

ΓΡΑΦΙΚΑ ΠΕΡΙΒΑΛΛΟΝΤΑ

Η βιβλιοθήκη SWING

Event-driven programming

Swing

- Τα **GUIs** (**Graphical User Interfaces**) είναι τα συνηθισμένα interfaces που χρησιμοποιούν παράθυρα, κουμπιά, menus, κλπ
- Η **Swing** είναι βιβλιοθήκη της Java για τον προγραμματισμό τέτοιων interfaces.
 - Η μετεξέλιξη του **AWT** (**Abstract Window Toolkit**) το οποίο ήταν το πρώτο αλλά όχι τόσο επιτυχημένο πακέτο της Java για GUI.
 - Τώρα έχει αντικατασταθεί από την βιβλιοθήκη JavaFX αλλά η Swing είναι πιο απλή για την εισαγωγή εννοιών.

Event driven programming

- Το Swing ακολουθεί το μοντέλο του **event-driven programming**
 - Υπάρχουν κάποια αντικείμενα που **πυροδοτούν συμβάντα** (firing an event)
 - Υπάρχουν κάποια άλλα αντικείμενα που είναι **ακροατές (listeners)** για συμβάντα.
 - Αν προκληθεί ένα συμβάν υπάρχουν ειδικοί **χειριστές** του συμβάντος (**event handlers**) – μέθοδοι που χειρίζονται ένα συμβάν
 - Το **συμβάν (event)** είναι κι αυτό ένα αντικείμενο το οποίο **μεταφέρει πληροφορία** μεταξύ του αντικειμένου που προκαλεί το συμβάν και του ακροατή.
- Σας θυμίζουν κάτι όλα αυτά?
 - Πολύ παρόμοιες αρχές υπάρχουν στην δημιουργία και τον χειρισμό **εξαιρέσεων**.

Swing

- Στην Swing βιβλιοθήκη ένα GUI αποτελείται από πολλά στοιχεία/συστατικά (**components**)
 - π.χ. παράθυρα, κουμπιά, μενού, κουτιά εισαγωγής κειμένου, κλπ.
- Τα components αυτά **πυροδοτούν συμβάντα**
 - Π.χ. το πάτημα ενός κουμπιού, η εισαγωγή κειμένου, η επιλογή σε ένα μενού, κλπ
- Τα συμβάντα αυτά τα χειρίζονται τα **αντικείμενα-ακροατές**, που έχουν ειδικές μεθόδους γι αυτά
 - Τι γίνεται όταν πατάμε ένα κουμπί, όταν κάνουμε μια επιλογή κλπ
- Όλο το πρόγραμμα κυλάει ως μια αλληλουχία από **συμβάντα** και τον **χειρισμό** τους από τους ακροατές.



JFrame

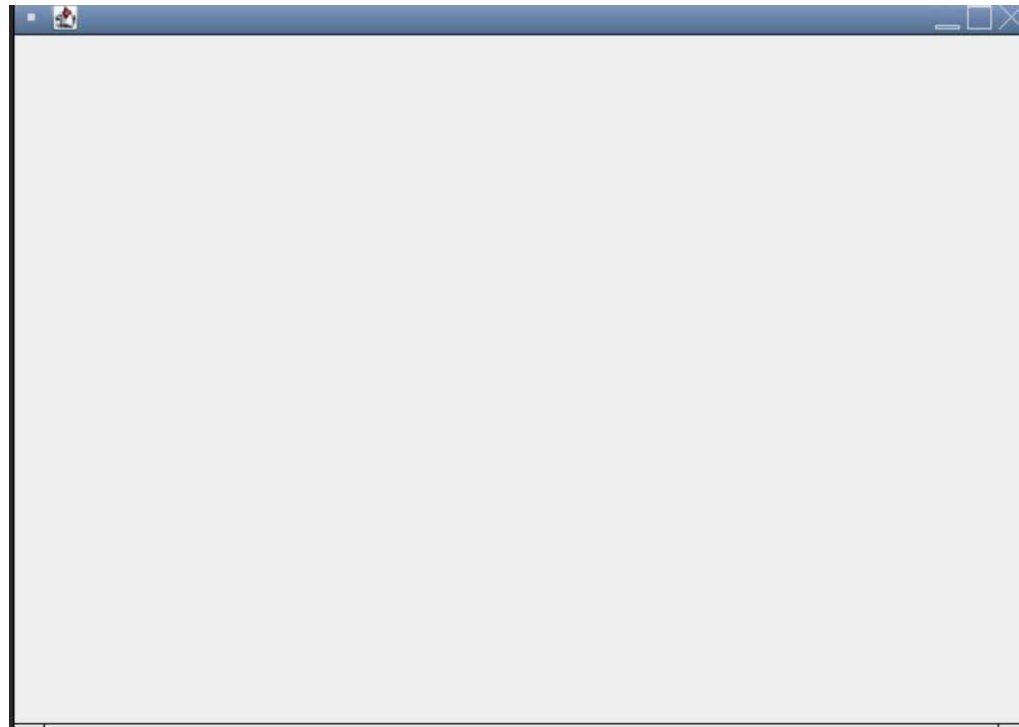
Το JFrame ορίζει ένα βασικό απλό παράθυρο.
Ο παρακάτω κώδικας δημιουργεί ένα παράθυρο

```
import javax.swing.JFrame;  
  
public class JFrameDemo  
{  
    public static final int WIDTH = 300;  
    public static final int HEIGHT = 200;  
  
    public static void main(String[] args)  
    {  
        JFrame firstWindow = new JFrame( );  
        firstWindow.setSize(WIDTH, HEIGHT);  
  
        firstWindow.setDefaultCloseOperation(  
            JFrame.EXIT_ON_CLOSE);  
  
        firstWindow.setVisible(true);  
    }  
}
```

Καθορίζει το μέγεθος
(πλάτος, ύψος) του
παραθύρου μετρημένο σε
pixels

Κάνει το παράθυρο ορατό

Καθορίζει τι κάνει το
παράθυρο όταν πατάμε
το κουμπί για κλείσιμο



JFrame

- Επιλογές για το `setDefaultCloseOperation`:
 - `EXIT_ON_CLOSE`: Καλεί την `System.exit()` και σταματάει το πρόγραμμα.
 - `DO_NOTHING_ON_CLOSE`: δεν κάνει τίποτα, ουσιαστικά δεν μας επιτρέπει να κλείσουμε το παράθυρο
 - `HIDE_ON_CLOSE`: Κρύβει το παράθυρο αλλά δεν σταματάει το πρόγραμμα.
- Άλλες μέθοδοι:
 - `add`: προσθέτει ένα συστατικό (component) στο παράθυρο (π.χ. ένα κουμπί)
 - `setTitle(String)`: δίνει ένα όνομα στο παράθυρο που δημιουργούμε.

ΕΤΙΚΕΤΕΣ

- Αφού έχουμε φτιάξει το βασικό παράθυρο μπορούμε πλέον να αρχίσουμε να προσθέτουμε συστατικά (**components**)
- Μπορούμε να προσθέσουμε ένα (σύντομο) κείμενο στο παράθυρο μας προσθέτοντας μια **ετικέτα (label)**
- **JLabel** class: μας επιτρέπει να δημιουργήσουμε μια ετικέτα με συγκεκριμένο κείμενο
 - `JLabel greeting = new JLabel("Hello World!");`
- Αφού δημιουργήσουμε την ετικέτα θα πρέπει να την προσθέσουμε μέσα στο παράθυρο μας.
 - Καλούμε την μέθοδο **add** της **JFrame**

Παράθυρο με ετικέτα

```
import javax.swing.JFrame;
import javax.swing.JLabel;

public class JLabelDemo
{
    public static final int WIDTH = 300;
    public static final int HEIGHT = 200;

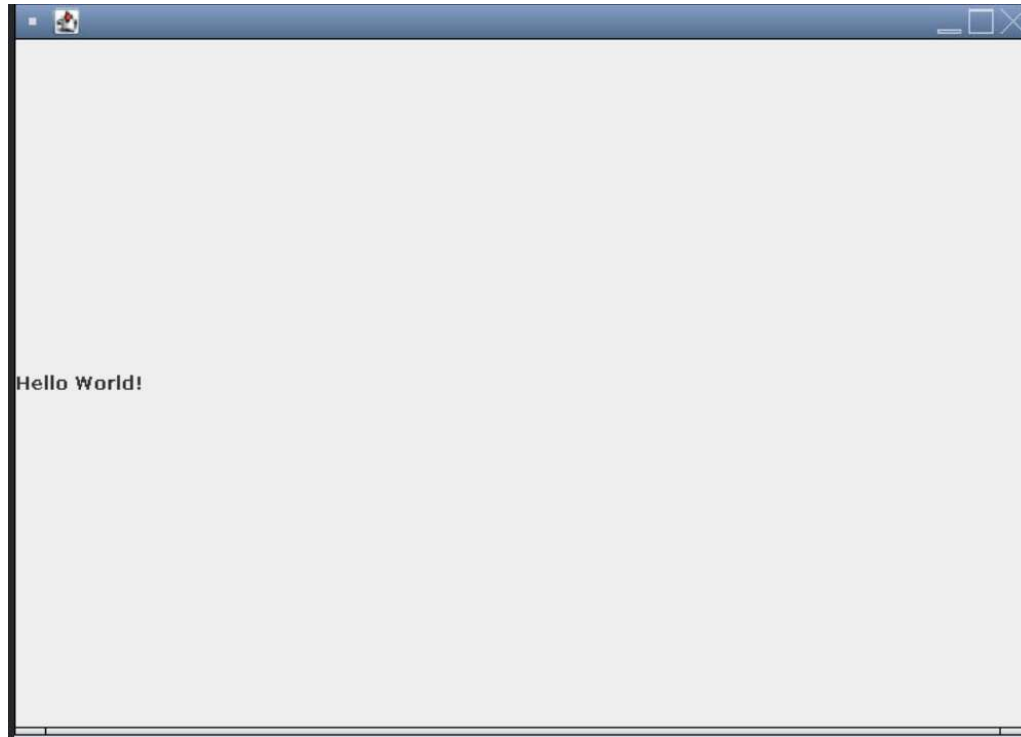
    public static void main(String[] args)
    {
        JFrame firstWindow = new JFrame( );
        firstWindow.setSize(WIDTH, HEIGHT);

        firstWindow.setDefaultCloseOperation(
            JFrame.EXIT_ON_CLOSE);

        JLabel label = new JLabel("Hello World!");
        firstWindow.add(label);

        firstWindow.setVisible(true);
    }
}
```

Δημιουργία της ετικέτας με την κλάση
JLabel και προσθήκη στο παράθυρο



Κουμπιά

- Ένα άλλο component για ένα γραφικό περιβάλλον είναι τα κουμπιά.
- Δημιουργούμε κουμπιά με την κλάση **JButton**.
 - `JButton button = new JButton("click me");`
 - Το κείμενο στον constructor είναι αυτό που εμφανίζεται πάνω στο κουμπί.
- Για να ξέρουμε τι κάνει το κουμπί όταν πατηθεί θα πρέπει να συνδέσουμε το κουμπί με ένα ακροατή.
 - Ο ακροατής είναι ένα αντικείμενο μιας κλάσης που υλοποιεί το **interface ActionListener** η οποία έχει την μέθοδο
 - `actionPerformed(ActionEvent e)`: χειρίζεται ένα συμβάν
 - Αφού δημιουργήσουμε το αντικείμενο του ακροατή το συνδέουμε (καταχωρούμε) με το κουμπί χρησιμοποιώντας την μέθοδο της **JButton**:
 - `addActionListener(ActionListener)`

```
import javax.swing.JFrame;
import javax.swing.JButton;

public class ButtonDemo
{
    public static final int WIDTH = 300;
    public static final int HEIGHT = 200;

    public static void main(String[] args)
    {
        JFrame firstWindow = new JFrame( );
        firstWindow.setSize(WIDTH, HEIGHT);

        firstWindow.setDefaultCloseOperation(
            JFrame.DO_NOTHING_ON_CLOSE);

        JButton endButton = new JButton("Click to end program.");

        EndingListener buttonEar = new EndingListener( );
        endButton.addActionListener(buttonEar);

        firstWindow.add(endButton);

        firstWindow.setVisible(true);
    }
}
```

