Διάλεξη 18: Προγραμματισμός με Γραφικά (Graphics Programming)

Στην ενότητα αυτή θα μελετηθούν τα εξής επιμέρους θέματα:

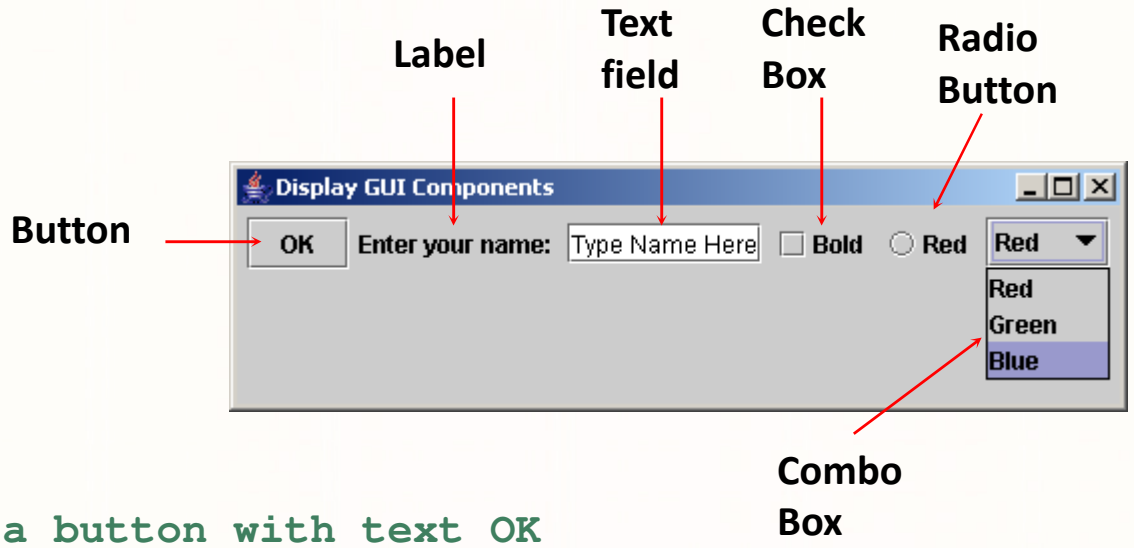
- Οι βιβλιοθήκες AWT και SWING, κύρια αντικείμενα
- Προγραμματισμός με συμβάντα (event-driven programming)

Διδάσκων: Παναγιώτης Ανδρέου

Προγραμματισμός με Γραφικά

- Εξαιρετικό παράδειγμα που συνδυάζει όλες τις έννοιες του αντικειμενοστρεφή προγραμματισμού
- Όλες οι γλώσσες προγραμματισμού (π.χ., JAVA:swing-awt, C#:System.Drawing, Visual Basic: System.Drawing, C++:OpenGL) περιέχουν βιβλιοθήκες για την δημιουργία γραφικών διαπροσωπειών
- Αυτές οι βιβλιοθήκες εξελίσσονται ραγδαία με κάθε ανανέωση του API!
- Υποστηρίζουν προγραμματισμό γραφικών για διαπροσωπείες υπολογιστή και διαδικτυακές εφαρμογές (π.χ., JAVA:applets, servlets, jsp)

Παράδειγμα Γραφικών στην JAVA

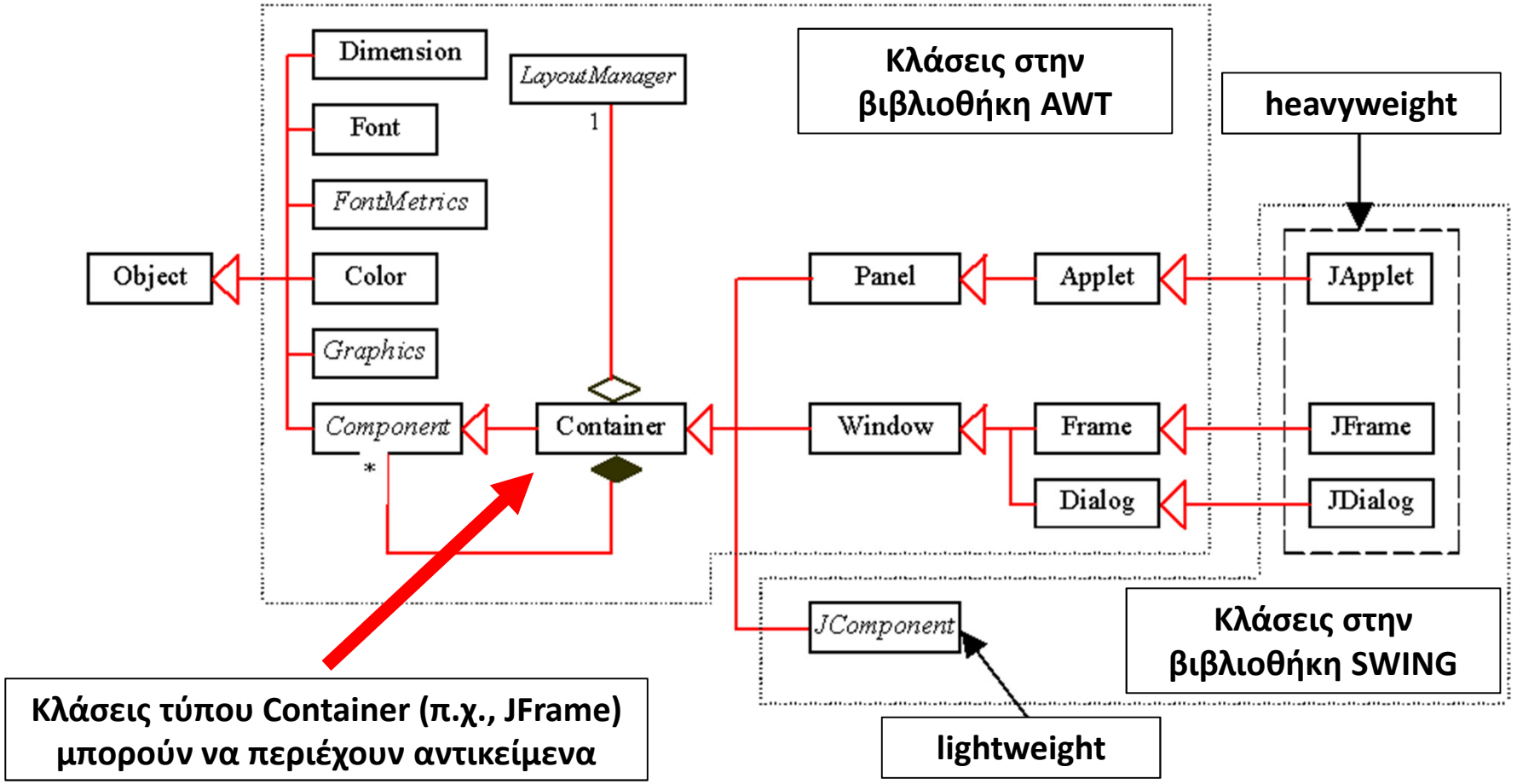



```
// Create a button with text OK
JButton jbtOK = new JButton("OK");
// Create a label with text "Enter your name: "
JLabel jlblName = new JLabel("Enter your name: ");
// Create a text field with text "Type Name Here"
JTextField jtfName = new JTextField("Type Name Here");
// Create a check box with text bold
JCheckBox jchkBold = new JCheckBox("Bold");
// Create a radio button with text red
JRadioButton jrbRed = new JRadioButton("Red");
// Create a combo box with choices red, green, and blue
JComboBox jcbColor = new JComboBox(new String[] { "Red","Green","Blue"});
```

