# ΧΡΗΣΗ ΤΗΣ ΒΙΒΛΙΟΘΗΚΗΣ SWING

## Το πρώτο παράδειγμα

Στο πρώτο παράδειγμα δείχνουμε ένα παράθυρο στην οθόνη.

```
import java.awt.EventQueue;
import javax.swing.JFrame;

public class SimpleEx extends JFrame {

    public SimpleEx() {

        initUI();
    }

    private void initUI() {

        setTitle("Simple example");
        setSize(300, 200);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(EXIT ON CLOSE);
    }

    public static void main(String[] args) {

        EventQueue.invokeLater(new Runnable() {

            @Override
            public void run() {
                SimpleEx ex = new SimpleEx();
                ex.setVisible(true);
            }
        });
    }
}
```

Αν και αυτός ο κώδικας είναι πολύ μικρός, το παράθυρο της εφαρμογής μπορεί να κάνει αρκετά. Μπορούμε να αλλάξουμε το μέγεθός του, να μεγιστοποιηθεί ή να ελαχιστοποιηθεί. Όλη η πολυπλοκότητα έχει κρυφτεί από τον προγραμματιστή της εφαρμογής. Εδώ κάνουμε import τις διάφορες Swing classes που χρησιμοποιούνται στο παράδειγμα.

```
import java.awt.EventQueue;
import javax.swing.JFrame;
```

```
public class SimpleEx extends JFrame {
```

Παρατηρείστε ότι η SimpleEx class κληρονομεί το JFrame component που είναι το top-level container. Η βασική λειτουργία των containers είναι να «κρατούν» τα components της εφαρμογής.

```
public SimpleEx() {

    initUI();
}
```

Είναι μια καλή πρακτική προγραμματισμού να μην βάζουμε τον κώδικα εφαρμογής σε κατασκευαστές, αλλά να αναθέτουμε αυτή την εργασία σε μια συγκεκριμένη μέθοδο.

```
setTitle("Simple example");
```

Εδώ βάζουμε τον τίτλο του JFrame με την `setTitle()` μέθοδο.

```
setSize(300, 200);
```

Κάνουμε resize το παράθυρο να είναι 300 px wide και 200 px tall.

```
setLocationRelativeTo(null);
```

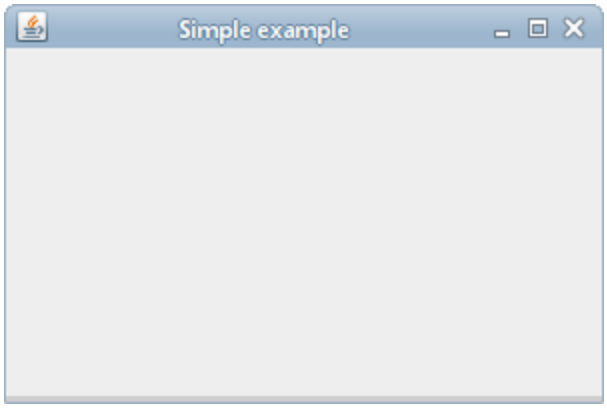
Αυτή η γραμμή κώδικα κεντράρει το παράθυρο στην οθόνη

```
setDefaultCloseOperation(EXIT_ON_CLOSE);
```

Αυτή η μέθοδος θα κλείσει το παράθυρο, αν κάνουμε κλικ στο κουμπί Κλείσιμο της γραμμής τίτλου. Από μόνο του δεν συμβαίνει τίποτα.

```
EventQueue.invokeLater(new Runnable() {
    @Override
    public void run() {
        SimpleExample ex = new SimpleExample();
        ex.setVisible(true);
    }
});
```

Δημιουργούμε ένα στιγμιότυπο του κώδικά μας και το κάνουμε ορατό στην οθόνη . Η μέθοδος `invokeLater()`τοποθετεί την εφαρμογή στο Swing Event Queue. Χρησιμοποιείται για να σιγουρέψουμε το γεγονός ότι όλα τα UI updates είναι concurrency-safe.



Διάγραμμα 1: Απλό παράδειγμα

## To Quit button

Στο παρακάτω παράδειγμα θα βάλουμε και ένα κουμπί το οποίο όταν τα το κάνουμε κλικ τότε η εφαρμογή θα τερματίζεται και το παράθυρο θα κλείνει

```
import java.awt.Container;
import java.awt.EventQueue;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.GroupLayout;
import javax.swing.JButton;
import javax.swing.JComponent;
import javax.swing.JFrame;

public class QuitButtonEx extends JFrame {

    public QuitButtonEx() {

        initUI();
    }

    private void initUI() {

        JButton quitButton = new JButton("Quit");

        quitButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent event) {
                System.exit(0);
            }
        });

        createLayout(quitButton);

        setTitle("Quit button");
        setSize(300, 200);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }

    private void createLayout(JComponent..... arg) {

        Container pane = getContentPane();
        GroupLayout gl = new GroupLayout(pane);
        pane.setLayout(gl);

        gl.setAutoCreateContainerGaps(true);

        gl.setHorizontalGroup(gl.createSequentialGroup()
                .addComponent(arg[0])
        );

        gl.setVerticalGroup(gl.createSequentialGroup()
                .addComponent(arg[0])
        );
    }

    public static void main(String[] args) {

        EventQueue.invokeLater(new Runnable() {

            @Override
            public void run() {
                QuitButtonEx ex = new QuitButtonEx();
                ex.setVisible(true);
            }
        });
    }
}
```

Βάζουμε ένα `JButton` στο παράθυρο και προσθέτουμε ένα action listener σε αυτό το κουμπί

```
JButton quitButton = new JButton("Quit");
```