Παράθυρο με κουμπί

Δημιουργία κουμπιού με
την κλάση **JButton**

Δημιουργία και **καταχώριση**
του ακροατή στο κουμπί

Προσθήκη κουμπιού
στο παράθυρο

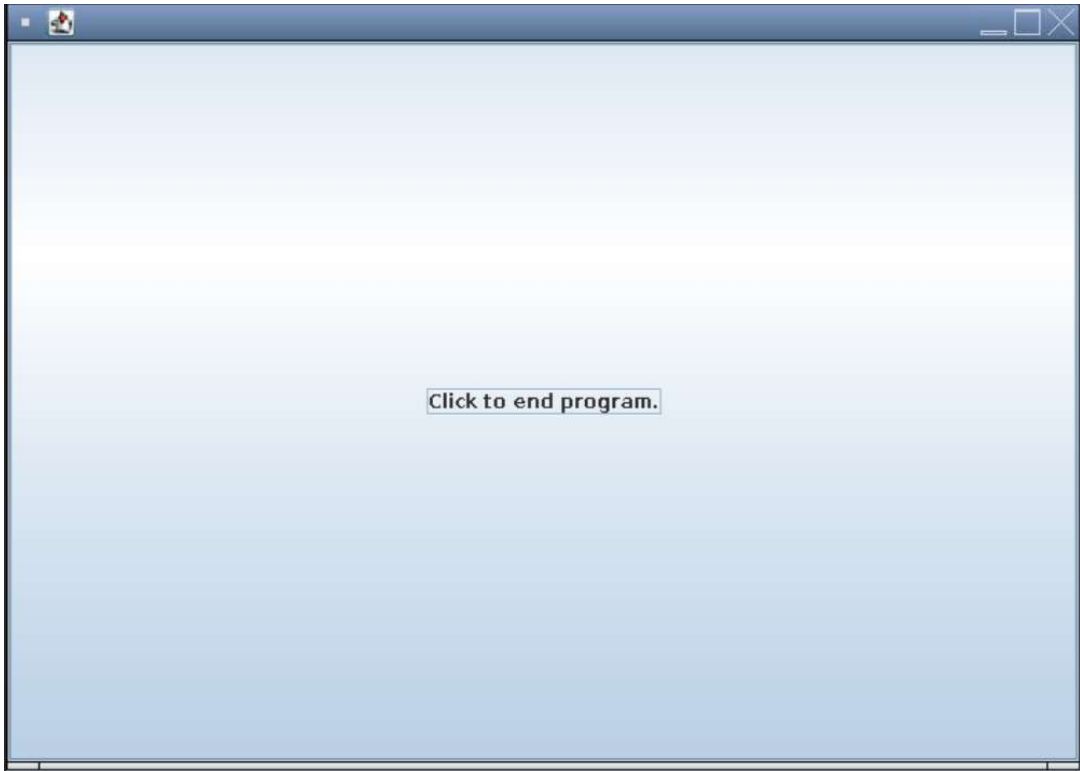
```
import java.awt.event.ActionListener;  
import java.awt.event.ActionEvent;  
  
public class EndingListener implements ActionListener  
{  
    public void actionPerformed(ActionEvent e)  
    {  
        System.exit(0);  
    }  
}
```

Ένας ακροατής υλοποιεί το **interface ActionListener** και πρέπει να υλοποιεί την μέθοδο **actionPerformed(ActionEvent)**

Όταν πατάμε το κουμπί στο GUI καλείται η μέθοδος **actionPerformed** του ακροατή που έχουμε καταχωρίσει για το κουμπί

Η κλήση της **actionPerformed** από τον **ActionListener** γίνεται αυτόματα μέσω της βιβλιοθήκης **Swing**, δεν την κάνει ο προγραμματιστής

Η παράμετρος **ActionEvent** περιέχει πληροφορία σχετικά με το συμβάν που μπορεί να χρησιμοποιηθεί.



```
import javax.swing.JFrame;  
import javax.swing.JButton;
```

Πιο σωστός τρόπος να ορίσουμε το παράθυρο μας ως ένα τύπο παράθυρου που επεκτείνει την κλάση JFrame

```
public class FirstWindow extends JFrame
```

```
{  
    public static final int WIDTH = 300;  
    public static final int HEIGHT = 200;  
  
    public FirstWindow( )  
    {  
        super( );  
        setSize(WIDTH, HEIGHT);  
  
        setTitle("First Window Class");  
  
        setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);  
  
        JButton endButton = new JButton("Click to end program.");  
        endButton.addActionListener(new EndingListener( ));  
        add(endButton);  
    }  
}
```

Η δημιουργία του ActionListener γίνεται ως ανώνυμο αντικείμενο μιας και δεν θα το χρησιμοποιήσουμε ποτέ άμεσα

```
public class DemoButtonWindow
{
    public static void main(String[] args)
    {
        FirstWindow w = new FirstWindow( );
        w.setVisible(true);
    }
}
```

Εδώ δημιουργούμε το παράθυρο μας

Αυτό είναι και το σωστό σημείο να αποφασίσουμε αν το παράθυρο θα είναι visible ή όχι.

Πολλά συστατικά

- Αν θέλουμε να βάλουμε **πολλά** components μέσα στο παράθυρο μας τότε θα πρέπει να προσδιορίσουμε **που** θα τοποθετηθούν αλλιώς θα μπουύνε το ένα πάνω στο άλλο.
- Αυτό γίνεται με την μέθοδο **setLayout** που καθορίζει την τοποθέτηση μέσα στο παράθυρο
 - Αυτό μπορεί να γίνει με διαφορετικούς τρόπους

FlowLayout

- Απλά τοποθετεί τα components το ένα μετά το άλλο από τα αριστερά προς τα δεξιά
- Καλούμε την μέθοδο

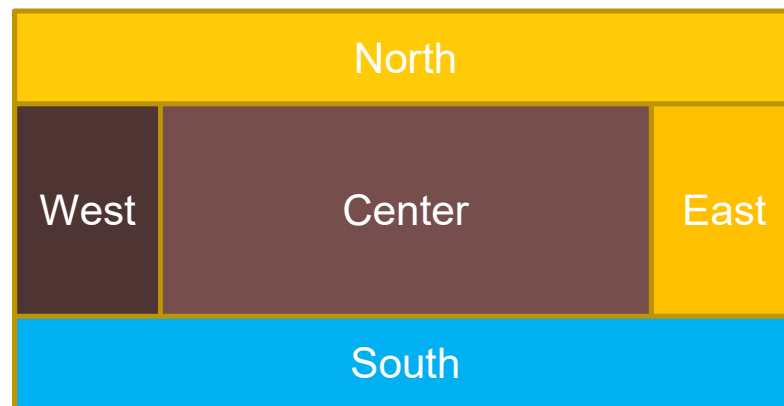
```
setLayout(new FlowLayout());
```

(Πρέπει να έχουμε κάνει `import java.awt.FlowLayout`)

- Μετά προσθέτουμε κανονικά τα components με την `add`.

BorderLayout

- Στην περίπτωση αυτή ο χώρος χωρίζεται σε πέντε περιοχές: North, South, East, West Center
- Καλούμε την μέθοδο
`setLayout(new BorderLayout());`
(Πρέπει να έχουμε κάνει `import java.awt.BorderLayout`)
- Μετά όταν προσθέτουμε τα components με την `add`, προσδιορίζουμε την περιοχή στην οποία θα προστεθούν.
 - Π.χ., `add(label, BorderLayout.CENTER)`



GridLayout

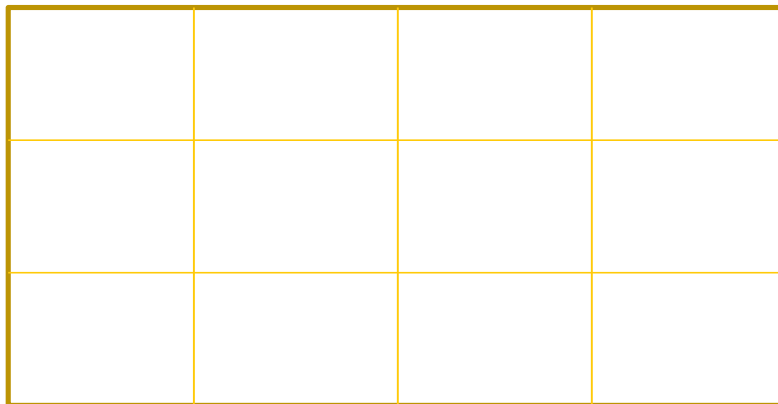
- Στην περίπτωση αυτή ορίζουμε ένα πλέγμα με n γραμμές και m στήλες και αυτό γεμίζει από τα αριστερά προς τα δεξιά και από πάνω προς τα κάτω
- Καλούμε την εντολή

```
setLayout(new GridLayout(n,m));
```

(Πρέπει να έχουμε κάνει `include java.awt.GridLayout`)

- Μετά προσθέτουμε κανονικά τα components με την `add`.

Grid 3x4



Παράδειγμα

- Δημιουργείστε ένα παράθυρο με τρία κουμπιά:
 - Το ένα κάνει το χρώμα του παραθύρου μπλε, το άλλο κόκκινο και το τρίτο κλείνει το παράθυρο.
 - Κώδικας: `MultiButtonWindow`

```
import javax.swing.JFrame;
import javax.swing.JButton;
import javax.swing.JLabel;
import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
```

Η κλάση υλοποιεί τον ακροατή
και την `actionPerformed` μεθοδο

```
public class MultiButtonWindow extends JFrame implements ActionListener
{
    public static final int WIDTH = 300;
    public static final int HEIGHT = 200;

    public MultiButtonWindow( )
    {
        super( "Multi-Color" );
        setSize(WIDTH, HEIGHT);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        setLayout(new FlowLayout());

        JLabel label = new JLabel("Pick A Color");
        add(label);

        JButton blueButton = new JButton("Blue");
        blueButton.addActionListener(this);
        add(blueButton);

        JButton redButton = new JButton("Red");
        redButton.addActionListener(this);
        add(redButton);

        JButton endButton = new JButton("Exit");
        endButton.addActionListener(this);
        add(endButton);
    }
}
```

Ορίζουμε τα χαρακτηριστικά
του βασικού παραθύρου

Δημιουργούμε τα τρία
κουμπιά και τα
προσθέτουμε στο frame

Ο ακροατής των
κουμπιών είναι το **ίδιο** το
αντικείμενο (**this**)

Συνέχεια στην
επόμενη

Συνέχεια από
την προηγούμενη

```
public void actionPerformed(ActionEvent e)
{
    String buttonType = e.getActionCommand( );

    switch (buttonType) {
        case "Blue":
            getContentPane().setBackground(Color.BLUE);
            break;
        case "Red":
            getContentPane().setBackground(Color.RED);
            break;
        case "Exit":
            System.exit(0);
    }
}
```

Το
αποτέλεσμα
του κάθε
διαφορετικού
κουμπιού.

