به نام خدا

دستور کار کارگاه برنامه‌نویسی پیشرفته

جلسه هفتم

آشنایی با طراحی رابط کاربری گرافیکی در جاوا

مقدمه

در این جلسه با مولفه‌های رابط کاربری گرافیکی در جاوا آشنا می‌شویم. در هر یک از این مولفه‌ها، نمایش گرافیکی و موارد استفاده آن‌ها را مورد بررسی قرار می‌دهیم. همین‌طور با روش‌های مدیریت چینش مولفه‌های مختلف در صفحات و پنل‌ها، و تغییر تم‌های گرافیکی آشنا می‌شویم. به علاوه برای آشنایی بیشتر و کسب مهارت در طراحی رابط‌های کاربری، رابط کاربری گرافیکی یک ماشین حساب را طراحی و پیاده‌سازی می‌کنیم.

تمام مولفه‌های مورد استفاده در این جلسه، در پکیج javax.swing پیاده‌سازی شده‌اند. برای آشنایی بیشتر با امکانات این مولفه‌ها می‌توانید به مستندات این پکیج مراجعه نمایید.

مراحل انجام کار

- ایجاد صفحه اصلی: برای پیاده‌سازی صفحه اصلی یک رابط کاربری گرافیکی، از کلاس JFrame استفاده می‌نماییم. یک نمونه از این کلاس در ابتدای فرایند ایجاد رابط کاربری ساخته می‌شود و سایر مولفه‌های گرافیکی برای نمایش در ادامه به آن اضافه می‌شوند.

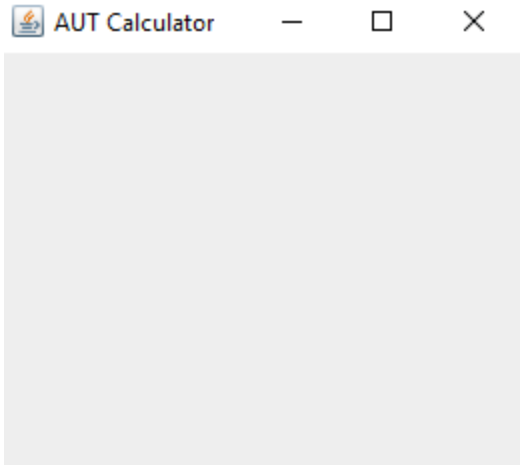
```
import javax.swing.*;

public class CalculatorGUI {
    JFrame calcFrame;

    public CalculatorGUI() {
        calcFrame = new JFrame();
        calcFrame.setTitle("AUT Calculator");
        calcFrame.setSize(300, 300);
        calcFrame.setLocation(100, 200);
        calcFrame.setLayout(null);
        calcFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        calcFrame.setVisible(true);
    }
}
```

```
public class Main {

    public static void main(String[] args) {
        CalculatorGUI calculator = new CalculatorGUI();
    }
}
```



شکل بالا خروجی قطعه کدهای نوشته‌شده را نمایش می‌دهد. این قطعه کد، یک قاب (پنجره) به عرض ۳۰۰ و ارتفاع ۳۰۰ پیکسل در نقطه شروع ۱۰۰ پیکسل از چپ و ۲۰۰ پیکسل از بالای صفحه نمایش ایجاد می‌نماید و عبارت "AUT Calculator" را در عنوان این قاب مشخص می‌کند.

متد setLayout در این قطعه کد، نحوه چینش مولفه‌ها در این قاب مشخص شده است. به نمونه‌های ساخته‌شده از کلاس‌هایی که وظیفه چینش مولفه‌های گرافیکی را بر عهده دارند، LayoutManager گفته می‌شود. انواع مختلفی از این Layout Managerها در پکیج java.awt پیاده‌سازی شده‌اند که می‌توان از آن‌ها استفاده کرد. هر یک از این کلاس‌ها، مولفه‌ها را به نحوه خاصی در صفحه می‌چینند (اینجا را مطالعه کنید) که بسته به نیاز می‌تواند مورد استفاده قرار بگیرد:

- BorderLayout
- BoxLayout
- CardLayout
- FlowLayout
- GridBagLayout
- GridLayout
- GroupLayout
- SpringLayout

