Sofia University
Department of Mathematics and Informatics

Course : OO Programming Java
Date: December 12, 2009
Student Name:

Lab No. 10 (GUI)

Submit the all Java files developed to solve the problems listed below. Use comments and Modified-Hungarian notation.

Exercises on Swing GUI

Problem 1

Creating custom GUI components

A) Consider **`class StillClock`** (provided as **`StillClock.rar`**). It draws a still clock in a **`JPanel`**. **Inherit `class AnimatedClock`** from **`class StillClock`** to develop an animated clock **making use** of a **`Timer` object** to set the time of the **`StillClock`** every **`1000 ms`** (i.e. every 1 sec) by means if the method **`setCurrentTime()`** of the given class **StillClock**. The constructor of **`class Timer`** takes two arguments. The **first argument** defines **the number of `millisecs`** the timer have to pass before it fires an **`ActionEvent`** object, the **second argument** is the **`ActionListener`** object (event handler) that processes this object in its **`actionPerformed()`** method. Define **the event handler** in an **inner class** of **class `AnimatedClock`**. **Note**, that after the **`Timer`** object is created , it has to be started using the **`start()`** method of **`class Timer`** in order to make it fire every time the predefined number of **`millisecs`** elapse.

Test `class AnimatedClock` in a **`JFrame`** application.

B) Inherit a new type of button from **`JButton`**. It should have a circle drawn in the middle. Initially, the circle has to be filled with green color. Each time you click this button, it should toggle its color to red. (use method **`getGraphics()`** as shown in the sample file **`Lab-11RoundButton.rar`)**

C) Inherit a new type of button from **`JButton`**. The new **`JButton`** should display as a **`Rectangle`** with rounded corners (**`RoundRectangle2D`**) and have custom color to display , which changes to another predefined color on **`click`** The two **`Color`**s, the width, height and the custom **`Button`** label must properties (set, get), as well as, the **`Button`** label text. (***see*** the sample file **`Lab- 11RoundButton.rar`)**

Problem 2

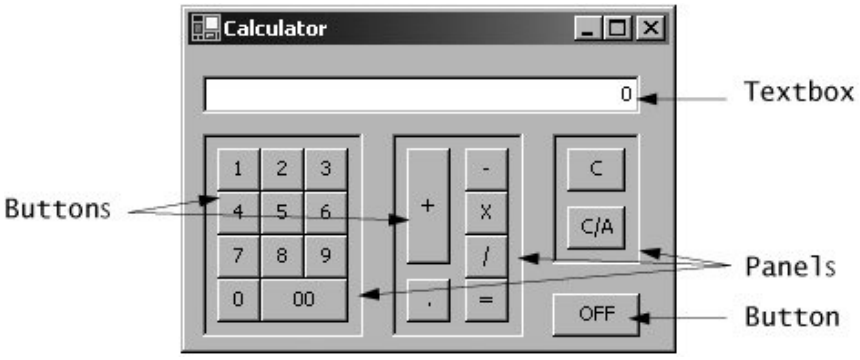
Create a calculator as a **`JPanel`** ***component*** that allows it to reused in **`JFrame`** and **`Applet`** applications. The **`Calculator`** should allow the user to input numbers in a textbox and choose an operation to perform on them (addition, multiplication, division, subtraction) with **`JButtons`** as it is done with a usual calculator (*see the design of the* **`Calculator`** *application in the Accessories Program group in the MS Windows environment*) . Design the **`Layout`** of the buttons and the textbox to execute these operations, as well as, support for handling the following events:

a) to remember the currently displayed number (**M** operation)
b) to add the currently displayed number with the number stored in memory and display the result (**M+** operation)
c) to subtract the currently displayed number with the number stored in memory and display the result (**M-** operation)
d) to clear the memory (**MC-** operation

The methods performing the Calculator operations must be **public.** There should be also **two** ***`public set`*** **properties** for the user numeric input, necessary to complete the calculator operations**.** There should be a ***`public get`*** **property** for the Calculator result.

Catch division by zero exceptions, by canceling the division operation and displaying an error message in the textbox. **Allow only legal numeric user input** in the textbox**.**

Write a `JFrame` to test this **`JPanel` calculator class**.



Problem 3

Extend the program of **`Fig. 11.34- 35`** from **lecture 10a** **to :**

a) **connect each point with the next point through a line while dragging the mouse** (read **`Lecture`** 10c, **`Fig 12.27`**).

b) **include options for changing the size and color of the lines** drawn as shown below. **Create a GUI** similar to the one below given (read Lecture 10c, Fig 12.29) using the Visual Modeler of NetBeans.

The user should be able to **draw on the application's Panel**. To retrieve a **Graphics** object for drawing, call method ***`panelName`*`.getGraphics(),`** substituting in the name of your **`Panel`**.