Η μέθοδος `actionPerformed`
που καλείται όταν πατηθούν τα
κουμπιά (μιας και το
αντικείμενο είναι και ακροατής)

Επιστρέφει το
`actionCommand`
String, το οποίο αν
δεν το έχουμε
αλλάξει είναι το
όνομα του κουμπιού

Η `getContentPane` μας δίνει πρόσβαση στα
χαρακτηριστικά του frame.
Η `setBackground` αλλάζει το χρώμα του frame

```
public static void main(String[] args)
{
    MultiButtonWindow w = new MultiButtonWindow();
    w.setVisible(true);
}
}
```

Δημιουργία του παραθύρου





Αξιοσημείωτα

- `public class MultiButtonWindow`
 - `extends JFrame`
 - `implements ActionListener`
- Μπορούμε να κάνουμε τον ακροατή να είναι το ίδιο το παράθυρο, αυτό θα αναλάβει να υλοποιήσει τη μέθοδο `actionPerformed`.
- Όταν καταχωρούμε τον ακροατή:
`blueButton.addActionListener(this);`
- `getContentPane().setBackground(Color.BLUE);`
 - Αλλάζει το background χρώμα του παραθύρου. Η κλάση `Color` μας δίνει τα χρώματα
- `String buttonType = e.getActionCommand();`
 - Με την εντολή αυτή παίρνουμε το `String` το οποίο δώσαμε σαν τίτλο στο κουμπί

actionCommand

- Ένα String πεδίο που κρατάει πληροφορία για το συμβάν
 - Αν δεν αλλάξουμε κάτι αυτό είναι το όνομα του κουμπιού
- Μπορούμε να διαβάσουμε το String με την εντολή `getActionCommand`.
- Μπορούμε να θέσουμε μια τιμή στο String με την εντολή `setActionCommand(String)`
- Π.χ.
`redButton.setActionCommand("RedButtonClick");`

Χρώματα

- Μπορούμε να ορίσουμε τα δικά μας χρώματα με την RGB σύμβαση
 - `Color myColor = new Color(200,100,4) ;`
 - Τα ορίσματα είναι οι RGB (Red, Green, Blue) τιμές

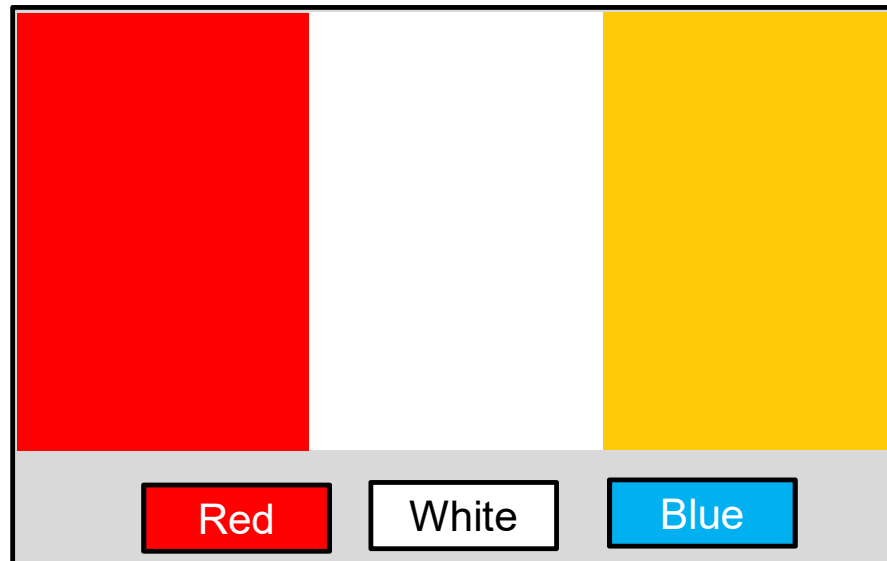
JPanel

- Το **panel** (τομέας) είναι ένας **container**
 - Μέσα σε ένα container μπορούμε να βάλουμε components και να ορίσουμε χειρισμό συμβάντων.
- Τα panels κατά μία έννοια ορίζουν ένα **παράθυρο μέσα στο παράθυρο**
 - Το panel έχει κι αυτό το δικό του layout και τοποθετούμε μέσα σε αυτό συστατικά.
 - Π.χ., ο παρακάτω κώδικας εκτελείται μέσα σε ένα JFrame.

```
setLayout(new BorderLayout());  
  
JPanel buttonPanel = new JPanel();  
buttonPanel.setLayout(new FlowLayout());  
  
JButton button1 = new JButton("one");  
buttonPanel.add(button1);  
  
JButton button2 = new JButton("two");  
buttonPanel.add(button2);  
  
add(buttonPanel, BorderLayout.SOUTH);
```

Παράδειγμα

- Θα δημιουργήσουμε ένα παράθυρο με τρία panels το κάθε panel θα παίρνει διαφορετικό χρώμα με ένα διαφορετικό κουμπί.



```
import javax.swing.JFrame;
import javax.swing.JPanel;
import java.awt.BorderLayout;
import java.awt.GridLayout;
import java.awt.FlowLayout;
import java.awt.Color;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
```

Η κλάση υλοποιεί τον ακροατή και την actionPerformed μεθοδο

```
public class PanelDemo extends JFrame implements ActionListener
{
    public static final int WIDTH = 300;
    public static final int HEIGHT = 200;

    private JPanel redPanel;
    private JPanel whitePanel;
    private JPanel bluePanel;
```

Δηλώνουμε τα τρία πάνελ με τα τρία χρώματα

```
public PanelDemo( )
{
    super("Panel Demonstration");
    setSize(WIDTH, HEIGHT);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setLayout(new BorderLayout( ));
```

Ορίζουμε τα χαρακτηριστικά του βασικού παραθύρου

Συνέχεια στην επόμενη

Συνέχεια από
την προηγούμενη

Δημιουργούμε ένα μεγάλο πάνελ
που θα κρατάει τα τρία
χρωματιστά πάνελ

```
JPanel biggerPanel = new JPanel( );  
biggerPanel.setLayout(new GridLayout(1, 3));  
  
redPanel = new JPanel( );  
redPanel.setBackground(Color.LIGHT_GRAY);  
biggerPanel.add(redPanel);  
  
whitePanel = new JPanel( );  
whitePanel.setBackground(Color.LIGHT_GRAY);  
biggerPanel.add(whitePanel);  
  
bluePanel = new JPanel( );  
bluePanel.setBackground(Color.LIGHT_GRAY);  
biggerPanel.add(bluePanel);  
  
add(biggerPanel, BorderLayout.CENTER);
```

Δημιουργούμε τα
χρωματιστά
πάνελ και τα
προσθέτουμε
στο μεγάλο
πάνελ

Συνέχεια στην
επόμενη

Βάζουμε το μεγάλο πάνελ
στο κέντρο του παραθύρου

Συνέχεια από
την προηγούμενη

Δημιουργούμε ένα πάνελ που θα
κρατάει τα τρία κουμπιά

```
JPanel buttonPanel = new JPanel( );  
buttonPanel.setBackground(Color.LIGHT_GRAY);  
buttonPanel.setLayout(new FlowLayout( ));
```

```
JButton redButton = new JButton("Red");  
redButton.setBackground(Color.RED);  
redButton.addActionListener(this);  
buttonPanel.add(redButton);
```

```
JButton whiteButton = new JButton("White");  
whiteButton.setBackground(Color.WHITE);  
whiteButton.addActionListener(this);  
buttonPanel.add(whiteButton);
```

```
JButton blueButton = new JButton("Blue");  
blueButton.setBackground(Color.BLUE);  
blueButton.addActionListener(this);  
buttonPanel.add(blueButton);
```

```
add(buttonPanel, BorderLayout.SOUTH);
```

```
} // τέλος του constructor
```

Δημιουργούμε τα
τρία κουμπιά και
τα προσθέτουμε
στο πάνελ

Ο ακροατής των
κουμπιών είναι
το ίδιο το
αντικείμενο

Βάζουμε το πάνελ με τα κουμπιά
στον πάτο του παραθύρου

Συνέχεια στην
επόμενη

Συνέχεια από
την προηγούμενη

```
public void actionPerformed(ActionEvent e)
{
    String buttonString = e.getActionCommand( );

    if (buttonString.equals("Red"))
        redPanel.setBackground(Color.RED);
    else if (buttonString.equals("White"))
        whitePanel.setBackground(Color.WHITE);
    else if (buttonString.equals("Blue"))
        bluePanel.setBackground(Color.BLUE);
    else
        System.out.println("Unexpected error.");
}

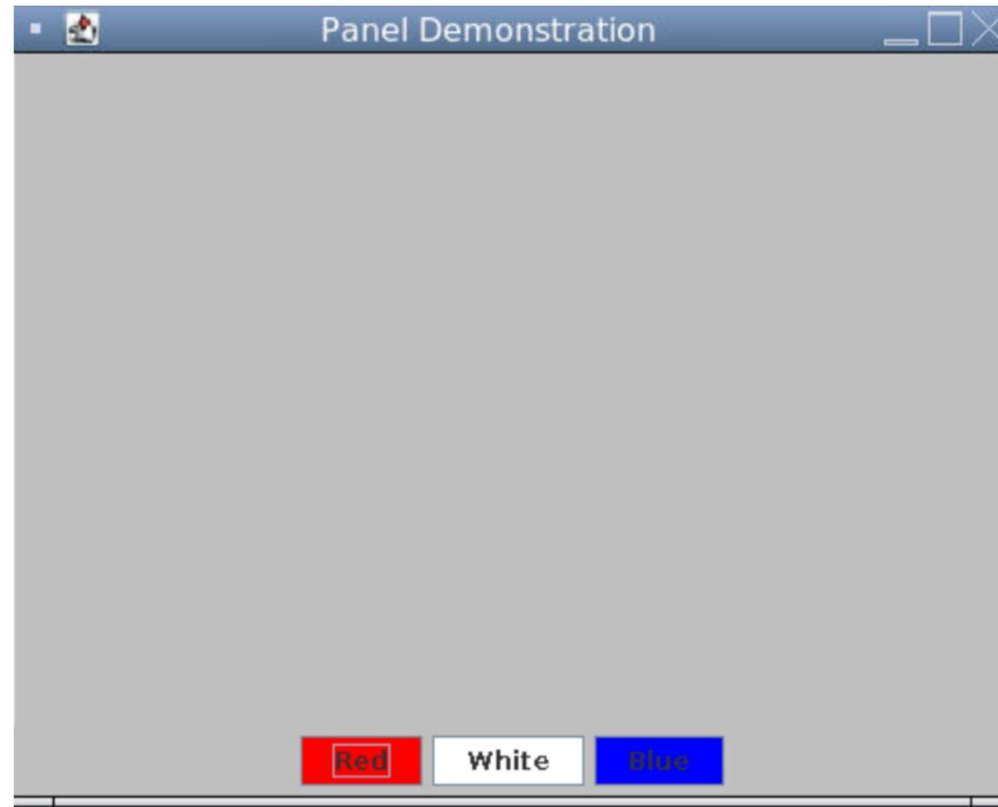
public static void main(String[] args)
{
    PanelDemo gui = new PanelDemo( );
    gui.setVisible(true);
}
}
```

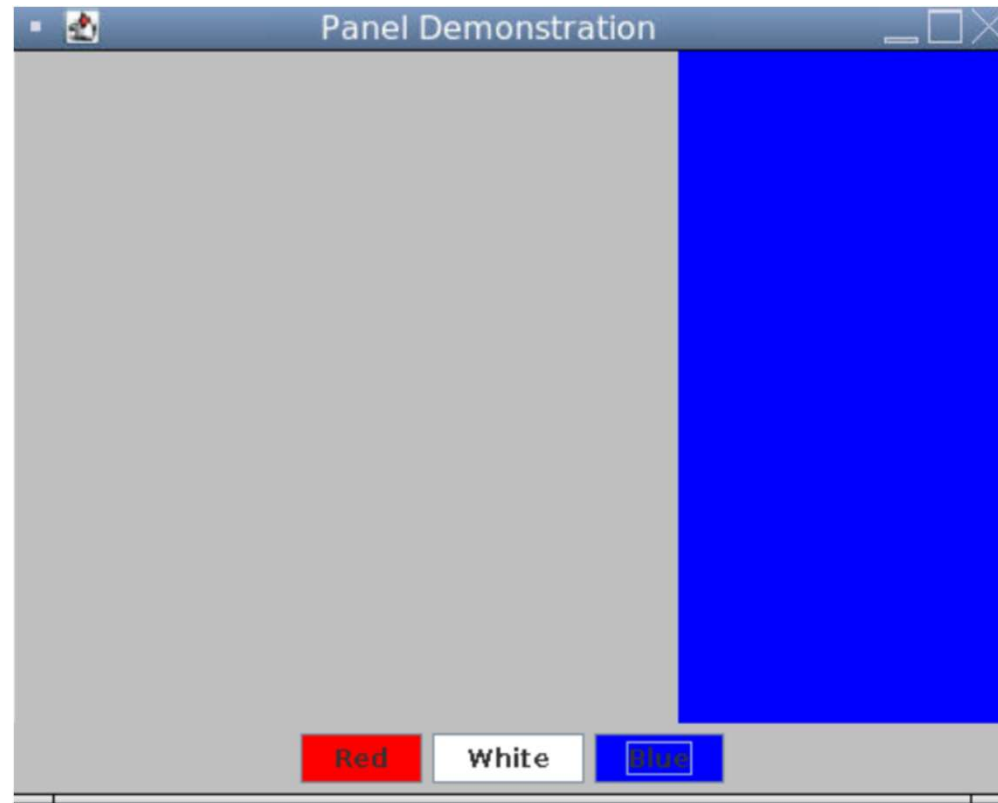
Η συνάρτηση `actionPerformed` που καλείται όταν πατηθούν τα κουμπιά (μιας και το αντικείμενο είναι και ακροατής)

Επιστρέφει το `actionCommand String`, το οποίο αν δεν το έχουμε αλλάξει είναι το όνομα του κουμπιού

Το αποτέλεσμα του κάθε διαφορετικού κουμπιού.