در صورتی‌که از هیچ‌یک از این کلاس‌ها برای چینش مولفه‌ها استفاده نشود، باید ابعاد و نقطه قرارگیری همه مولفه‌ها صریحاً و با استفاده از متدهای setSize و setLocation مشخص شود. علی‌رغم این‌که استفاده از Layout Managerها باعث responsiveشدن مولفه‌ها در واکنش به تغییرات قاب می‌شود و بهتر است همواره از آنها استفاده کرد، در خط ۹ این قطعه کد، برای چینش مولفه‌ها در قاب اصلی از روش تعیین دقیق مولفه‌ها استفاده کرده‌ایم تا نحوه چینش در این شرایط را تمرین کرده باشیم. با پاس‌دادن آرگومان null به متد setLayout، مولفه‌های گرافیکی swing از چینش خودکار مولفه‌ها در قاب جلوگیری کرده و این امکان را به برنامه‌نویسان می‌دهد که ابعاد و نقاط قرارگیری مولفه‌های مختلف را خودشان انتخاب نمایند. در ادامه این دستور کار با layoutهای مختلف که در پکیج awt موجود است، آشنا می‌شویم.

متد setDefaultCloseOperation عکس‌العمل برنامه در هنگام بستن قاب اصلی را مشخص می‌کند. به طور پیش‌فرض برنامه با بستن قابل اصلی پایان نمی‌گیرد. با ارسال آرگومان JFrame.EXIT_ON_CLOSE که یک enum تعریف شده در کلاس JFrame است، بستن قاب منجر به پایان‌یافتن برنامه می‌شود.

در انتها برای نمایش قاب اصلی و تمام مولفه‌های اضافه‌شده به آن، باید از متد setVisible استفاده نماییم. true قراردادن آرگومان این متد، باعث نمایش قاب اصلی در صفحه می‌شود. این متد معمولاً پس از اضافه‌کردن تمام مولفه‌ها به قاب، فراخوانی می‌شود.

- بخش‌بندی مولفه‌ها: یکی از تمرین‌های خوب برنامه‌نویسی در تولید رابط‌های کاربری گرافیکی، بخش‌بندی مولفه‌های گرافیکی است. این بخش‌بندی معمولاً بر اساس کاربرد مولفه‌ها انجام می‌گیرد. کلاس JPanel به منظور بخش‌بندی مولفه‌ها و دسته‌بندی آن‌ها برای کنترل بهتر مورد استفاده قرار می‌گیرد.

```
JPanel keyboardPanel = new JPanel();
keyboardPanel.setSize(200,200);
keyboardPanel.setLocation(10, 150);
keyboardPanel.setLayout(new GridLayout(4,3));

calcFrame.add(keyboardPanel);
```

در این قطعه کد، یک پنل به عرض ۲۰۰ و ارتفاع ۲۰۰ پیکسل در نقطه ۱۰ و ۱۵۰ نسبت به قاب اصلی ایجاد شده است. برای چینش مولفه‌هایی که به این پنل اضافه می‌شوند، از GridLayout استفاده

کرده‌ایم. به این ترتیب، پنل به یک جدول با ۴ سطر و ۳ ستون تقسیم شده و تمام مولفه‌هایی را که پس از این به پنل اضافه می‌شود، به ترتیب از چپ به راست و از بالا به پایین به پنل اضافه می‌کند. البته می‌توان در اضافه‌کردن مولفه‌ها به پنل، سطر و ستون دلخواه را نیز مشخص کرد. ابعاد تمام مولفه‌هایی که به این پنل اضافه خواهند شد، با هم برابر خواهد بود.

- طراحی کی‌بورد: در این قسمت می‌خواهیم یک کی‌بورد شامل ارقام برای این ماشین حساب طراحی نماییم. برای این منظور نیاز به ایجاد ۱۰ دکمه که هر یک مربوط به یکی از ارقام ۰ تا ۹ باشند و دو دکمه دیگر برای جمع (+) و مساوی (=) در نظر می‌گیریم.

```
for (int i = 9; i > 0; i--) {
    JButton btn = new JButton();
    btn.setText("" + i);
    keyboardPanel.add(btn);
}

JButton sumBtn = new JButton();
sumBtn.setText("+");
keyboardPanel.add(sumBtn);

JButton zeroBtn = new JButton();
zeroBtn.setText("0");
keyboardPanel.add(zeroBtn);

JButton doBtn = new JButton();
doBtn.setText("=");
keyboardPanel.add(doBtn);
```