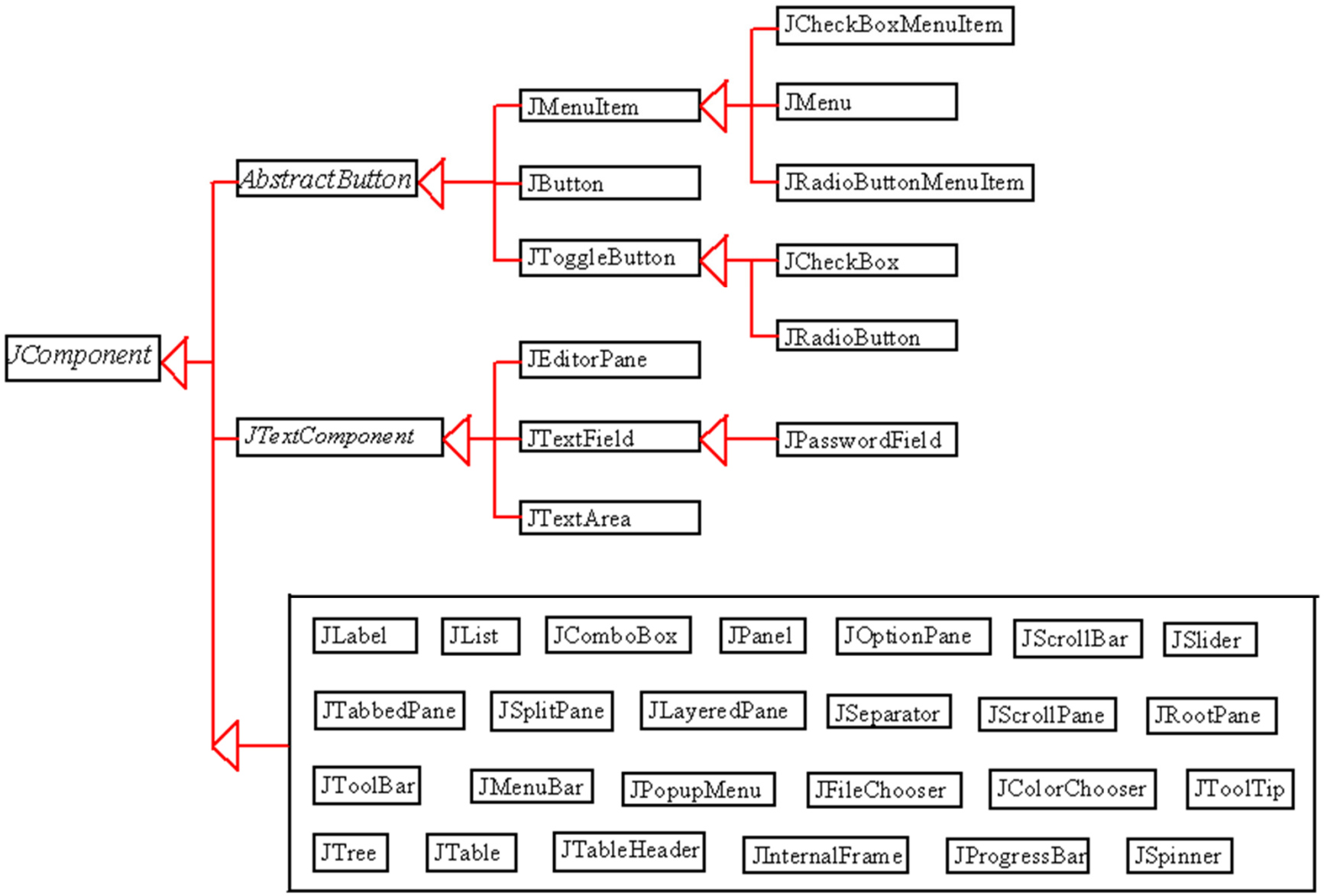
Βιβλιοθήκες Γραφικών (javax.swing vs. java.awt)