Δημιουργία του παραθύρου





Menu

- Drop-down menus:
 - **JMenuItem**: κρατάει μία από τις επιλογές του menu
 - **JMenu**: κρατάει όλα τα JMenuItem
 - **JMenuBar**: κρατάει το Jmenu
 - **setJMenuBar (JMenu)** : θέτει το menu στην κορυφή του JFrame.
Μπορούμε να χρησιμοποιήσουμε και τη γνωστή εντολή **add**.

Παράδειγμα

Δημιουργεί ένα drop-down menu

Δημιουργεί τις επιλογές του μενού και τις προσθέτει στο μενού

```
JMenu colorMenu = new JMenu("Add Colors");

JMenuItem redChoice = new JMenuItem("Red");
redChoice.addActionListener(this);
colorMenu.add(redChoice);

JMenuItem whiteChoice = new JMenuItem("White");
whiteChoice.addActionListener(this);
colorMenu.add(whiteChoice);

JMenuItem blueChoice = new JMenuItem("Blue");
blueChoice.addActionListener(this);
colorMenu.add(blueChoice);

JMenuBar bar = new JMenuBar();
bar.add(colorMenu);
setJMenuBar(bar);
```

Δημιουργεί ένα menu bar στην κορυφή του παραθύρου και προσθέτει το menu σε αυτό

ΠΑΡΑΔΕΙΓΜΑ

```
import javax.swing.*;
class MenuExample
{
    JMenuBar mb;
    JMenu menu, submenu;
    JMenuItem i1, i2, i3, i4, i5;

    public MenuExample(){
        JFrame f= new JFrame("Menu and
        JMenuItem Example");
        mb=new JMenuBar();
        menu=new JMenu("Menu");
        submenu=new JMenu("Sub Menu");

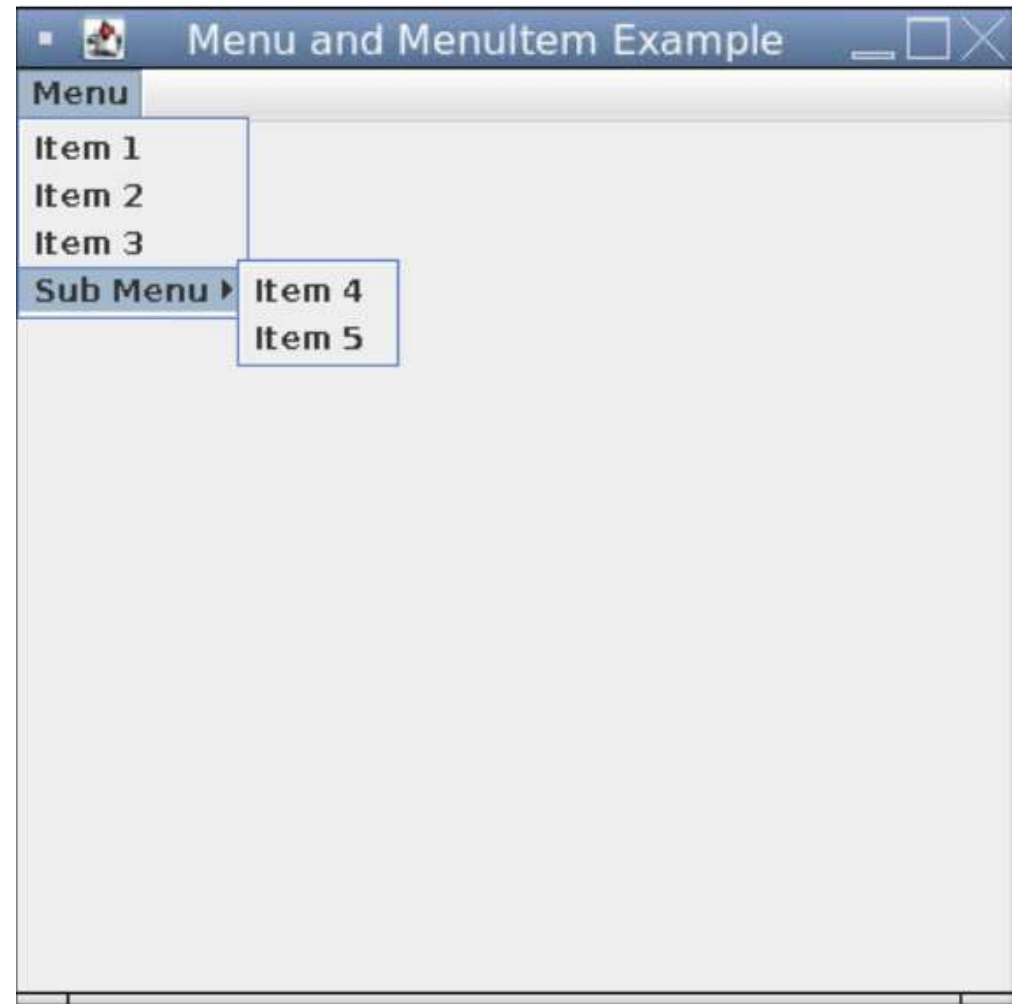
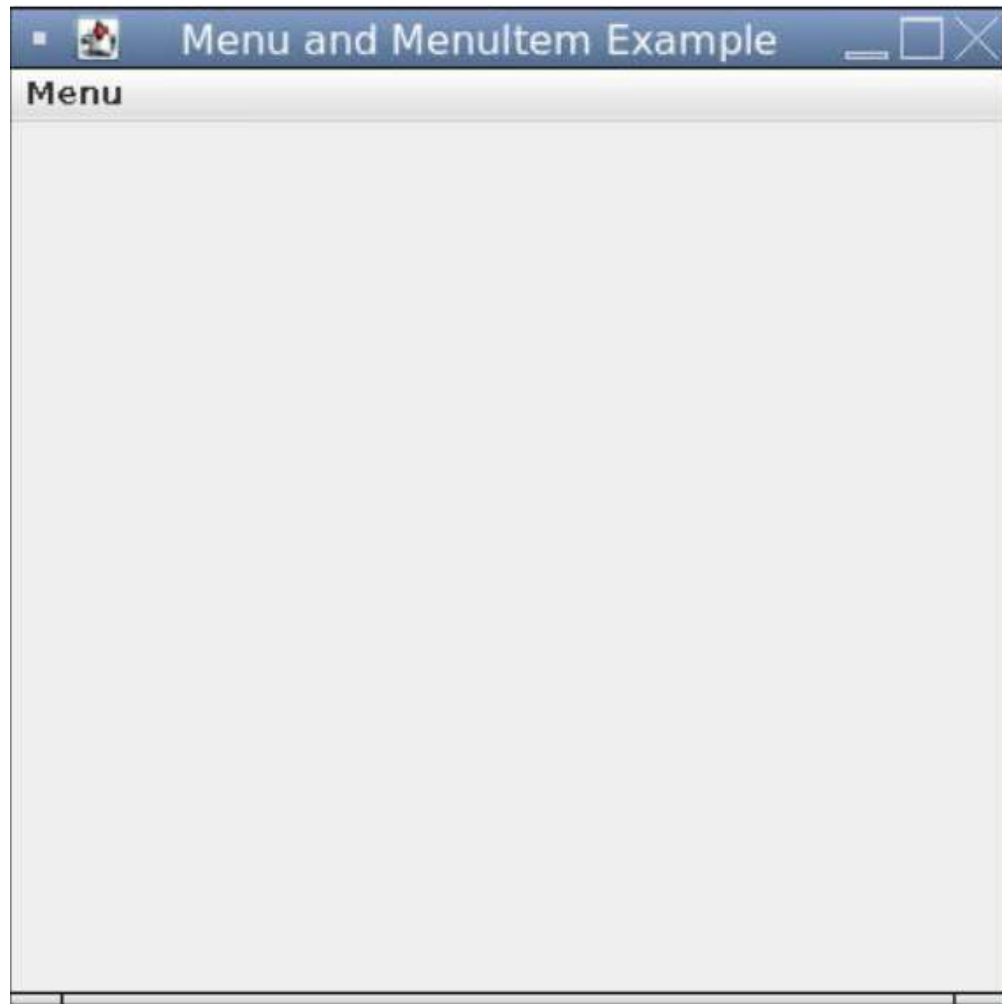
        i1=new JMenuItem("Item 1");
        i2=new JMenuItem("Item 2");
        i3=new JMenuItem("Item 3");
        i4=new JMenuItem("Item 4");
        i5=new JMenuItem("Item 5");

        menu.add(i1); menu.add(i2); menu.add(i3);

        submenu.add(i4); submenu.add(i5);

        menu.add(submenu);
        mb.add(menu);
        f.setJMenuBar(mb);
        f.setSize(400,400);
        f.setLayout(null);
        f.setVisible(true);
    }

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



ΠΑΡΑΔΕΙΓΜΑ II

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

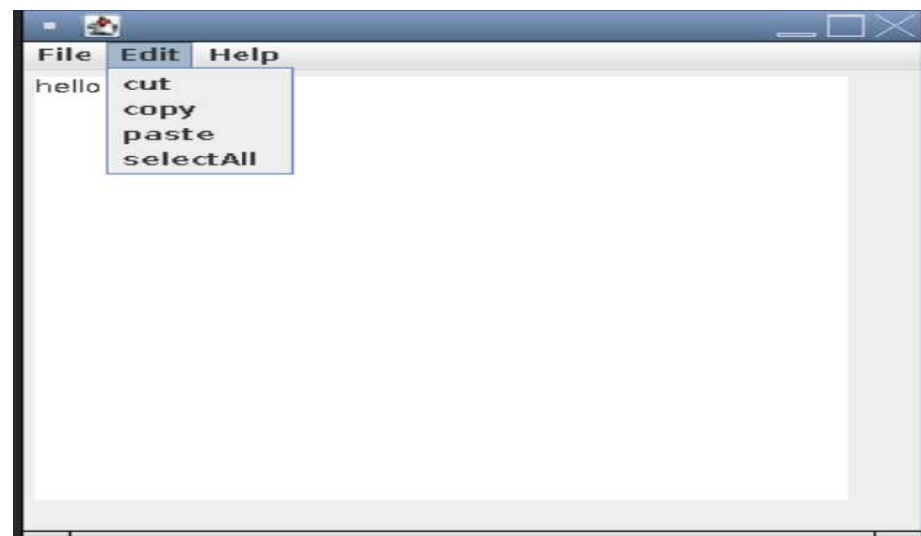
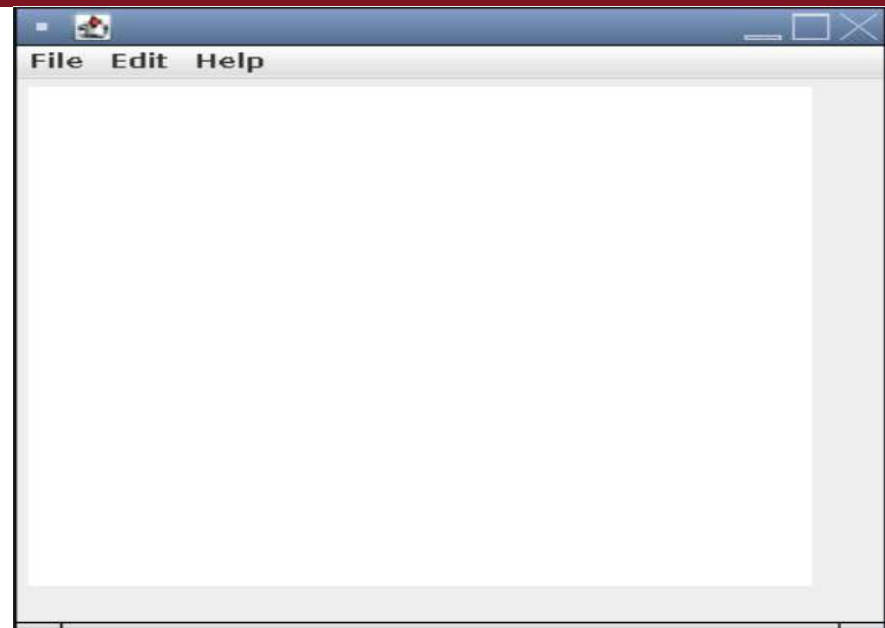
public class MenuExample implements ActionListener{
    JFrame f;
    JMenuBar mb;
    JMenu file,edit,help;
    JMenuItem cut,copy,paste,selectAll;
    JTextArea ta;

    public MenuExample(){
        f=new JFrame();
        cut=new JMenuItem("cut");
        copy=new JMenuItem("copy");
        paste=new JMenuItem("paste");
        selectAll=new JMenuItem("selectAll");
        cut.addActionListener(this);
        copy.addActionListener(this);
        paste.addActionListener(this);
        selectAll.addActionListener(this);
        mb=new JMenuBar();
        file=new JMenu("File");
        edit=new JMenu("Edit");
        help=new JMenu("Help");
```

```
        edit.add(cut);edit.add(copy);edit.add(paste);edit.add(selectAll);
        mb.add(file);mb.add(edit);mb.add(help);
        ta=new JTextArea();
        ta.setBounds(5,5,360,320);
        f.add(mb);f.add(ta);
        f.setJMenuBar(mb);
        f.setLayout(null);
        f.setSize(400,400);
        f.setVisible(true);
    }

    public void actionPerformed(ActionEvent e) {
        if(e.getSource()==cut)
            ta.cut();
        if(e.getSource()==paste)
            ta.paste();
        if(e.getSource()==copy)
            ta.copy();
        if(e.getSource()==selectAll)
            ta.selectAll();
    }

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



Text Box

- Μπορούμε να δημιουργήσουμε ένα πεδίο κειμένου με την κλάση **JTextField**.
 - Το JTextField δημιουργεί ένα **text box** μίας γραμμής
 - **getText()** : με την εντολή αυτή **διαβάζουμε** το κείμενο που δόθηκε σαν είσοδος στο text box.
 - **setText(String)** : με την εντολή αυτή **θέτουμε** το κείμενο στο text box.
- Για ένα πεδίο κειμένου μεγαλύτερο από μία γραμμή μπορούμε να χρησιμοποιήσουμε την κλάση **JTextArea**

Παράδειγμα

