

ΤΕΧΝΙΚΕΣ ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΟΥΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Graphical User Interfaces (GUI)
SWING

Swing

- Τα **GUIs** (**Graphical User Interfaces**) είναι τα συνηθισμένα interfaces που χρησιμοποιούν παράθυρα, κουμπιά, menus, κλπ
- Η **Swing** είναι η 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.

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** των ακροατών.



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)`

Παράθυρο με κουμπί

```
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);  
    }  
}
```

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.

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

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

```
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 μπούνε το ένα πάνω στο άλλο.
- Αυτό γίνεται με την εντολή **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 εντολή

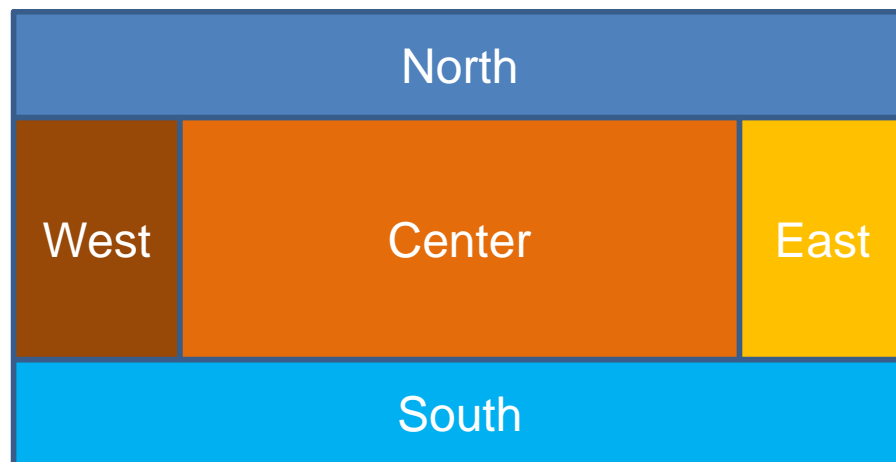
```
setLayout(new FlowLayout());
```

(Πρέπει να έχουμε κάνει `include 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 εντολή
`setLayout(new BorderLayout());`
(Πρέπει να έχουμε κάνει `include 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

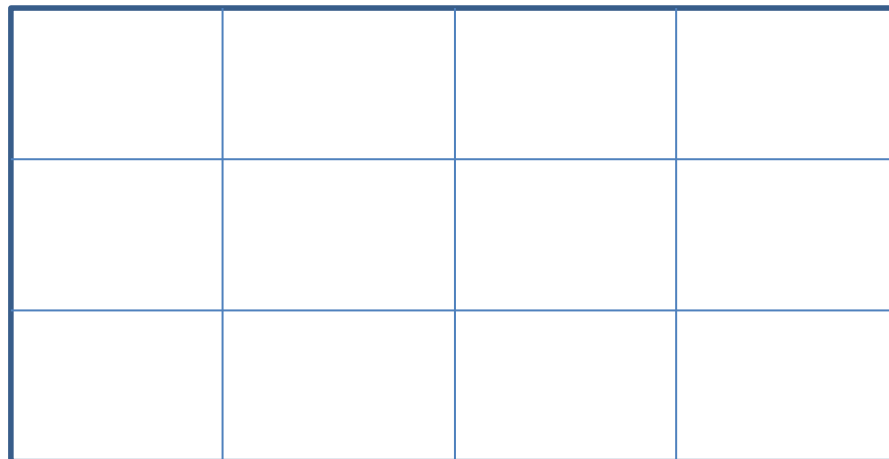
- Στην περίπτωση αυτή ορίζουμε ένα πλέγμα με n γραμμές και m στήλες και αυτό γεμίζει από τα αριστερά προς τα δεξιά και από πάνω προς τα κάτω
- Καλούμε την εντολή

```
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**

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

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);
```