- Έχουμε χρησιμοποιήσει αντικείμενα με όνομα **J**<…>, π.χ., **J**button
- Αυτά τα αντικείμενα ανήκουν στην βιβλιοθήκη javax.swing
- Γιατί δεν ονομάστηκε απλά Button η κλάση;
- Επειδή ήδη υπάρχει η κλάση Button στην βιβλιοθήκη java.awt
- Η βιβλιοθήκη java.awt (Abstract Windows Toolkit) ήταν η πρώτη βιβλιοθήκη που δημιουργήθηκε για γραφικά στην JAVA
- Τα αντικείμενα της AWT συνδέονται αυτόματα με συστατικά της πλατφόρμας με την οποία τρέχει:
  - **Πρόβλημα: Προβλήματα της πλατφόρμας ανάγονται στο πρόγραμμα**
- Με την έκδοση της JAVA 2 ενσωματώθηκε η βιβλιοθήκη Swing η οποία επιτρέπει τον σχεδιασμό αντικειμένων απευθείας με java κώδικα (εκτός από java.awt.Window or java.awt.Panel) που πρέπει να σχεδιαστούν από την πλατφόρμα/λειτουργικό.
  - Πλεονέκτημα 1: μεγάλη ανεξαρτησία από την πλατφόρμα
  - Πλεονέκτημα 2: χρήση πιο λίγων πόρων από την πλατφόρμα
- Τα αντικείμενα της Swing αναφέρονται σαν lightweight components ενώ της AWT σαν heavyweight

Ιεραρχία Κλάσεων swing

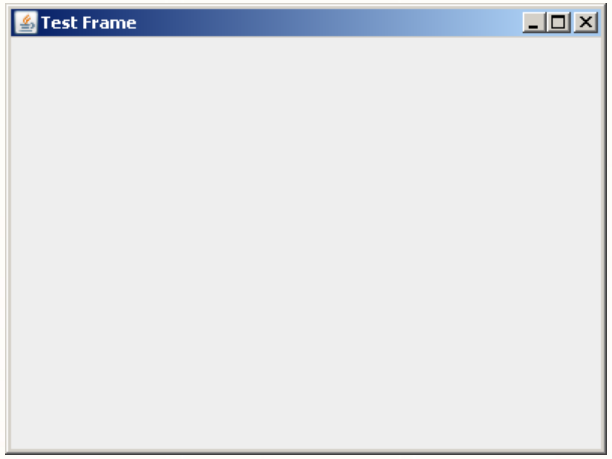


Συστατικά GUI (swing)



Frames

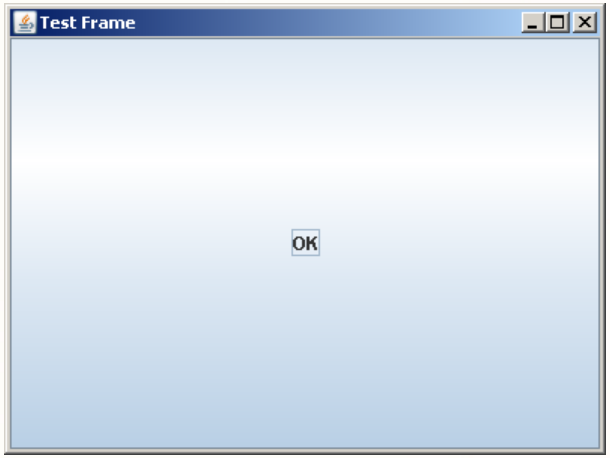
- Το frame είναι ένα είδος παράθυρου το οποίο δεν περιέχεται σε άλλο παράθυρο
- **Είναι container**: μπορεί να περιέχει άλλα αντικείμενα




```
import javax.swing.JFrame;

public class MyFrame {

  public static void main(String[] args) {
    JFrame frame = new JFrame("Test Frame");
    frame.setSize(400, 300);
    frame.setVisible(true);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  }

}
```

Πρόσθεση αντικειμένων σε Frames

- Η πρόσθεση αντικειμένων γίνεται με την συνάρτηση frame.add() (frame.getContentPane().add()):
  1. με πρώτα την δημιουργία του αντικειμένου και μετά πρόσθεση του
  2. Με ταυτόχρονη δημιουργία του αντικειμένου και πρόσθεση του




```
import javax.swing.JFrame;
public class MyFrame {
  public static void main(String[] args) {
    JFrame frame = new JFrame("Test Frame");
    frame.set …

    JButton jbtOK = new JButton("OK");
    frame.add(jbtOK);

    frame.add(new JButton("OK"));
  }
}
```

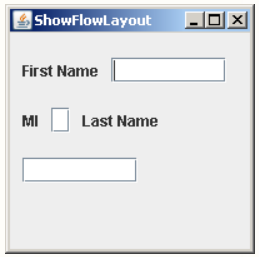
Διαχειριστές Σχεδίου (Layout Managers)

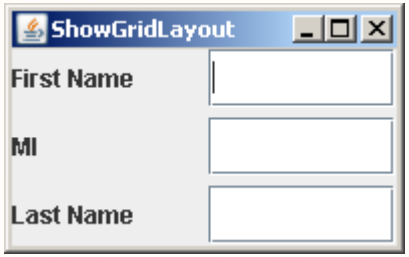
- Οι layout managers καθορίζουν πως θα εμφανίζεται η διαπροωπεία σε οποιαδήποτε πλατφόρμα που υποστηρίζει παράθυρα (windows)
- Κάθε container έχει ένα layout manager που διαρρυθμίζει τα γραφικά αντικείμενα μέσα στο container.