```
JTextField name = new JTextField(NUMBER_OF_CHAR);  
namePanel.add(name, BorderLayout.SOUTH);
```

```
JButton actionButton = new JButton("Click me");  
actionButton.addActionListener(this);  
buttonPanel.add(actionButton);
```

```
JButton clearButton = new JButton("Clear");  
clearButton.addActionListener(this);  
buttonPanel.add(clearButton);
```

```
public void actionPerformed(ActionEvent e)  
{  
    String actionCommand = e.getActionCommand( );  
  
    if (actionCommand.equals("Click me"))  
        name.setText("Hello " + name.getText( ));  
    else if (actionCommand.equals("Clear"))  
        name.setText("");  
    else  
        name.setText("Unexpected error.");  
}
```

ΠΑΡΑΔΕΙΓΜΑ

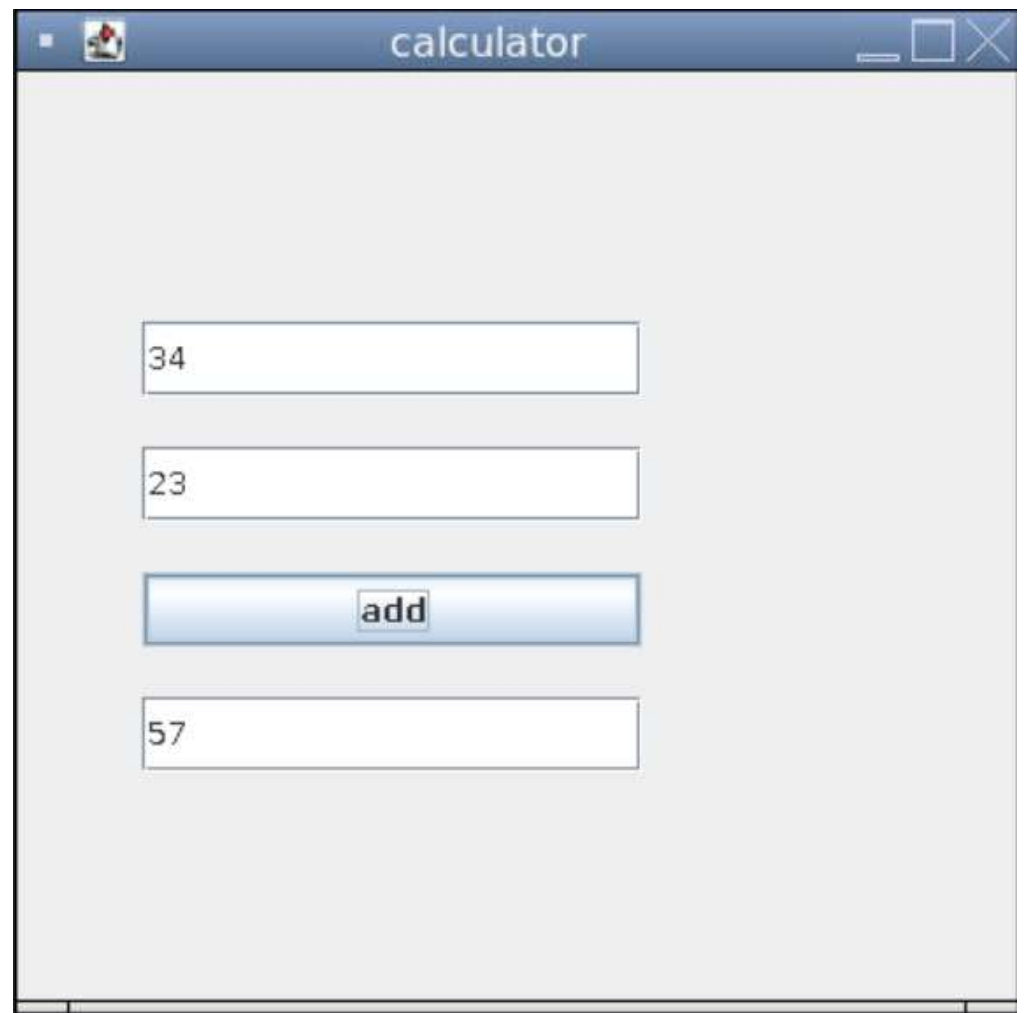
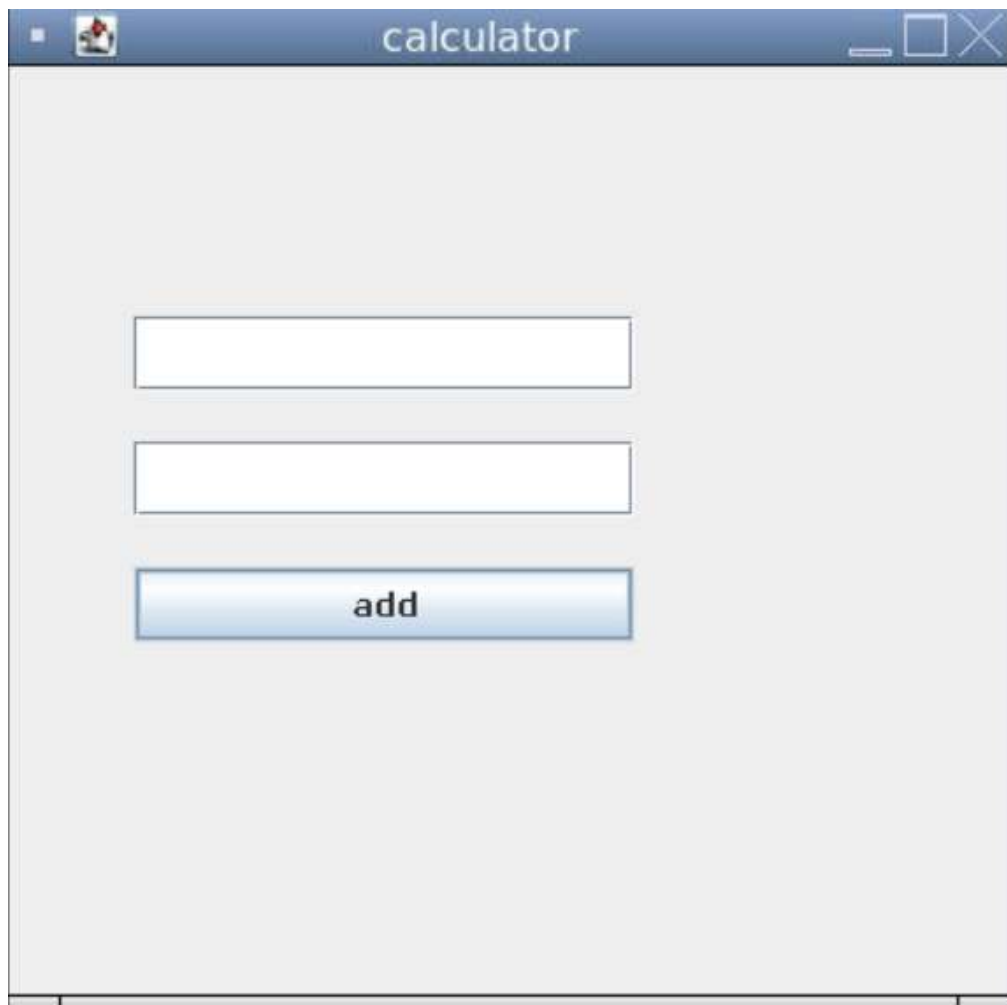
```
import javax.swing.*;
import java.awt.BorderLayout;
import java.awt.GridLayout;
import java.awt.FlowLayout;
import java.awt.Color;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

class Calc implements ActionListener
{ JFrame f;
  JTextField t1,t2,t3;
  JButton addButton;

  public Calc(){
    f= new JFrame("calculator");
    t1=new JTextField("");
    t1.setBounds(50,100, 200,30);
    t2=new JTextField("");
    t2.setBounds(50,150, 200,30);
    addButton = new JButton("add");
    addButton.addActionListener(this);
    addButton.setBounds(50,200, 200,30);
    f.add(t1); f.add(t2);
    f.add(addButton);
    f.setSize(400,400);
    f.setLayout(null);
    f.setVisible(true);      }
}
```

```
public void actionPerformed(ActionEvent e)
{Integer i=new
Integer(Integer.parseInt(t1.getText())+Integer.parseInt(t2.getText()))
);
JTextField t3=new JTextField(i.toString());
t3.setBounds(50,250,200,30);
f.add(t3);
f.setSize(400,400);
f.setLayout(null);
f.setVisible(true);
}
}

public static void main(String args[])
{
Main m=new Calc();
}
}
```



Pop-up Windows

- Αν θέλουμε να δημιουργήσουμε παράθυρα διαλόγου μπορούμε να χρησιμοποιήσουμε την κλάση `JOptionPane`
 - Πετάνει (pops up) ένα παράθυρο το οποίο μπορεί να μας ζητάει είσοδο, ή να ζητάει επιβεβαίωση.
 - Η δημιουργία και η διαχείριση των παραθύρων γίνεται με **στατικές μεθόδους**.

