آموزش زبان برنامه نویسی جاوا

گرافیک در جاوا – پکیج Swing

جلسه سیزدهم

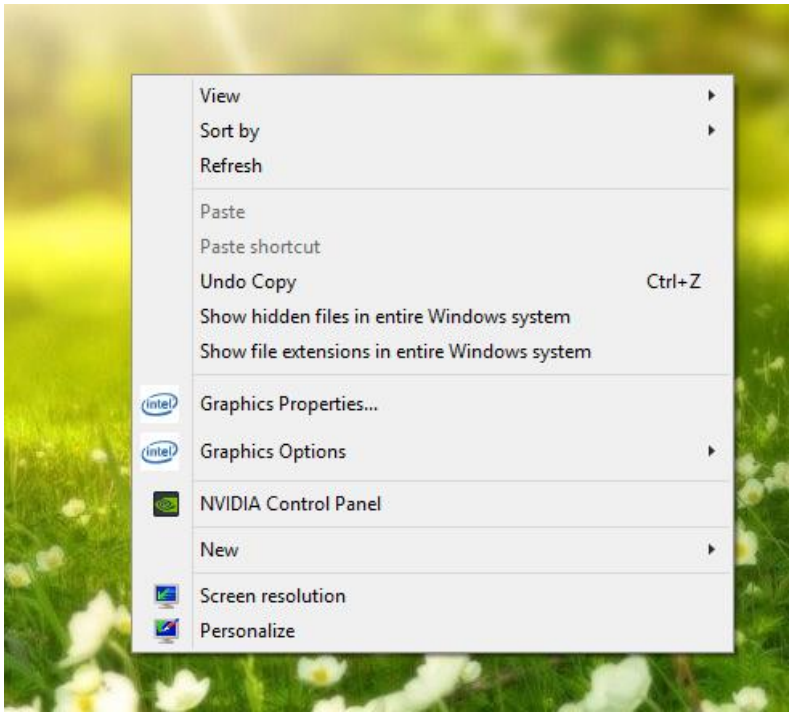
کلاس JPopupMenu

نویسنده : رحمان زارعی

جاوا را ساده،آسان و شیرین بنوشید!!!!

یکی دیگر از اجزای گرافیکی که جاوا برای ساخت برنامه های دسکتاپی ارائه کرده کلاس JPopupMenu می باشد.

کلاس PopupMenu **منویی** است که می تواند بصورت **پویا** در یک **موقعیت مشخص** در یک **اجزای گرافیکی** ظهور کند. برای درک بهتر از شکل ظاهری PopupMenu می توانیم به مثال کلیک سمت راست در صفحه دسکتاپ کامپیوتر اشاره کنیم،وقتی ما در یک موقعیت مشخص از صفحه دسکتاپ با موس کلیک سمت راست می کنیم یک لیستی از گزینه ها نمایش داده می شود که این لیست همان PopupMenu می باشد.تصویر(۱)



تصویر(۱)

- همان طور که میدانید ما در هر مکان یا موقعیت از صفحه دکستاپ کلیک سمت راست کنیم این منو پویا نمایش داده می شود. خب در این جلسه قصد داریم روش ساخت این منو را یاد بگیریم که با کلیک کردن (راست یا چپ) موس در برنامه خود یک PopupMenu نمایش داده شود.

• سازنده های پرکاربرد کلاس JPopupMenu :

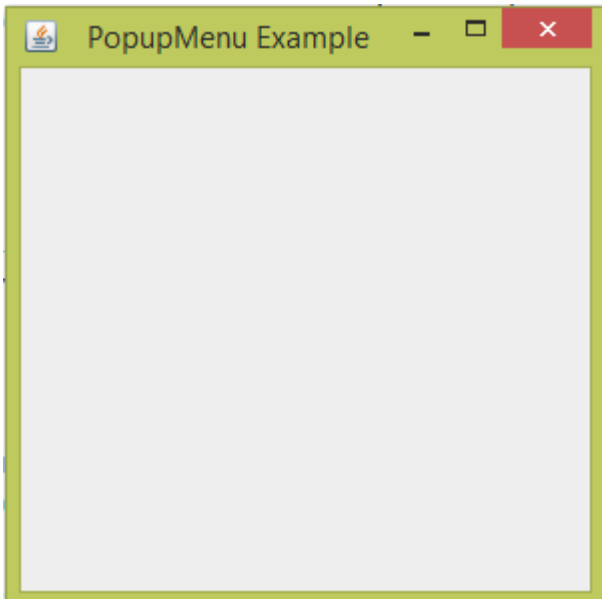
کاربرد	سازنده
برای ایجاد یک JPopupMenu بدون پارامتر	JPopupMenu()
برای ایجاد یک JPopupMenu با عنوان مشخص	JPopupMenu(String label)