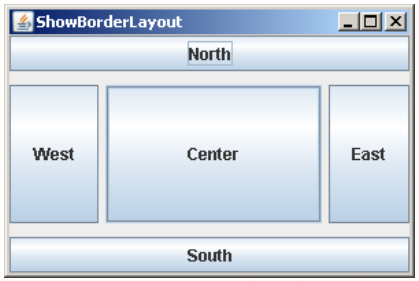
- Για να καθορίσουμε τον διαχειριστή ενός container χρησιμοποιούμε την μέθοδο setLayout(LayoutManager).

- Παραδείγματα LayoutManager:
  - FlowLayout: το ένα αντικείμενο κάτω από το άλλο σειριακά
  - GridLayout: διαρρύθμιση μέσα σε πίνακα
  - BorderLayout: διαρρύθμιση σε East, South, West, North, Center.

Παραδείγματα Layout Managers








```
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.JFrame;
import java.awt.FlowLayout;

public class ShowFlowLayout extends JFrame {
  public ShowFlowLayout() {
    // Set FlowLayout, aligned left with
horizontal gap 10
    // and vertical gap 20 between components
    setLayout(new FlowLayout(FlowLayout.LEFT,
10, 20));

    add(new JLabel("First Name"));
    add(new JTextField(8));
    add(new JLabel("MI"));
    add(new JTextField(1));
    add(new JLabel("Last Name"));
    add(new JTextField(8));
  }

  public static void main(String[] args) {
    ShowFlowLayout frame = new
ShowFlowLayout();
    frame.setTitle("ShowFlowLayout");
    frame.setSize(200, 200);
    frame.setLocationRelativeTo(null); //
Center the frame

frame.setDefaultCloseOperation(JFrame.EXIT_ON_
CLOSE);
    frame.setVisible(true);
  }
}
```

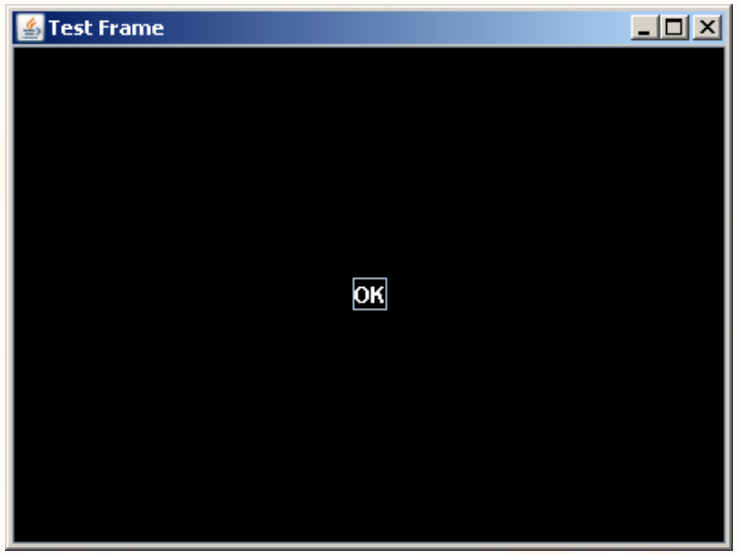
```
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.JFrame;
import java.awt.GridLayout;

public class ShowGridLayout extends JFrame {
  public ShowGridLayout() {
    // Set GridLayout, 3 rows, 2 columns, and
gaps 5 between components horizontally and
vertically
    setLayout(new GridLayout(3, 2, 5, 5));

    // Add labels and text fields to the frame
    add(new JLabel("First Name"));
    add(new JTextField(8));
    add(new JLabel("MI"));
    add(new JTextField(1));
    add(new JLabel("Last Name"));
    add(new JTextField(8));
  }

  /** Main method */
  public static void main(String[] args) {
    ShowGridLayout frame = new
ShowGridLayout();
    frame.setTitle("ShowGridLayout");
    frame.setSize(200, 125);
    frame.setLocationRelativeTo(null); //
Center the frame

frame.setDefaultCloseOperation(JFrame.EXIT_ON_
CLOSE);
    frame.setVisible(true);
  }
}
```

```
import javax.swing.JButton;
import javax.swing.JFrame;
import java.awt.BorderLayout;

public class ShowBorderLayout extends JFrame {
  public ShowBorderLayout() {
    // Set BorderLayout with horizontal gap 5
and vertical gap 10
    setLayout(new BorderLayout(5, 10));

    // Add buttons to the frame
    add(new JButton("East"),
BorderLayout.EAST);
    add(new JButton("South"),
BorderLayout.SOUTH);
    add(new JButton("West"),
BorderLayout.WEST);
    add(new JButton("North"),
BorderLayout.NORTH);
    add(new JButton("Center"),
BorderLayout.CENTER);
  }

  /** Main method */
  public static void main(String[] args) {
    ShowBorderLayout frame = new
ShowBorderLayout();
    frame.setTitle("ShowBorderLayout");
    frame.setSize(300, 200);
    frame.setLocationRelativeTo(null); //
Center the frame

frame.setDefaultCloseOperation(JFrame.EXIT_ON_
CLOSE);
    frame.setVisible(true);
  }
}
```