```
import javax.swing.JOptionPane;
```

```
public class PopUpDemo
```

```
{
```

```
    public static void main(String args[])
```

```
    {
        boolean done = false;
```

```
        while (!done){
```

```
            String classes =
```

```
                JOptionPane.showInputDialog("Enter number of classes");
```

```
            String students =
```

```
                JOptionPane.showInputDialog("Enter number of students");
```

```
            int totalStudents =
```

```
                Integer.parseInt(classes)*Integer.parseInt(students);
```

```
            JOptionPane.showMessageDialog(null,
                "Total number of students = "+totalStudents);
```

```
            int answer =
```

```
                JOptionPane.showConfirmDialog(null,
```

```
                "Continue?",
```

```
                "Confirm",
```

```
                JOptionPane.YES_NO_OPTION);
```

```
            done = (answer == JOptionPane.NO_OPTION);
```

```
        }
        System.exit(0);
```

```
    }
```

```
}
```

Εμφανίζει ένα παράθυρο διαλόγου που ζητάει από τον χρήστη να δώσει είσοδο. Η είσοδος αποθηκεύεται στο String που επιστρέφεται

Το αντικείμενο (component) που είναι πατέρας του pop-up, null η default τιμή

Εμφανίζει ένα παράθυρο που τυπώνει ένα μήνυμα

Η ερώτηση στο χρήστη

Τίτλος παραθύρου

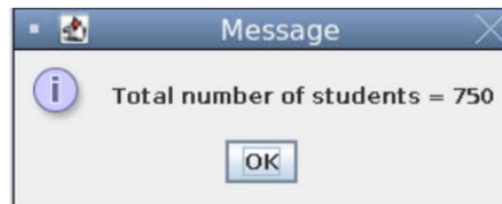
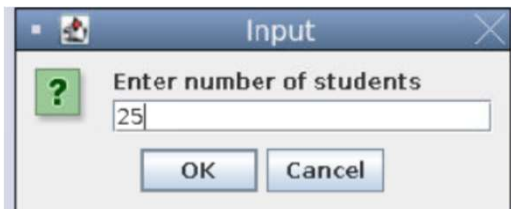
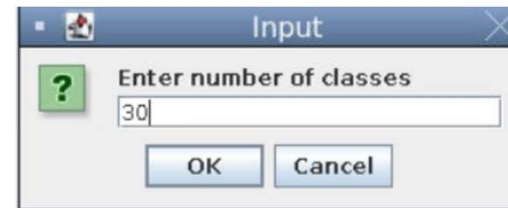
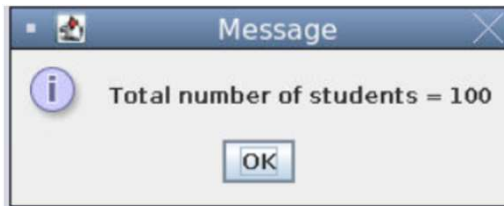
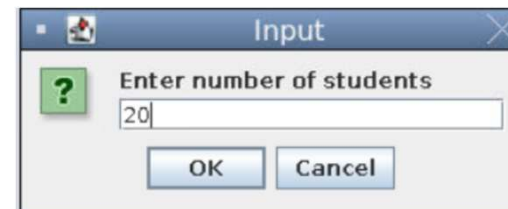
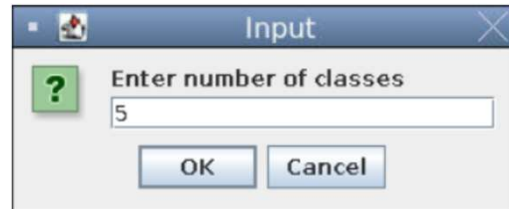
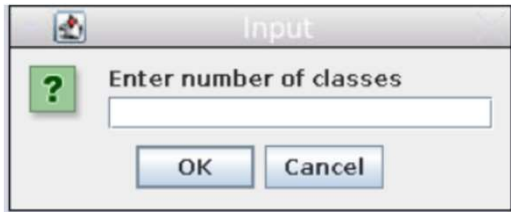
Εμφανίζει ένα παράθυρο επιβεβαίωσης

Τύπος επιβεβαίωσης

Σταθερά για την επιλογή (YES_OPTION για ΝΑΙ)

Άλλοι τύποι επιβεβαίωσης:

- OK_CANCEL_OPTION
- YES_NO_CANCEL_OPTION



Icons

- Μπορούμε να βάλουμε μέσα στο GUI μας και εικονίδια
- Παράδειγμα

Δημιουργεί ένα εικονίδιο από μία εικόνα

```
ImageIcon dukeIcon = new ImageIcon("duke_waving.gif");  
JLabel dukeLabel = new JLabel("Mood check");  
dukeLabel.setIcon(dukeIcon);
```

Προσθέτει το εικονίδιο σε ένα label

```
ImageIcon happyIcon = new ImageIcon("smiley.gif");  
JButton happyButton = new JButton("Happy");  
happyButton.setIcon(happyIcon);
```

Προσθέτει το εικονίδιο σε ένα button

Ακροατές

- Στο πρόγραμμα μας ορίσαμε την κλάση που δημιουργεί το παράθυρο (`extends JFrame`) να είναι και ο ακροατής (`implements ActionListener`) των συμβάντων μέσα στο παράθυρο.
 - Αυτό είναι μια βολική λύση γιατί όλος ο κώδικας είναι στο **ίδιο** σημείο
 - Έχει το πρόβλημα ότι έχουμε **μία μόνο** μέθοδο `actionPerformed` στην οποία θα πρέπει να ξεχωρίσουμε όλες τις περιπτώσεις.
- Πιο βολικό να έχουμε ένα **διαφορετικό** `ActionListener` για κάθε διαφορετικό συμβάν
 - Προβλήματα:
 - Θα πρέπει να ορίσουμε **πολλαπλές κλάσεις** ακροατών σε πολλαπλά αρχεία
 - Θα πρέπει να περνάμε σαν παραμέτρους τα στοιχεία που θέλουμε να αλλάξουμε.

Ακροατές

- **Λύση:** Να ορίσουμε τους ακροατές που χρειάζεται το παράθυρο μας ως **εσωτερικές κλάσεις**
- **Υπενθύμιση:** μια εσωτερική κλάση ορίζεται μέσα σε μία άλλη κλάση και την βλέπει μόνο η κλάση που την ορίζει
- **Πλεονεκτήματα:**
 - Οι κλάσεις είναι πλέον **τοπικές** στον κώδικα που τις καλεί, μπορούμε να επαναχρησιμοποιούμε τα ίδια ονόματα
 - Οι κλάσεις έχουν πρόσβαση σε **ιδιωτικά πεδία**

```
    JButton redButton = new JButton("Red");  
    redButton.setBackground(Color.RED);  
    redButton.addActionListener(new RedListener());  
    buttonPanel.add(redButton);
```

```
    JButton whiteButton = new JButton("White");  
    whiteButton.setBackground(Color.WHITE);  
    whiteButton.addActionListener(new WhiteListener());  
    buttonPanel.add(whiteButton);
```

```
    JButton blueButton = new JButton("Blue");  
    blueButton.setBackground(Color.BLUE);  
    blueButton.addActionListener(new BlueListener());  
    buttonPanel.add(blueButton);
```

Ορισμός των εσωτερικών κλάσεων-ακροατών
(Κώδικας στο InnerListeners.java)

```
private class RedListener implements ActionListener{  
    public void actionPerformed(ActionEvent e)  
    {  
        redPanel.setBackground(Color.RED);  
    }  
}
```

```
private class WhiteListener implements ActionListener{  
    public void actionPerformed(ActionEvent e)  
    {  
        whitePanel.setBackground(Color.WHITE);  
    }  
}
```

```
private class BlueListener implements ActionListener{  
    public void actionPerformed(ActionEvent e)  
    {  
        bluePanel.setBackground(Color.BLUE);  
    }  
}
```

Οι εσωτερικές κλάσεις έχουν πρόσβαση στα
ιδιωτικά αντικείμενα πάνελ

Ανώνυμες κλάσεις

- Τα αντικείμενα-ακροατές είναι **ανώνυμα** αντικείμενα
 - `redButton.addActionListener(new RedListener());`
- Μπορούμε να κάνουμε τον κώδικα ακόμη πιο συνοπτικό ορίζοντας μια **ανώνυμη κλάση**
 - Ο ορισμός της κλάσης γίνεται εκεί που τον χρειαζόμαστε μόνο και **υλοποιεί ένα Interface**
 - Δεν συνίσταται αλλά μπορεί να το συναντήσετε σε κώδικα που δημιουργείται από IDEs

```
redButton.addActionListener(new ActionListener()  
{  
    public void actionPerformed(ActionEvent e) {  
        redPanel.setBackground(Color.RED);  
    }  
});
```

Ο ορισμός της κλάσης
Χρησιμοποιούμε το **όνομα του interface**

JCheckBoxMenuItem

Constructor	Περιγραφή
JCheckBoxMenuItem()	Δημιουργεί στο μενού με μη επιλεγμένο με check box χωρίς καθορισμένο κείμενο ή εικονίδιο.
JCheckBoxMenuItem(Action a)	Δημιουργεί στο μενού του οποίου οι ιδιότητες του παίρνονται από την ενέργεια a
JCheckBoxMenuItem(Icon icon)	Δημιουργεί στο μενού με μη επιλεγμένο με check box με εικονίδιο.
JCheckBoxMenuItem(String text)	Δημιουργεί στο μενού με μη επιλεγμένο με check box με κείμενο.
JCheckBoxMenuItem(String text, boolean b)	Δημιουργεί στο μενού ένα check box με κείμενο και επιλογή.
JCheckBoxMenuItem(String text, Icon icon)	Δημιουργεί στο μενού με μη επιλεγμένο με check box με κείμενο και εικονίδιο
JCheckBoxMenuItem(String text, Icon icon, boolean b)	It creates a check box menu item with the specified text, icon, and selection state.

ΠΑΡΑΔΕΙΓΜΑ

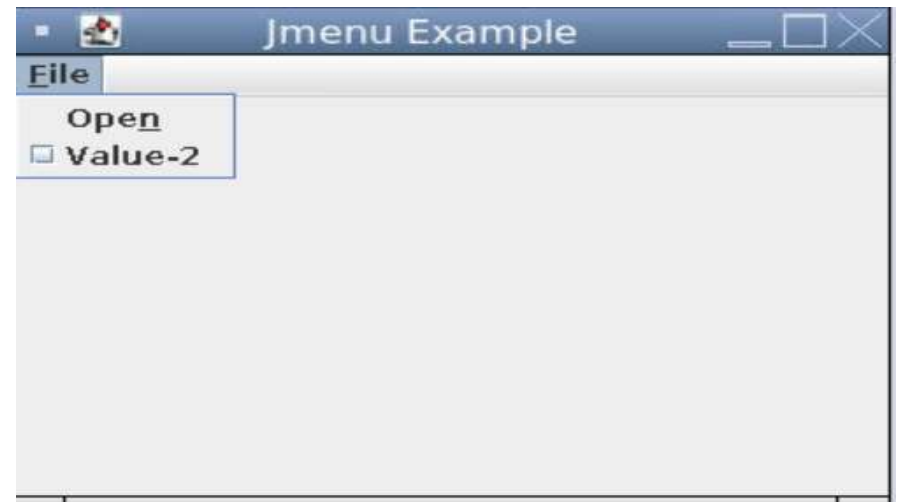
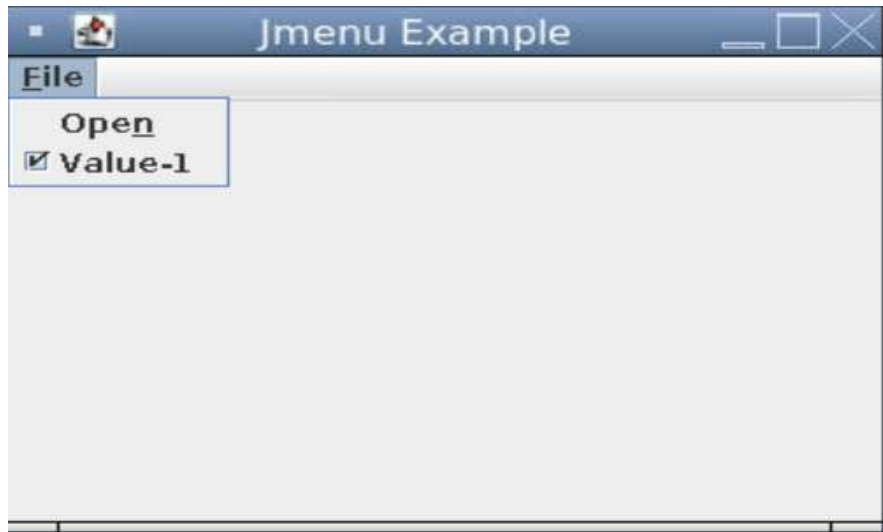
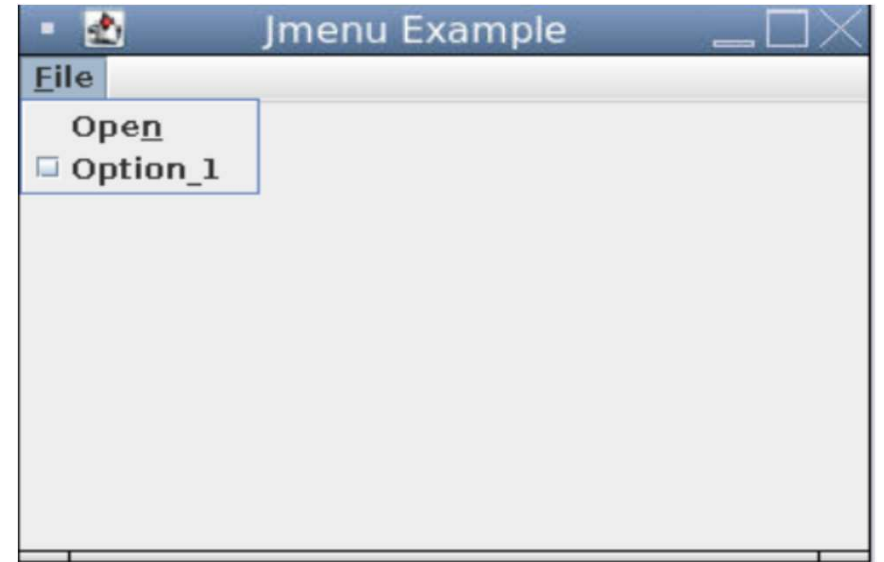
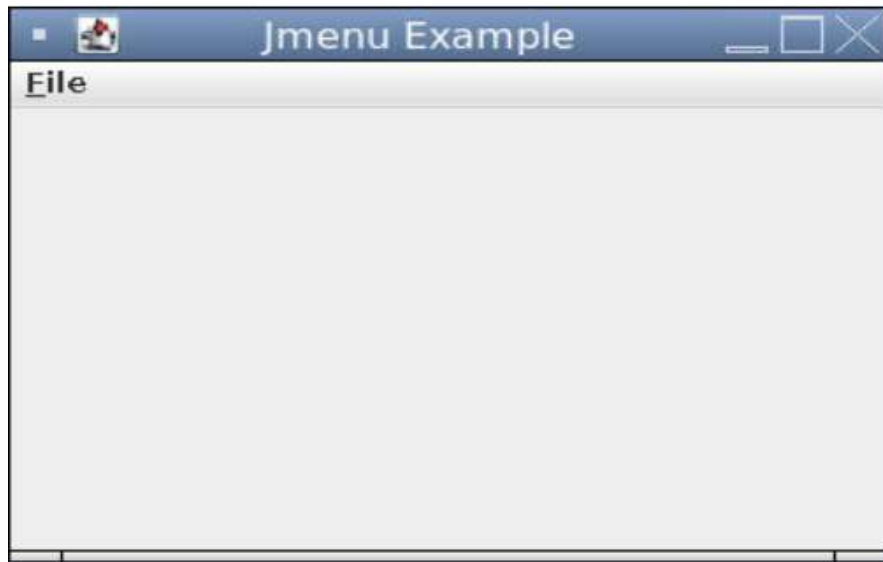
```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import javax.swing.AbstractButton;
import javax.swing.JCheckBoxMenuItem;
import javax.swing.JFrame;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;