خب توضیحات زیادی هم خوب نیست! بریم سراغ حل مثال ☺

مثال:

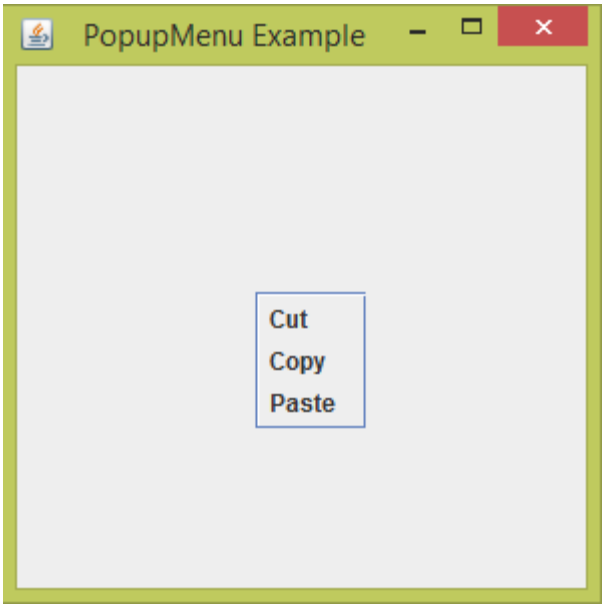
```
package javalike;

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

class PopupMenuExample {
      PopupMenuExample() {
            final JFrame f = new JFrame("PopupMenu Example");
            final JPopupMenu popupmenu = new JPopupMenu("Edit");
            JMenuItem cut = new JMenuItem("Cut");
            JMenuItem copy = new JMenuItem("Copy");
            JMenuItem paste = new JMenuItem("Paste");
            popupmenu.add(cut);
            popupmenu.add(copy);
            popupmenu.add(paste);
            f.addMouseListener(new MouseAdapter() {
                  public void mouseClicked(MouseEvent e) {
                        popupmenu.show(f, e.getX(), e.getY());
                  }
            });
            f.add(popupmenu);
            f.setSize(300, 300);
            f.setLayout(null);
            f.setVisible(true);
      }

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

خروجی: برنامه هنگام اجرا به صورت تصویر(۲) خواهد بود:



تصویر(۲)

- حال اگر هر جایی از فریم برنامه کلیک سمت راست یا چپ کنیم JPopupMenu ما نمایش داده می شود.تصویر(۳)



تصویر(۳)

```
final JPopupMenu popupmenu = new JPopupMenu("Edit");
```

- مثل هر اجزای گرافیکی دیگر برای استفاده از JPopupMenu از آن شی ایجاد کرده و متنی را که نقش عنوان منوی ما را دارد جایگزین پارامتر سازنده آن می کنیم.

```
JMenuItem cut = new JMenuItem("Cut");
JMenuItem copy = new JMenuItem("Copy");
JMenuItem paste = new JMenuItem("Paste");
```

- آیتم های JPopupMenu خود را با استفاده از کلاس JMenuItem می سازیم.متن درون سازنده کلاس JMenuItem عنوان هر آیتم را نشان می دهد.

```
popupmenu.add(cut);
popupmenu.add(copy);
popupmenu.add(paste);
```

- آیتم های خود را به منو JPopupMenu خود اضاف می کنیم.

