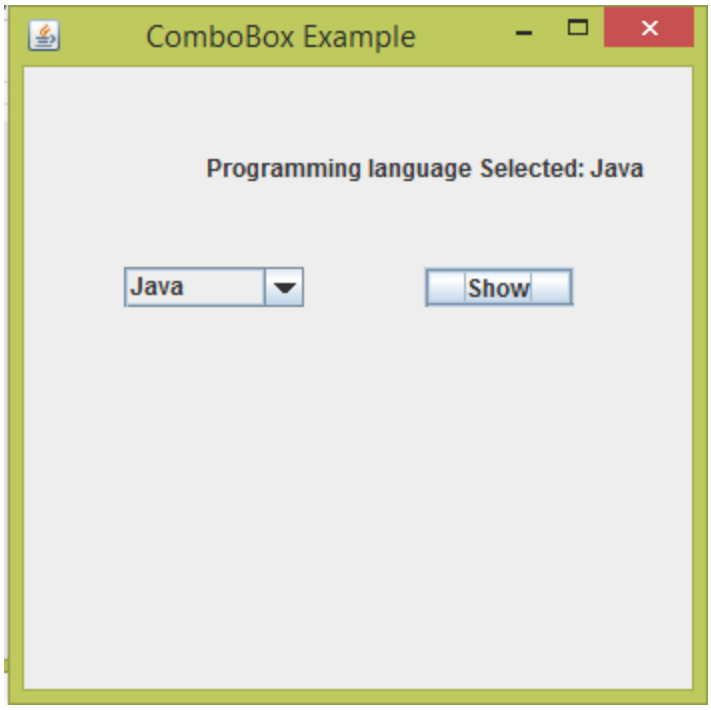
```
package javalike;

import javax.swing.*;
import java.awt.event.*;

public class ComboBoxExample {
	JFrame f;

	ComboBoxExample() {
		f = new JFrame("ComboBox Example");
		final JLabel label = new JLabel();
		label.setHorizontalAlignment(JLabel.CENTER);
		label.setSize(400, 100);
		JButton b = new JButton("Show");
		b.setBounds(200, 100, 75, 20);
		String languages[] = { "C", "C++", "C#", "Java", "PHP" };
		final JComboBox cb = new JComboBox(languages);
		cb.setBounds(50, 100, 90, 20);
		f.add(cb);
		f.add(label);
		f.add(b);
		f.setLayout(null);
		f.setSize(350, 350);
		f.setVisible(true);
		b.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				String data = "Programming language Selected: "
						+ cb.getItemAt(cb.getSelectedIndex());
				label.setText(data);
			}
		});
	}

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

خروجی: هنگام اجرای برنامه خروجی به صورت تصویر(۳) می باشد:



تصویر(۳)

- طبق تصویر(۳) ما یک لیست کشویی داریم که آیتم های آن را نام های زبان برنامه نویسی تشکیل داده است.با انتخاب یکی از نام ها و کلیک کردن روی دکمه `show` پیامی در `label` برنامه در بالا فریم نمایش داده می شود.ما آیتم `Java` را انتخاب می کنیم و دکمه `show`: تصویر(۴)



تصویر(۴)

توضیحات:

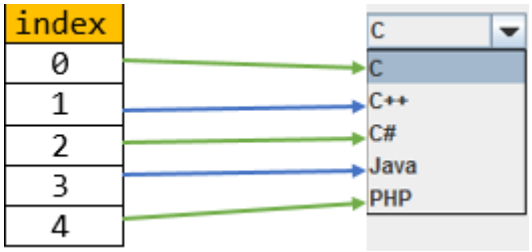
ما اکثر دستوراتی که در این کد هستش را در این جلسه و جلسات قبل بررسی کردیم. در اینجا دستورات زیر از مثال بالا را بررسی می کنیم:

```
1. b.addActionListener(new ActionListener() {
2. public void actionPerformed(ActionEvent e) {
3. String data = "Programming language Selected: "
4. + cb.getItemAt(cb.getSelectedIndex());
5. label.setText(data);
6. }
7. });
```

۱. برای اضاف کردن یک `ActionListener` به لیست کشویی استفاده می شود.برای افزودن یک `ActionListener` به لیست کشویی ، یک شی بصورت مستقیم از اینترفیس `ActionListener` ایجاد می کنیم .در بدنه این اینترفیس ، متد `actionPerformed` قرار دارد که وظیفه دریافت رویدادها را برعهده دارد مثلا وقتی روی یکی آیتم های لیست کشویی کلیک می کنیم یک رویداد رخ می دهد که توسط متد `actionPerformed` دریافت می شود و دستورات درون بدنه این متد اجرا می شود.

۳. یک متغیر از نوع `String` تعریف کرده که درون آن متن "`Programming language Selected`: " به همراه نام آیتم مورد نظر را می ریزد.

آیتم های درون یک لیست کشویی هر کدام یک ایندکس دارند که نشان دهنده مکان آن ها در بین آیتم های لیست کشویی است.مثل آرایه ها ،ایندکس آیتم های لیست کشویی از شماره `0` شروع می شود. اولین آیتم از بالای لیست دارای ایندکس شماره `0` می باشد و به ترتیب آیتم دوم از بالا دارای ایندکس شماره ۱ و آیتم سوم دارای ایندکس شماره ۲ و....آیتم `n` ام لیست کشویی دارای ایندکس شماره `n-1` می باشد. برای درک بهتر تصویر(۵) را مشاهده کنید:



تصویر(۵)

تا اینجا یادگرفتیم که هر آیتم از یک لیست کشویی `JComboBox` یک ایندکس دارد، پس راه دسرسی به هر ایتم از لیست کشویی `JComboBox` از طریق همین ایندکس ها می باشد.

خب حالا چطوری از طریق ایندکس به آیتم های یک لیست کشویی دسری پیدا کنیم؟

پاسخ: برای دسرسی به آیتم های لیست کشویی از متد زیر استفاده می کنیم:

```
getItemAt(int arg);
```

- با جایگزین کردن شماره ایندکس آیتم مربوطه با پارامتر `arg` این متد آیتم مورد نظر را به ما می دهد، مثلا اگر در تصویر(۵) شماره ایندکس ۳ را به متد `getItemAt` دهیم این متد آیتم `Java` را به ما پس می دهد.
- خب این متد با دریافت ایندکس ، آیتم را از لیست کشویی به ما پس می دهد، حالا چطور با انتخاب هر آیتم شماره ایندکس آن را دریافت کنیم؟ متد زیر شماره ایندکس آیتم انتخاب شده از لیست کشویی را به ما پس می دهد:

```
getSelectedIndex();
```

- با کلیک کردن روی یکی از آیتم های لیست کشویی و انتخاب آن ، این متد شماره ایندکس آیتم انتخاب شده را برمی گرداند.

```
cb.getItemAt(cb.getSelectedIndex());
```

- این دستور `cb.getSelectedIndex` شماره ایندکس آیتم انتخاب شده توسط کاربر را برمیگرداند و مقدار برگردانده شده را درون پارامتر متد `getItemAt` می ریزد و متد `getItemAt` آیتم مربوط به شماره ایندکس دریافت شده را برای ما برمیگرداند.

```
label.setText(data);
```

در پایان متغیر `data` که حاوی متن "`Programming language Selected`: " و آیتم انتخاب شده از لیست کشویی هستش را درون `label` برای نمایش در فریم قرار می دهیم.

برای بهبود کیفیت و کمیت آموزش جاوا در وب فارسی ما را حمایت کنید.

پیروز و موفق باشید

هر روز مفاهیم و مثال های جدید به سایت اضافه می شود برای اطلاع از مطالب جدید روی سایت عضو کانال شوید.