Διαχείριση Χρωμάτων

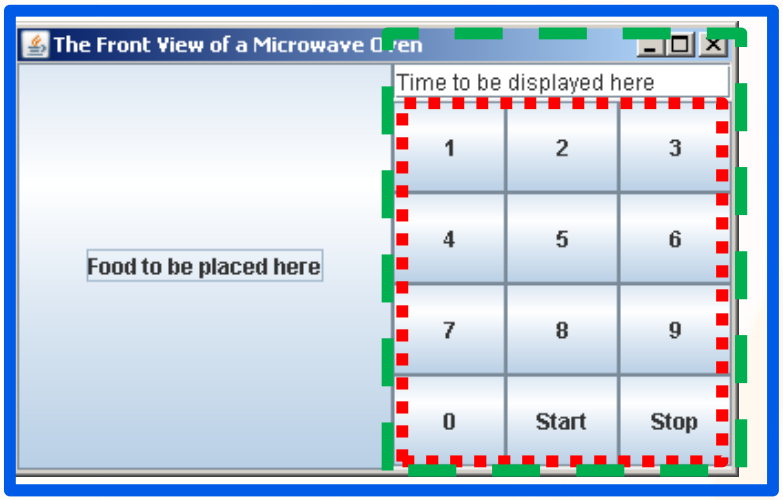
- Ο καθορισμός χρωμάτων γίνεται μέσω της java.awt.Color βάση του πρότυπου red-green-blue (RGB)
- Δημιουργία Χρωμάτων: `Color c = new Color(r, g, b);`
- Παράδειγμα γραφικών μεθόδων που χρησιμοποιούν χρώματα:
  - Για το background: setBackground(Color c)
  - Για το foreground: setForeground(Color c)

```
import javax.swing.JFrame;
public class MyFrame {
  public static void main(String[] args) {
    JFrame frame = new JFrame("Test Frame");
    frame.set …

    JButton jbtOK = new JButton("OK");

     jbtOK.setForeground(
         new Color(255,255,255));
     jbtOK.setBackground(
         new Color(0,0,0));

    frame.add(jbtOK);
  }
}
```



Panels

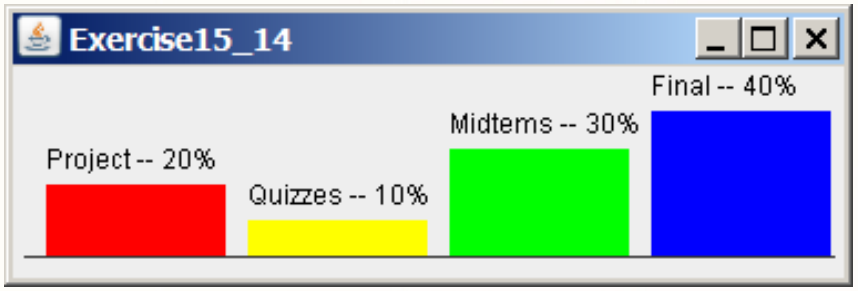
- Τα Panels χρησιμοποιούνται σαν sub-containers για οργάνωση
- Συνήθως, τοποθετούμε τα components μέσα σε panels και μετά τα panels στο frame
- Παράδειγμα:
  - JPanel p = new JPanel();
  - p.add(new JButton("OK"));

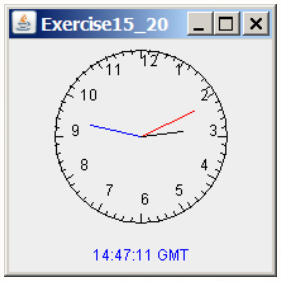



```
// In class code….
   JPanel p1 = new JPanel();
   p1.setLayout(new GridLayout(4, 3));
   // Add buttons to the panel
   for (int i = 1; i <= 9; i++) {
     p1.add(new JButton("" + i));
   }
   p1.add(new JButton("" + 0));
   p1.add(new JButton("Start"));
   p1.add(new JButton("Stop"));

   JPanel p2 = new JPanel(
                       new BorderLayout());
   p2.add(new JTextField(
             "Time to be displayed here"),
             BorderLayout.NORTH);
   p2.add(p1, BorderLayout.CENTER);

   // add contents into the frame
   add(p2, BorderLayout.EAST);
   add(new JButton(
         "Food to be placed here"),
         BorderLayout.CENTER);
```

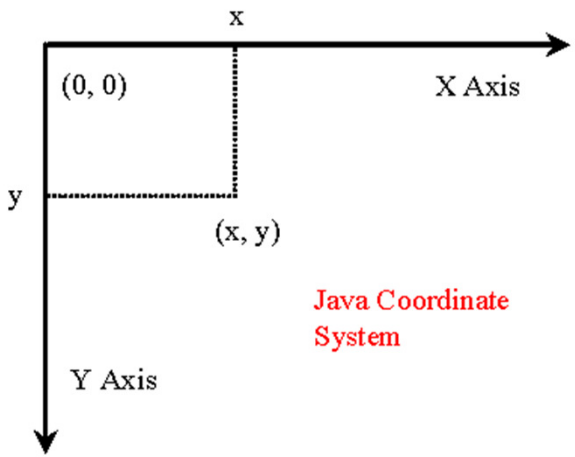
Σχεδίαση Αυθαίρετων Γραφικών (Custom Graphics)

- Πως σχεδιάζουμε αυθαίρετα (custom) γραφικά;