- صفحه نمایش: برای تولید صفحه نمایش از یک JTextArea استفاده می‌نماییم. این کلاس معمولاً برای دریافت ویا نمایش متن طولانی از کاربر استفاده می‌شود.

```
JTextArea display = new JTextArea(3,10);
display.setEditable(false);
display.setFont(new Font("Arial", 14,14));
```

این قطعه کد، یک مولفه گرافیکی ایجاد می‌کند که می‌توان یک متن را در آن قرار داد. خط دوم از این قطعه کد، امکان ویرایش این متن توسط کاربر را غیرفعال می‌کند. در نتیجه این مولفه فقط برای نمایش یک متن به کاربر مورد استفاده قرار می‌گیرد.

از آنجا که در این ماشین‌حساب قصد داریم تمام عملیات‌های فعلی و قبلی کاربر را نمایش دهیم، ممکن است لیست این عملیات طولانی شود. برای نمایش راحت‌تر به کاربر باید این صفحه نمایش این امکان را داشته باشد که با استفاده از یک میله لغزان بتوان در لیست عملیات‌ها بالا و پایین رفت. برای ایجاد میله لغزان روی این صفحه نمایش از کلاس JScrollPane استفاده می‌نماییم.

```
JScrollPane scrollPane = new JScrollPane(display);
scrollPane.setPreferredSize(new Dimension(200, 100));
scrollPane.setLocation(50,20);
calcFrame.getContentPane().add(scrollPane);
```

- ارتقاء رابط کاربری: یکی از راه‌های مناسب برای بهبود رابط کاربری، استفاده از LookAndFeelهای مختلف است. نمونه‌های مختلف LookAndFeel، شکل ظاهری و تم مولفه‌های گرافیکی را تغییر می‌دهند. قطعه کد زیر نمونه‌ای از تغییر LookAndFeel پیش‌فرض سیستم را نمایش می‌دهد. تم "Nimbus" یکی از LookAndFeelهای موجود برای ایجاد رابط‌های کاربری گرافیکی به شمار می‌رود.

```
try {
    for (UIManager.LookAndFeelInfo info : UIManager.getInstalledLookAndFeels()) {
        if ("Nimbus".equals(info.getName())) {
            UIManager.setLookAndFeel(info.getClassName());
            break;
        }
    }
} catch (Exception e) {
    // If Nimbus is not available, you can set the GUI to another look and feel.
}
```

انجام دهید

یک رابط کاربری گرافیکی برای یک ماشین حساب مهندسی طراحی نمایید که امکان تغییر حالت از حالت ساده به حالت مهندسی را داشته باشد. برای این کار از مولفه گرافیکی JTabbedPane استفاده نمایید.

در حالت ساده، این ماشین حساب شامل عملیات جمع، ضرب، تقسیم، تفریق و باقی‌مانده‌گیری است.

در حالت مهندسی، این ماشین‌حساب علاوه بر تمام امکانات حالت ساده، قادر به محاسبه توابع مثلثاتی از جمله sin/cos و tan/cot، توابع نمایی و لگاریتمی شامل exp و log است. علاوه بر این

یک دکمه دیگر برای استفاده از مقادیر عدد پی و عدد نپر در محاسبات باید در کی‌بورد این ماشین‌حساب وجود داشته باشد.

یک دکمه به نام shift به دکمه‌های این ماشین حساب در حالت مهندسی اضافه نمایید که در صورت زدن آن بتوان عملکرد توابع مثلثاتی، نمایی و لگاریتمی را با یک‌دیگر جابجا کرد. حالت پیش‌فرض دکمه sin/cos (و tan/cot) را سینوس (تانژانت) در نظر بگیرید که با فشردن shift، به کسینوس (کتانژانت) تبدیل می‌شود.

توجه! در این جلسه تنها طراحی و پیاده‌سازی رابط کاربری گرافیکی مورد نظر است و پیاده‌سازی عملکردها در جلسات آتی انجام خواهد شد.