```
f.addMouseListener(new MouseAdapter() {
                public void mouseClicked(MouseEvent e) {
                    popupmenu.show(f, e.getX(), e.getY());
                }
            });
```

- همان طور که در ابتدا گفتیم JPopupMenu با کلیک کردن موس در یک مکان از برنامه نمایش داده می شود.پس ما باید از دستوری استفاده کنیم که بتواند رویداد مربوط به کلیک کردن موس در برنامه را شناسایی و دریافت کند و سپس عمل مورد نظر ما را اجرا کند.
- MouseAdapter یک کلاس انتزاعی یا abstract است که برای دریافت رویدادهای مربوط به موس استفاده می شود. منظور از رویدادهای موس همان کلیک کردن و دکمه های موس می باشد.
- کلاس MouseAdapter در بدنه خودش متدهایی دارد که یکی از آنها متد mouseClicked می باشد، متد mouseClicked وظیفه اش دریافت رویدادهای مربوط به کلیک کردن موس می باشد ، یعنی بار هر بار کلیک کردن موس این متد صدا زده می شود، و دستورات درون بدنه آن اجرا می شود.
- addMouseListener برای ثبت رویدادهای مربوط به موس در یک اجزای گرافیکی استفاده می شود. مثلا در اینجا شی f که از نوع کلاس JFrame هستش این متد را صدا زده است.این باعث میشه هر کجای فریم با موس کلیک کنیم ، رویداد رخ داده به فریم اضاف شود. به بیان ساده تر این متد باعث میشه frame به کلیک کردن موس روی آن حساس شود.

```
f.addMouseListener(new MouseAdapter() {
                public void mouseClicked(MouseEvent e) {
```

```
                popupmenu.show(f, e.getX(), e.getY());
            }
        });
```

- در دستور بالا برای اضاف کردن رویداد موس به فریم متد addMouseListener را صدا زده ایم.
- شی از کلاس انتزاعی MouseAdapter به جای پارامتر متد addMouseListener ایجاد می کنیم.
- متد mouseClicked را overrid یا بازنویسی می کنیم.

```
popupmenu.show(f, e.getX(), e.getY());
```

- متد show یکی از متدهای مربوط به کلاس JPopupMenu می باشد. این متد برای نمایش منو JPopupMenu در مختصات x و y اجزای گرافیکی ما کاربرد دارد. در اینجا برای نمایش JPopupMenu ما در مختصات x و y فریم ما استفاده شده است.
- سنتکس یا نحوه نوشتن متد show بصورت زیر است:

```
public void show( Component origin, int x, int y )
```

۱. بجای پارامتر Component origin یک شی از اجزای گرافیکی که قراره منو JPopupMenu با کلیک کردن روی آن نمایش داده شود، قرار می گیرد.

۲. x و y نیز موقعیت نمایش منو JPopupMenu را در اجزای گرافیکی مشخص می کند/

```
3. popupmenu.show(f, e.getX(), e.getY());
```

- در اینجا شی f که از نوع کلاس JFrame می باشد را به عنوان اجزای گرافیکی که قراره منو ما در آن نمایش داده شود به عنوان پارامتر اول متد show قرار داده ایم.

```
e.getX(), e.getY()
```

- e از نوع کلاس MouseEvent می باشد، یعنی رویدادهایی که از طرف موس رخ می دهد در شی e ذخیره می شود.
- حال قصد داریم مختصات نقطه ای از اجزای گرافیکی که روی آن نقطه با موس کلیک شده است را دریافت کنیم، برای این کار از طریق شی e متدهای getX و getY را صدا می زنیم.
- متد getX مقدار x مکان کلیک شده را می دهد.
- متد getY مقدار y مکان کلیک شده را می دهد.