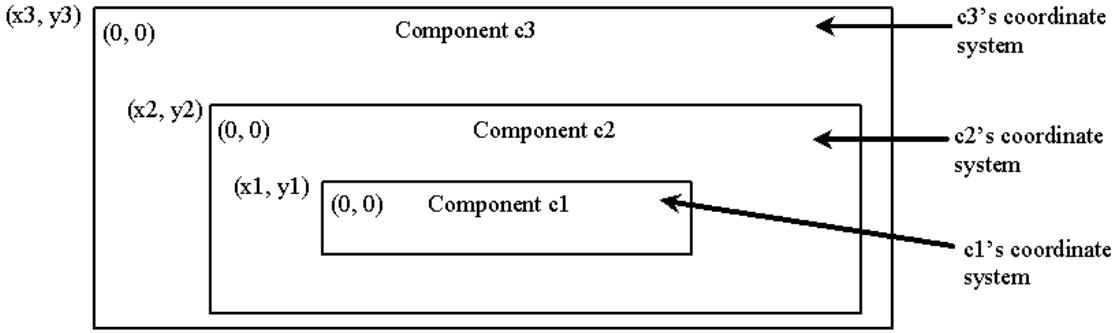




- Με την χρήση της κλάσης java.awt.Graphics
- Παρέχει μεθόδους για σχεδίαση γραμμών, ορθογώνιων, κτλ.
- Σύστημα Συντεταγμένων JAVA



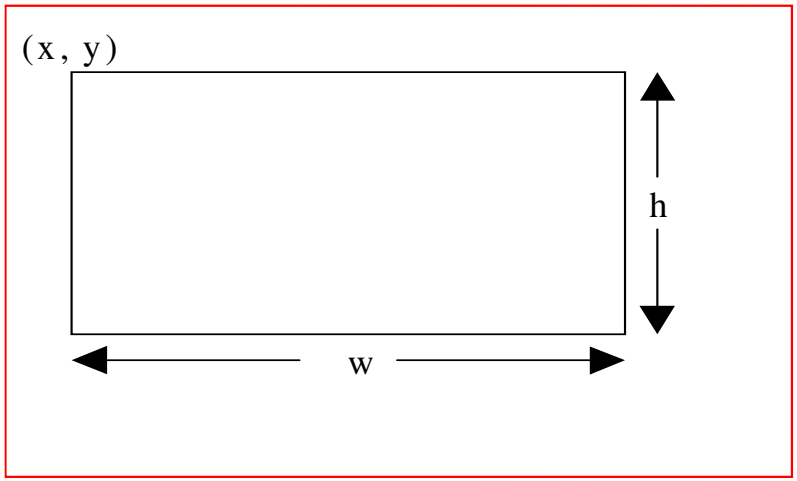
Σύστημα Συντεταγμένων σε αντικείμενα

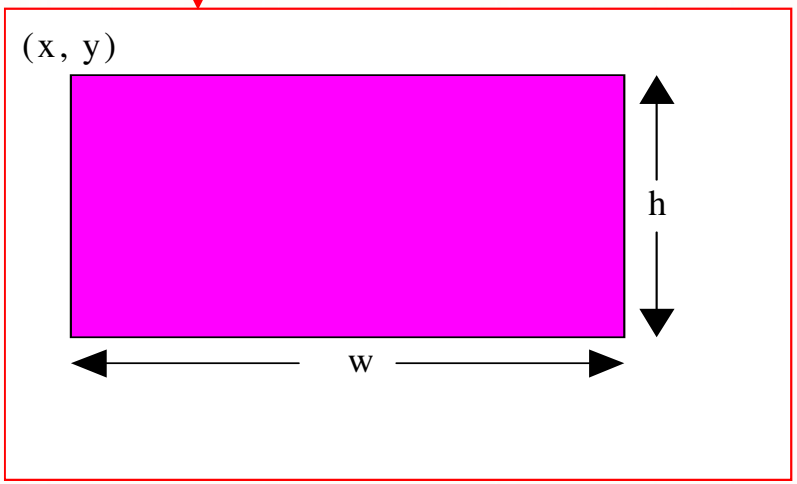


Παράδειγμα Σχεδίασης Αυθαίρετων Γραφικών

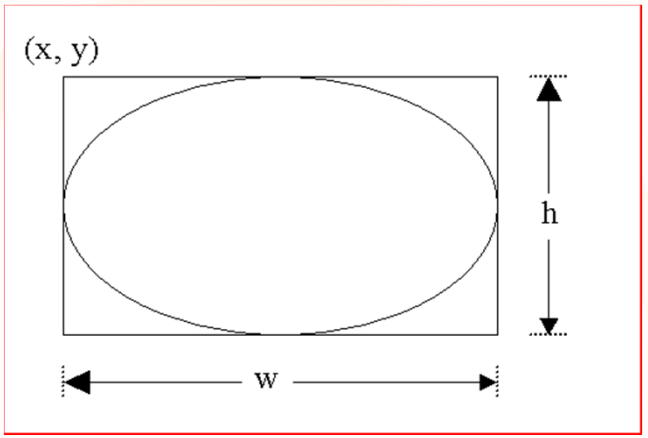
```
drawRect(int x, int y, int w, int h);
```

```
fillRect(int x, int y, int w, int h);
```






```
drawOval(int x, int y,
         int w, int h);

fillOval(int x, int y,
         int w, int h);
```



Διαχείριση Συμβάντων (Event handling)

- **Ερώτηση:** Έστω ότι θέλουμε να διαχειριστούμε το τι συμβαίνει όταν ο χρήστης πατήσει ένα κουμπί. Πως γίνεται αυτό;
  Παράδειγμα: όταν πατήσω το κουμπί "OK" να εκτελείται
  `System.out.println("OK clicked");`

- **Απάντηση:** Χρησιμοποιώντας προγραμματισμός με συμβάντα (event-driven programming)