public class Main {
    public static void main(final String args[]) {
        JFrame frame = new JFrame("Jmenu Example");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JMenuBar menuBar = new JMenuBar();
        // File Menu, F - Mnemonic
        JMenu fileMenu = new JMenu("File");
        fileMenu.setMnemonic(KeyEvent.VK_F);
        menuBar.add(fileMenu);
        // File->New, N - Mnemonic
        JMenuItem menuItem1 = new JMenuItem("Open", KeyEvent.VK_N);
        fileMenu.add(menuItem1);
        JCheckBoxMenuItem caseMenuItem = new
JCheckBoxMenuItem("Option_1");
        caseMenuItem.setMnemonic(KeyEvent.VK_C);
        fileMenu.add(caseMenuItem);
    }
}
```

```
ActionListener aListener = new ActionListener()
{
    public void actionPerformed(ActionEvent event)
    {
        AbstractButton aButton = (AbstractButton) event.getSource();
        boolean selected = aButton.getModel().isSelected();
        String newLabel;
        if (selected) {
            newLabel = "Value-1";
        } else {
            newLabel = "Value-2";
        }
        aButton.setText(newLabel);
    }
};

caseMenuItem.addActionListener(aListener);
frame.setJMenuBar(menuBar);
frame.setSize(350, 250);
frame.setVisible(true);
}
```

Η μέθοδος `setMnemonic` συσχετίζει μνημονικά στοιχεία με το μενού. Τα μνημονικά στοιχεία επιτρέπουν στο χρήστη να αλληλεπιδρά με τα στοιχεία του μενού χρησιμοποιώντας τα πλήκτρα στο πληκτρολόγιο



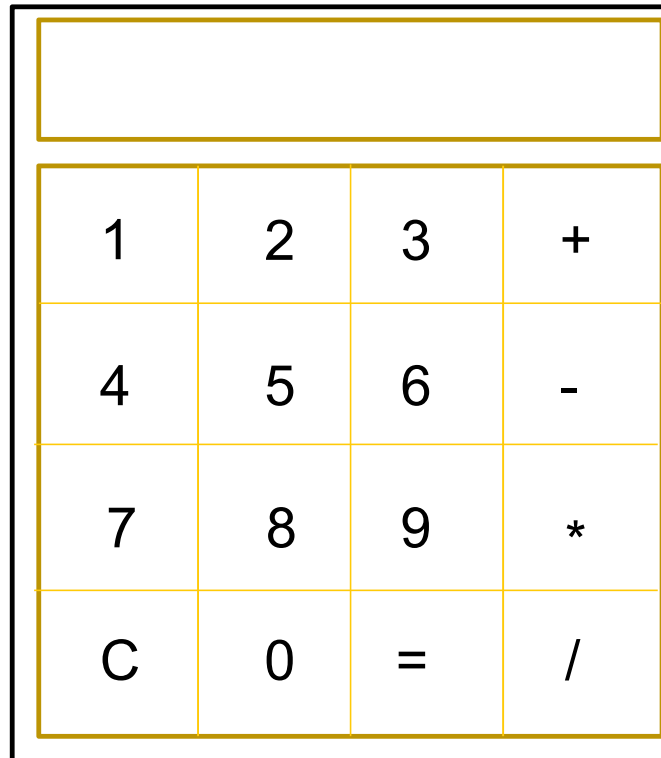
Παράδειγμα

Δημιουργήστε μια αριθμομηχανή.

Σχεδιασμός

Τα αποτελέσματα εμφανίζονται στην κορυφή τα πλήκτρα από κάτω

Χρειαζόμαστε ένα border layout για να βάλουμε το textbox στην κορυφή. Στο κέντρο θα βάλουμε τα κουμπιά. Βάζουμε ένα panel με grid layout



Textbox για να εκτυπώνει το αποτέλεσμα

Κουμπιά για καθένα από τα πλήκτρα

Υλοποίηση

```
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.border.Border;
import java.awt.BorderLayout;
import java.awt.GridLayout;
import java.awt.FlowLayout;
import java.awt.Color;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

public class Main extends JFrame implements ActionListener
{
    public static final int WIDTH = 300;
    public static final int HEIGHT = 200;
    private JPanel topPanel;
    private JPanel centralPanel;
    private JPanel gridPanel;

    public Main( )
    {
        super("Panel Demonstration");
        setSize(WIDTH, HEIGHT);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout( ));
```

```
        topPanel = new JPanel( );
        JTextField top=new JTextField("");
        topPanel.setLayout( new FlowLayout());
        topPanel.add(top);
        add(topPanel, BorderLayout.NORTH);
```

```
        centralPanel=new JPanel();
        centralPanel.setLayout(new GridLayout(4,4));
```

```
        JButton n1,n2,n3,n4,n5,n6,n7,n8,n9,n0,add,minus,mul,div,eq,c;
        n1= new JButton("1");
        n1.addActionListener(this);
        n2= new JButton("2");
        n2.addActionListener(this);
        n3= new JButton("3");
        n3.addActionListener(this);
        n4= new JButton("4");
        n4.addActionListener(this);
        n5= new JButton("5");
        n5.addActionListener(this);
```

ΣΥΝΕΧΕΙΑ

```
n6= new JButton("6");
n6.addActionListener(this);
n7= new JButton("7");
n7.addActionListener(this);
n8= new JButton("8");
n8.addActionListener(this);
n0= new JButton("0");
n8.addActionListener(this);
n9= new JButton("9");
n9.addActionListener(this);
add= new JButton("+");
add.addActionListener(this);
minus= new JButton("-");
minus.addActionListener(this);
mul= new JButton("*");
mul.addActionListener(this);
div= new JButton("/");
div.addActionListener(this);
c= new JButton("C");
c.addActionListener(this);
eq= new JButton("=");
eq.addActionListener(this);
}
```

```
centralPanel.add(n1);
    centralPanel.add(n2);
    centralPanel.add(n3);
    centralPanel.add(add);
    centralPanel.add(n4);
    centralPanel.add(n5);
    centralPanel.add(n6);
    centralPanel.add(minus);
    centralPanel.add(n7);
    centralPanel.add(n8);
    centralPanel.add(n9);
    centralPanel.add(mul);
    centralPanel.add(c);
    centralPanel.add(n0);
    centralPanel.add(eq);
    centralPanel.add(div);

    add(centralPanel, BorderLayout.CENTER);

    setVisible(true);
}

public static void main(String [] args)
{ Main m=new Main();}

public void actionPerformed(ActionEvent e) {
    System.exit(0); }
```

Υλοποίηση actionPerformed