```
1.      f.add(popupmenu);
2.           f.setSize(300, 300);
3.           f.setLayout(null);
4.           f.setVisible(true);
```

۱. افزودن شی از نوع کلاس JPopupMenu به فریم.

۲. تعیین اندازه فریم

۳. چون از طرح بندی خاصی استفاده نکردیم مقدار پارامتر این متد را `null` قرار دادیم.

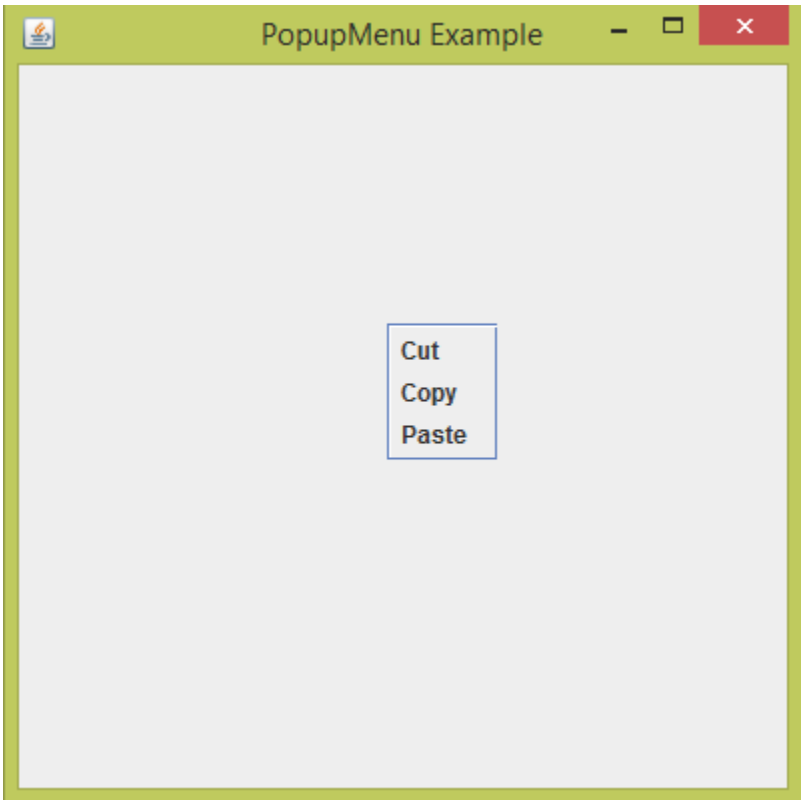
۴. برای نمایش فریم و تمام اجزای گرافیکی این متد را صدا زده و پارامتر آن را `true` قرار می دهیم.

مثال:

```
package javalike;

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

class PopupMenuExample {
     PopupMenuExample() {
          final JFrame f = new JFrame("PopupMenu Example");
          final JLabel label = new JLabel();
          label.setHorizontalAlignment(JLabel.CENTER);
          label.setSize(400, 100);
          final JPopupMenu popupmenu = new JPopupMenu("Edit");
          JMenuItem cut = new JMenuItem("Cut");
          JMenuItem copy = new JMenuItem("Copy");
          JMenuItem paste = new JMenuItem("Paste");
          popupmenu.add(cut);
          popupmenu.add(copy);
          popupmenu.add(paste);
          f.addMouseListener(new MouseAdapter() {
               public void mouseClicked(MouseEvent e) {
                    popupmenu.show(f, e.getX(), e.getY());
               }
          });
          cut.addActionListener(new ActionListener() {
               public void actionPerformed(ActionEvent e) {
                    label.setText("cut MenuItem clicked.");
               }
          });
          copy.addActionListener(new ActionListener() {
               public void actionPerformed(ActionEvent e) {
                    label.setText("copy MenuItem clicked.");
               }
          });
          paste.addActionListener(new ActionListener() {
               public void actionPerformed(ActionEvent e) {
                    label.setText("paste MenuItem clicked.");
```

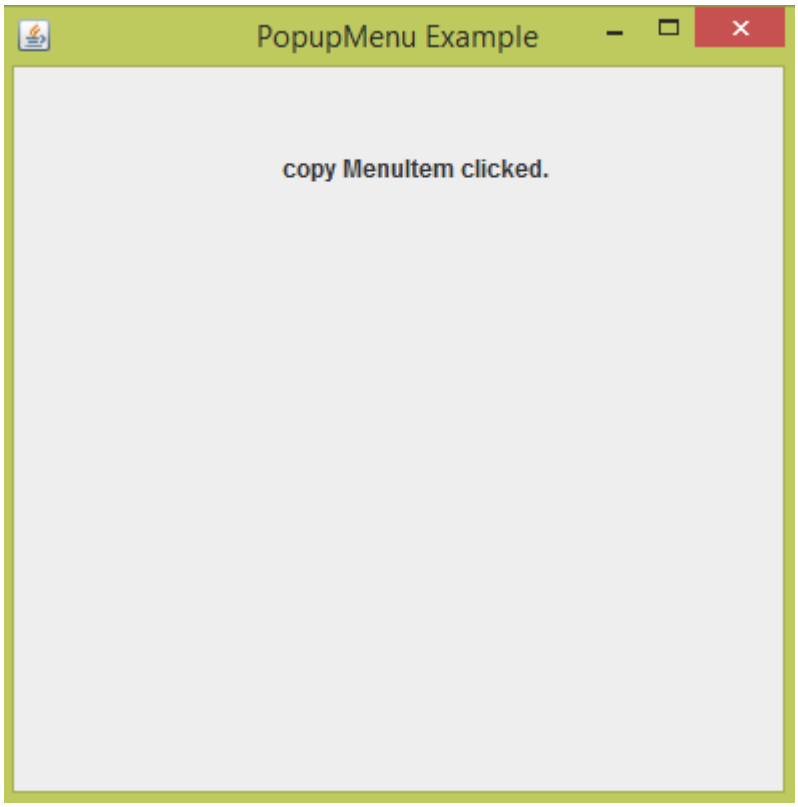
```
                }
            });
            f.add(label);
            f.add(popupmenu);
            f.setSize(400, 400);
            f.setLayout(null);
            f.setVisible(true);
      }