- **Βασική διαφορά Procedural vs. Event-driven Programming**
  - **Procedural Programming:** η εκτέλεση είναι σειριακή βάση μίας ροής
  - **Event-driven Programming:** η εκτέλεση γίνεται όταν συμβεί κάποιο συμβάν (event) π.χ., πάτημα ενός κουμπιού

Παράδειγμα Event Handling με Listener

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

public class HandleEvent extends JFrame {
  public HandleEvent() {

    // Create and add button ok
    JButton jbtOK = new JButton("OK");
    add(jbtOK);

    // Register listeners
    OKListenerClass listener1 = new OKListenerClass();
    jbtOK.addActionListener(listener1);
  }

  public static void main(String[] args) {
    JFrame frame = new HandleEvent();
    frame.setTitle("Handle Event");
    frame.setSize(200, 150);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setVisible(true);
  }
}

class OKListenerClass implements ActionListener {
  public void actionPerformed(ActionEvent e) {
    System.out.println("OK button clicked");
  }
}
```

Η HandleEvent «είναι» ένα JFrame

Δημιουργία ενός listener.
Ο listener «ακούει» για διάφορα συμβάντα.

Που είναι ορισμένος;

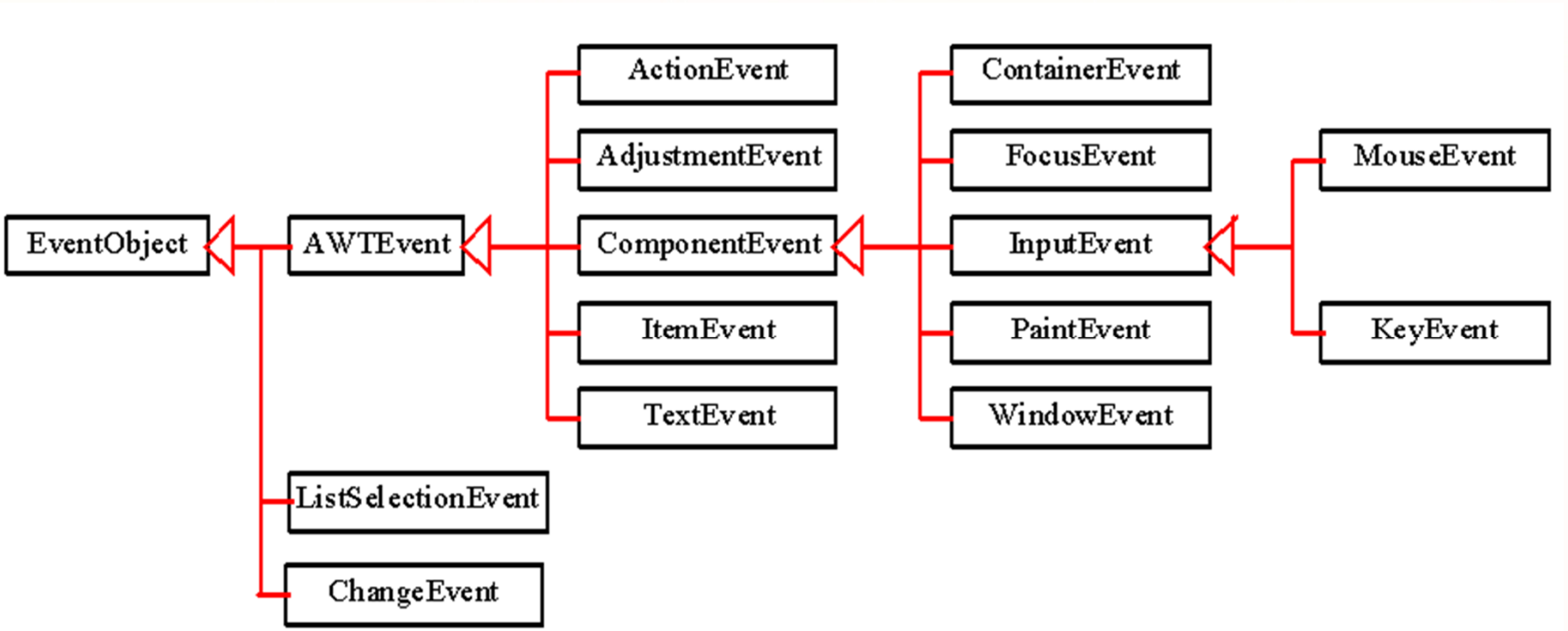
Διαχείριση Συμβάντων (Event handling) (συν.)

- Η συνάρτηση
  void addActionListener(ActionListener l)
  προσθέτει ένα Listener (ActionListener) στο κουμπί jbtOK.
- java.awt.event.ActionListener: είναι μία διαπροσωπεία (interface) που πρέπει να υλοποιεί κάθε Listener κλάση
- Η κλάση Listener που δημιουργούμε, παρέχει δηλώσεις που ορίζουν τι θα γίνει όταν συμβεί ένα event.