```
public void actionPerformed(ActionEvent e) {
    String buttonString = e.getActionCommand( );
    if (buttonString.equals("1")|| buttonString.equals("2") ||
        buttonString.equals("3")|| buttonString.equals("4") ||
        buttonString.equals("5")|| buttonString.equals("6") ||
        buttonString.equals("7")|| buttonString.equals("8") ||
        buttonString.equals("9")|| buttonString.equals("0") ||
        buttonString.equals("+")||buttonString.equals("-")||
        buttonString.equals("*")||buttonString.equals("/"))
        {top.setText(top.getText()+buttonString);
        setVisible(true);}
    else if (buttonString.equals("C"))
        {top.setText("");
        setVisible(true);}
    else if (buttonString.equals("="))
        {top.setText("calculate the result");
        setVisible(true);}
    else
        System.exit(0); }
```

1	2	3	+
4	5	6	-
7	8	9	*
C	0	=	/

1588-69*3+47			
1	2	3	+
4	5	6	-
7	8	9	*
C	0	=	/

calculate the result			
1	2	3	+
4	5	6	-
7	8	9	*
C	0	=	/

1	2	3	+
4	5	6	-
7	8	9	*
C	0	=	/

JLIST

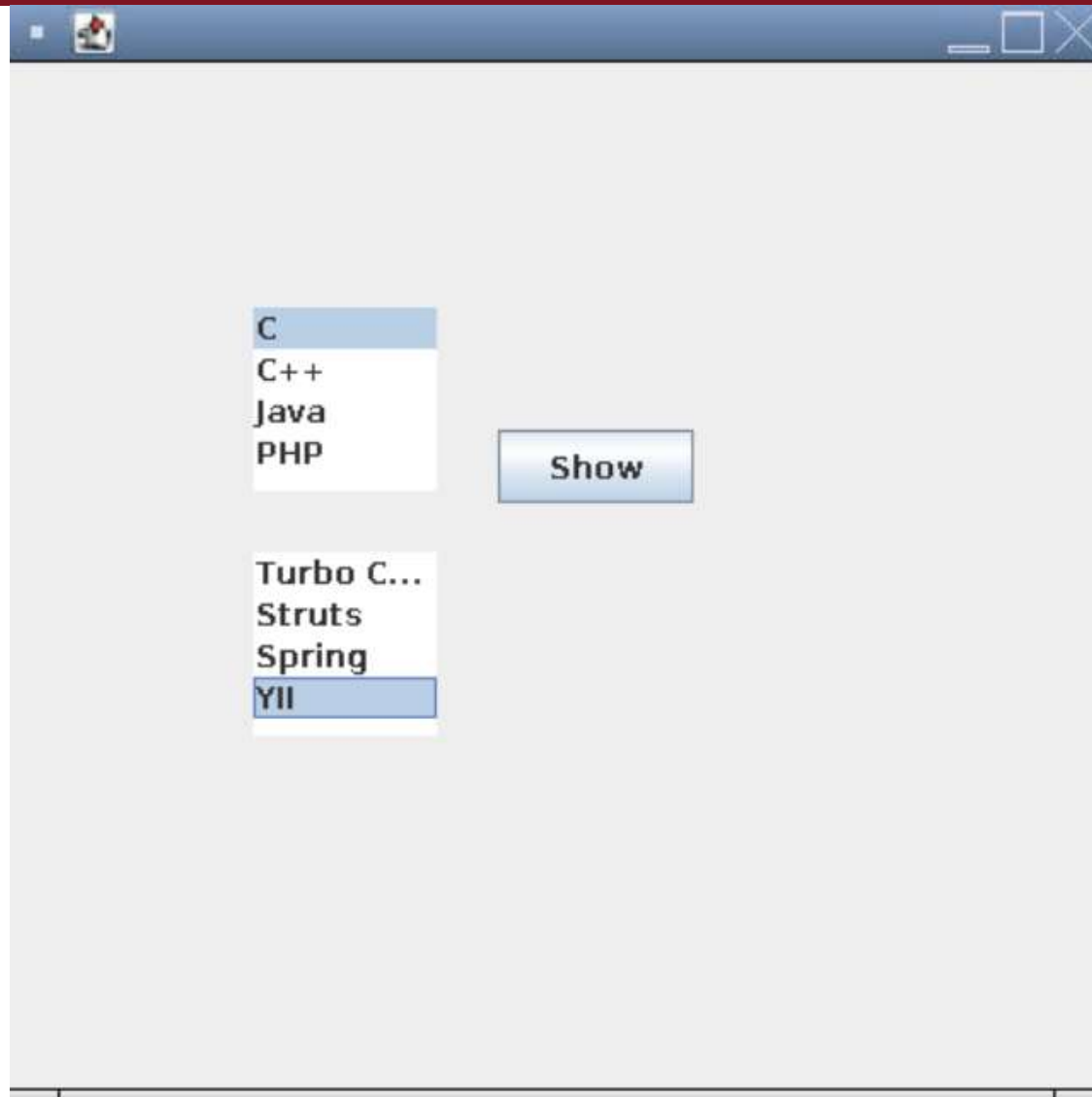
- Ένα στοιχείο που εμφανίζει μια λίστα αντικειμένων και επιτρέπει στο χρήστη να επιλέξει ένα ή περισσότερα αντικείμενα. Ένα ξεχωριστό μοντέλο, `ListModel`, διατηρεί τα περιεχόμενα της λίστας.
- Είναι εύκολο να εμφανίσετε έναν πίνακα ή ένα `Vector` αντικειμένων, χρησιμοποιώντας τον κατασκευαστή `JList` που δημιουργεί αυτόματα μία εμφάνιση `ListModel` μόνο για ανάγνωση για εσάς:

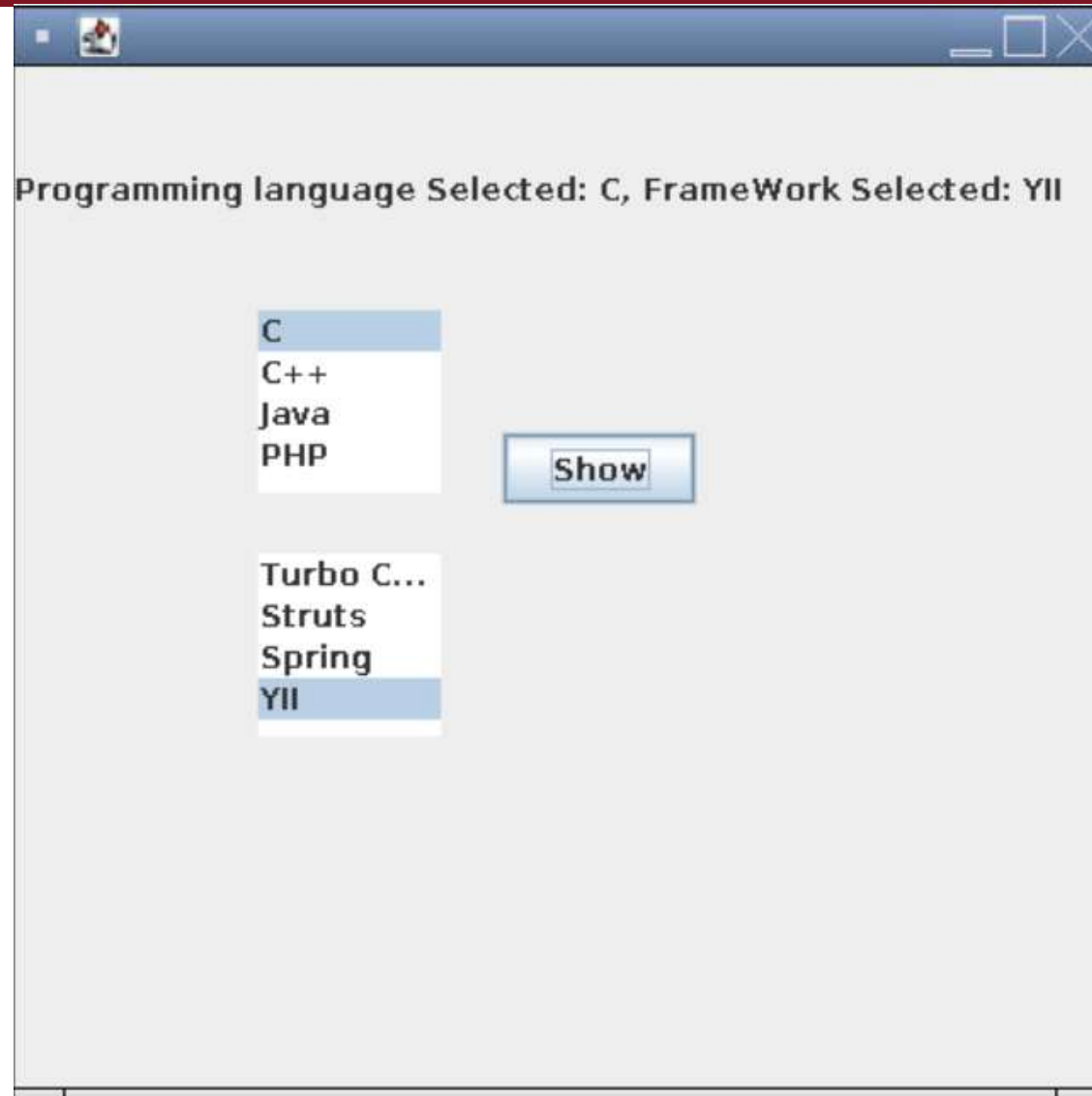
```
// Create a JList that displays strings from an array
String[] data = {"one", "two", "three", "four"};
JList<String> myList = new JList<String>(data);
```

ΠΑΡΑΔΕΙΓΜΑ JList

```
import javax.swing.*;
import java.awt.event.*;
public class ListExample
{
    ListExample(){
        JFrame f= new JFrame();
        final JLabel label = new JLabel();
        label.setSize(500,100);
        JButton b=new JButton("Show");
        b.setBounds(200,150,80,30);
        final DefaultListModel<String> l1 = new DefaultListModel<>();
        l1.addElement("C");
        l1.addElement("C++");
        l1.addElement("Java");
        l1.addElement("PHP");
        final JList<String> list1 = new JList<>(l1);
        list1.setBounds(100,100, 75,75);
        DefaultListModel<String> l2 = new DefaultListModel<>();
        l2.addElement("Turbo C++");
        l2.addElement("Struts");
        l2.addElement("Spring");
        l2.addElement("YII");
        final JList<String> list2 = new JList<>(l2);
        list2.setBounds(100,200, 75,75);
        f.add(list1); f.add(list2); f.add(b); f.add(label);
        f.setSize(450,450);
        f.setLayout(null);
        f.setVisible(true);
    }
}
```

```
        b.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                String data = "";
                if (list1.getSelectedIndex() != -1) {
                    data = "Programming language Selected: " +
list1.getSelectedValue();
                    label.setText(data);
                }
                if(list2.getSelectedIndex() != -1){
                    data += ", FrameWork Selected: ";
                    for(Object frame :list2.getSelectedValues()){
                        data += frame + " ";
                    }
                }
                label.setText(data);
            }
        });
    }
    public static void main(String args[])
    {
        new ListExample();
    }
}
```





Eclipse

- Η eclipse (αλλά και άλλα IDEs) μας δίνει πολλά έτοιμα εργαλεία για την δημιουργία GUIs
- Εγκαταστήσετε το plug-in *Windows Builder Pro*
- **Παράδειγμα:** Δημιουργήστε μια αριθμομηχανή.

Εισαγωγή μίας διαφάνειας στο Eclipse

- Το Eclipse οργανώνει τον κώδικα σε **projects**.
- Οι **κλάσεις** στη συνέχεια προστίθενται μέσα στο project.
- Πρέπει να έχετε εγκαταστήσει το plugin **Windows Builder Pro**.
- Για να φτιάξετε ένα GUI
 - Αρχικά πρέπει να φτιάξετε ένα Java Project
 - Συνέχεια προσθέτετε στο project. Επιλέξτε **Other>WindowsBuilder>SWING>Application Window**.
- Στη συνέχεια θα έχετε ένα **μενού** από τα διάφορα components τα οποία μπορείτε να προσθέτετε στο στην εφαρμογή σας.
 - Μπορείτε να δουλεύετε είτε με το **Design** είτε με τον **Source** κώδικα

Java - test/src/simple.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer

- DM-ex5
 - test
 - src
 - (default package)
 - simple.java

- JRE System Library [JavaSE-1.7]

IndexFiles.java simple.java

Structure

Components

- btnNewButton_2 - "-"
- btnNewButton_3 - "+"
- btnNewButton_1 - "4"
- button - "5"
- btnNewButton_5 - "6"
- button_1 - "-"
- button_2 - "7"
- btnC - "8"
- button_4 - "9"
- button_5 - "*"
- btnC_1 - "C"
- button_6 - "0"
- button_7 - "="

Properties

Variable	button_6
Constructor (Constructor properties)	
Class	javax.swing.JButton
background	240,240,240
enabled	<input checked="" type="checkbox"/> true
font	Tahoma 11
foreground	0,0,0
horizontalAlign...	CENTER
icon	
mnemonic(char)	
selectedIcon	
text	0
toolTipText	
verticalAlign	CENTER

Palette

- System
 - Selection
 - Marquee
 - Choose c...
 - Tab Order
- Containers
 - JPanel
 - JScrollPane
 - JSplitPane
 - JTabbedPane
 - JToolBar
 - JLayered...
 - JDesktop...
 - JInternal...
- Layouts
 - Absolute...
 - FlowLayout
 - BorderLa...
 - GridLayout
 - GridBagL...
 - CardLay...
 - BoxLayout
 - SpringLa...
 - FormLay...
 - MigLayout
 - GroupLa...
- Struts & Springs
- Components
 - JLabel
 - JTextField
 - JCombo...
 - JButt...
 - JCheckBox
 - JRadi...
 - JToggleB...
 - JText...
 - JFormat...
 - JPasswor...
 - JTextPane
 - JEditorPa...

JTextField

A lightweight component that allows the editing of a single line of text.

Problems @ Javadoc Declaration

java.awt.event.ActionListener

The listener interface for receiving action events. The class that is interested in processing an action event implements this interface, and the object created with that class is registered with the component's addActionListener method. When the action event occurs, that object's actionPerformed method is invoked.

Since: 1.1

Author: Carl Quinn

Writable Smart Insert 139 : 28

Δημιουργία κώδικα

- Τα IDEs μας επιτρέπουν να διαχωρίζουμε το **design** από τον **κώδικα**
 - Το πλεονέκτημα είναι ότι έχουμε ένα **WYSIWYG** interface με το οποίο μπορούμε να σχεδιάσουμε το GUI
 - Το μειονέκτημα είναι ότι δημιουργείται πολύς κώδικας **αυτόματα** ο οποίος δεν είναι πάντα όπως τον θέλουμε.
- Ο **διαχωρισμός** του σχεδιαστικού κομματιού από τις πράξεις που εκτελούν είναι γενικά μια καλή προγραμματιστική πρακτική.

Δημιουργία κώδικα

- Η δημιουργία ενός κουμπιού δημιουργεί αυτό τον κώδικα

```
 JButton button_6 = new JButton("0");  
 panel.add(button_6);
```

- Αν πατήσουμε πάνω στο κουμπί (double-click) δημιουργείται ο ακροατής του κουμπιού αυτόματα ως μια ανώνυμη κλάση

```
 JButton button_6 = new JButton("0");  
 button_6.addActionListener(new ActionListener() {  
     public void actionPerformed(ActionEvent e) {  
     }  
 });  
 panel.add(button_6);
```

Δημιουργία κώδικα

- Η δημιουργία ενός κουμπιού δημιουργεί αυτό τον κώδικα

```
 JButton button_6 = new JButton("0");  
 panel.add(button_6);
```

- Αν πατήσουμε πάνω στο κουμπί (double-click) δημιουργείται ο ακροατής του κουμπιού αυτόματα ως μια ανώνυμη κλάση
 - Εμείς συμπληρώνουμε τον κώδικα

```
 JButton button_6 = new JButton("0");  
 button_6.addActionListener(new ActionListener() {  
     public void actionPerformed(ActionEvent e) {  
         textField.setText(textField.getText()+"0");  
     }  
 });  
 panel.add(button_6);
```