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

خروجی: بعد از اجرای برنامه و کلیک کردن روی فریم خروجی بصورت تصویر(۴) خواهد بود:



تصویر(۴)

- حال گزینه copy را از میان آیتم های PopupMenu انتخاب می کنیم، خروجی بصورت تصویر(۵) خواهد بود:



تصویر(۵)

- همان طور که در تصویر مشاهده می کنید با انتخاب گزینه copy از لیست منو ظاهر شده، پیامی در لیبل موجود در بالای فریم برنامه نمایش داده شده است.

توضیحات:

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

- برای استفاده تمام رویدادهای موس و صفحه کلید و.... از این پکیج در برنامه خود استفاده می کنیم.

```
label.setHorizontalAlignment(JLabel.CENTER);
```

- متد setHorizontalAlignment برای تنظیم حالت افقی label ما کاربرد دارد. و پارامتر JLabel.CENTER میگه لیبل ما را وسط یا مرکز حالت افقی در فریم قرار بده.

```
    cut.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                label.setText("cut MenuItem clicked.");
            }
        });
```

- اگر جلسات قبل رو مطالعه کرده باشید باید با متد addActionListener آشنا باشید ، این متد برای اضاف کردن یک ActionListener به اجزای گرافیکی ما نظیر دکمه ها، منوها،آیتم ها و...استفاده می شود.
- در اینجا قصد داریم یک ActionListener را به شی cut که از نوع کلاس JMenuItem می باشد اضاف کنیم. که با انتخاب این آیتم عمل خاصی اجرا شود.

```
public void actionPerformed(ActionEvent e) {
                    label.setText("cut MenuItem clicked.");
                }
```

- متد actionPerformed رویدادهای مربوط به انتخاب یا کلیک کردن روی اجزای گرافیکی را دریافت می کند.مثل فشردن دکمه button، گزینه ای در منو ها و...
- وقتی رویدادی رخ دهد متد actionPerformed صدا زده می شود و دستورات درون بدنه ان اجرا می شود.

```
label.setText("cut MenuItem clicked.");
```

- اگر متد actionPerformed صدا زده شود دستور فوق اجرا می شود.
- این دستور یک متن را درون label قرار میدهد.

سایر موارد در این جلسه و جلسات قبل توضیح داده ایم.

پیروز و موفق باشید

بازدید از سایت

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

دخل و تصرف ، ویرایش و کپی زدن تمامی آموزش های جاوالایک به دور از اخلاق حرفه ای ست و حرام می باشد.