```
class OKListenerClass implements ActionListener {
  public void actionPerformed(ActionEvent e) {
    System.out.println("OK button clicked");
  }
}
```

- Ένα event μπορεί να θεωρηθεί ένα σήμα (signal) από το πρόγραμμα που δηλώνει ότι κάτι έχει συμβεί
- **Έχουμε ορίσει ακόμα μία top-level κλάση στο ίδιο αρχείο!**

Κλάσεις Συμβάντων (Event Classes)



Συσχέτιση Ενεργειών Χρήστη με Events

User Action	Source Object	Event Type generated
Click a button	JButton	ActionEvent
Click a check box	JCheckBox	ItemEvent, ActionEvent
Click a radio button	JRadioButton	ItemEvent, ActionEvent
Press return on a text field	JTextField	ActionEvent
Select a new item	JComboBox	ItemEvent, ActionEvent
Window opened, closed, etc.	Window	WindowEvent
Mouse pressed, released, etc.	Component	MouseEvent
Key released, pressed, etc.	Component	KeyEvent

Συσχέτιση Events με Event Handlers

Event Class	Listener Interface	Listener Methods (Handlers)
ActionEvent	ActionListener	actionPerformed(ActionEvent)
ItemEvent	ItemListener	itemStateChanged(ItemEvent)
WindowEvent	WindowListener	windowClosing(WindowEvent)
		windowOpened(WindowEvent)
		windowIconified(WindowEvent)
		windowDeiconified(WindowEvent)
		windowClosed(WindowEvent)
		windowActivated(WindowEvent)
		windowDeactivated(WindowEvent)
ContainerEvent	ContainerListener	componentAdded(ContainerEvent)
		componentRemoved(ContainerEvent)
MouseEvent	MouseListener	mousePressed(MouseEvent)
		mouseReleased(MouseEvent)
		mouseClicked(MouseEvent)
		mouseExited(MouseEvent)
		mouseEntered(MouseEvent)
KeyEvent	KeyListener	keyPressed(KeyEvent)
		keyReleased(KeyEvent)
		keyTypeed(KeyEvent)

Παράδειγμα Event Handling με ΔΥΟ Listeners

```
    JButton jbtOK = new JButton("OK");
    add(jbtOK);
    // Register listeners
    OKListenerClass listener1 = new OKActionListener();
    jbtOK.addActionListener(listener1);

    OKListenerClass listener2 = new OKMouseListener();
    jbtOK.addMouseListener(listener2);
  }
  public static void main(String[] args) {
    JFrame frame = new HandleEvent();
    …
  }
}
class OKActionListener implements ActionListener {
  public void actionPerformed(ActionEvent e) {
    System.out.println("OK button clicked");
  }
}
class OKMouseListener implements MouseListener {
  public void mouseClicked(MouseEvent arg0) {
    System.out.println("mouseClicked");
  }
  public void mouseEntered(MouseEvent arg0) {
    System.out.println("mouseEntered");
  }
}
```

Listener1
ActionListener

Listener2
MouseListener

Προβλήματα Κλάσεων Listeners

- Στο προηγούμενο παράδειγμα έχουμε δηλώσει 2 listeners σαν δύο ξεχωριστές κλάσεις
- Οι κλάσεις αυτές θα χρησιμοποιηθούν μόνο μέσα στο συγκεκριμένο πρόγραμμα/κλάση που έχουμε αναπτύξει.
- **Ερώτηση:** Μήπως έχουν και άλλες κλάσεις πρόσβαση σε αυτές τις κλάσεις;
- **Απάντηση:** Δυστυχώς ΝΑΙ! (τουλάχιστον στο ίδιο πακέτο)

- **Ερώτηση:** Μπορώ να δηλώσω μία κλάση listener (ActionListener) που να την χρησιμοποιήσω με όλα τα αντικείμενα που έχω;
- **Απάντηση:** Δυστυχώς ΟΧΙ!
  Μπορώ όμως να υλοποιήσω την διαπροσωπεία ActionListener στο top-level επίπεδο και να διαχειριστώ όλα τα ActionEvents από όλα τα αντικείμενα

Διαχείριση Events από πολλά αντικ. εντός κλάσης

```
import …
public class HandleMultipleComponents
        extends JFrame implements ActionListener{
  // Create two buttons
  JButton jbtOK = new JButton("OK");
  JButton jbtCancel = new JButton("Cancel");

  public HandleMultipleComponents() {
    setLayout(new FlowLayout(FlowLayout.LEFT, 10, 20));
    add(jbtOK); add(jbtCancel);
    // Register class listener
    jbtOK.addActionListener(this);
    jbtCancel.addActionListener(this);
  }
  //Implement actionPerformed
  public void actionPerformed(ActionEvent e) {
    if (e.getSource() == jbtOK)
      System.out.println("OK");
    else if (e.getSource() == jbtCancel)
      System.out.println("Cancel");

  public static void main(String[] args) {
    JFrame frame = new HandleMultipleComponents();
    …
  }
 }
}
```

Υλοποίηση Διαπροσωπείας ActionListener Πρέπει να υλοποιηθεί η μέθοδος actionPerformed()

Πέρασμα του αντικειμένου της κλάσης (this) σαν παραμέτρο

Παρατηρήσεις

- Αν και η λύση της προηγούμενης διαφάνειας είναι ελκυστική ας αναλύσουμε τι συμβαίνει στις εξής περιπτώσεις:

1. Η διαπροσωπεία μας περιέχει ***n*** components
   Χρειαζόμαστε ***n*** if statements
2. Η διαπροσωπεία μας διαχειρίζεται ***m*** events
   Χρειαζόμαστε ***m*** μεθόδους διαχείρισης (actionPerformed)
3. Η διαπροσωπεία μας περιέχει *n* components και το καθένα εγείρει *m* events
   Χρειαζόμαστε ***m*** μεθόδους διαχείρισης (actionPerformed) και ο καθένας θα διαχειρίζεται ***n*** if statements (***n*** x ***m***)

- … και αυτό χωρίς τον κώδικα διαχείρισης (δηλ. τι θα γράψουμε μέσα στη μέθοδο)

- **Υπάρχει καλύτερη λύση; ΝΑΙ, τα nested (inner) classes.**