Εδώ δημιουργούμε ένα button component. Αυτός ο δημιουργός παίρνει σαν παράμετρο την ετικέτα του κουμπιού.

```
quitButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent event) {
        System.exit(0);
    }
});
```

Εδώ βάζουμε τον action listener στο κουμπί. Η μέθοδος είναι η `actionPerformed()` που θα καλεστεί όταν κάνουμε κλικ στο κουμπί. Η δράση αυτή τερματίζει την εφαρμογή καλώντας την μέθοδο `System.exit()`.

```
createLayout(quitButton);
```

Τα child components χρειάζεται να τοποθετηθούν σε ένα container. Αναθέτουμε αυτό το έργο στη μέθοδο `createLayout()`.

```
Container pane = getContentPane();
GroupLayout gl = new GroupLayout(pane);
pane.setLayout(gl);
```

Το τμήμα περιεχομένου του παραθύρου (content pane) ενός `JFrame` δηλαδή είναι μία περιοχή τοποθετούνται διάφορα child components. Τα διάφορα child components είναι οργανωμένα με ειδικά «μη-ορατά» components που ονομάζονται layout managers. Το default layout manager ενός τμήματος περιεχομένου παραθύρου, δηλαδή ενός content pane είναι το `BorderLayout` manager. Αυτός ο manager είναι πολύ απλός και χρήσιμος μόνο σε κάποιες ειδικές περιπτώσεις. Εδώ θα χρησιμοποιήσουμε το `GroupLayout` manager που είναι πολύ πιο δυνατός και ευέλικτος.

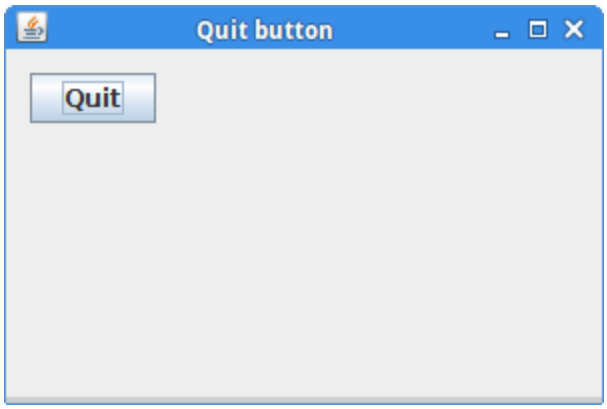
```
gl.setAutoCreateContainerGaps(true);
```

Η μέθοδος `setAutoCreateContainerGaps()` δημιουργεί κενά ανάμεσα στα components και στις άκρες του container. Τα κενά ανάμεσα στα διάφορα tools ή components είναι πολύ σημαντικό κομμάτι στο σχεδιασμό κάθε εφαρμογής.

```
gl.setHorizontalGroup(gl.createSequentialGroup()
        .addComponent(quitButton)
);

gl.setVerticalGroup(gl.createSequentialGroup()
        .addComponent(quitButton)
);
```

Ο `GroupLayout` manager καθορίζει τη διάταξη των components για κάθε διάσταση ξεχωριστά. Στο πρώτο βήμα, βάζουμε τα διάφορα components μαζί στον οριζόντιο άξονα. Στο

επόμενο βήμα, βάζουμε τα διάφορα components μαζί στον κάθετο άξονα. Και στα δύο είδη σχεδιασμού μπορούμε να οργανώσουμε τα components είτε διαδοχικά είτε παράλληλα. Σε μια οριζόντια διάταξη, μία σειρά από components ονομάζεται διαδοχική ομάδα και μια στήλη από components ονομάζεται μια παράλληλη ομάδα. Σε μια κάθετη διάταξη, μια στήλη από components ονομάζεται διαδοχική ομάδα και μια σειρά από components μια παράλληλη ομάδα. Στο παράδειγμα μας έχουμε μόνο ένα κουμπί επομένως ο σχεδιασμός δηλαδή το layout είναι πολύ απλό. Για κάθε διάσταση, καλούμε την μέθοδο `addComponent()` με το κουμπί σαν παράμετρο. (Κάθε child component πρέπει να προστεθεί και στις δύο διαστάσεις.)



Διάγραμμα 2: Το Quit button

## Η χρήση των tooltips

Τα tooltips είναι κομμάτι του εσωτερικού συστήματος βοήθειας των εφαρμογών. Η Swing δείχνει ένα μικρό παράθυρο αν σύρουμε πάνω από κάποιο component το mouse pointer το οποίο δείχνει ένα μικρό μήνυμα σχετικό με το component.

```
import java.awt.EventQueue;
import javax.swing.GroupLayout;
import javax.swing.JButton;
import javax.swing.JComponent;
import javax.swing.JFrame;
import javax.swing.JPanel;


public class TooltipEx extends JFrame {

    public TooltipEx() {

        initUI();
    }

    private void initUI() {

        JButton btn = new JButton("Button");
        btn.setToolTipText("A button component");

        createLayout(btn);

        setTitle("Tooltip");
```

```
        setLocationRelativeTo(null);
        setDefaultCloseOperation(EXIT ON CLOSE);
    }

    private void createLayout(JComponent... arg) {

        JPanel pane = (JPanel) getContentPane();
        GroupLayout gl = new GroupLayout(pane);
        pane.setLayout(gl);

        pane.setToolTipText("Content pane");

        gl.setAutoCreateContainerGaps(true);

        gl.setHorizontalGroup(gl.createSequentialGroup()
                .addComponent(arg[0])
                .addGap(200)
        );

        gl.setVerticalGroup(gl.createSequentialGroup()
                .addComponent(arg[0])
                .addGap(120)
        );

        pack();
    }

    public static void main(String[] args) {

        EventQueue.invokeLater(new Runnable() {

            @Override
            public void run() {
                TooltipEx ex = new TooltipEx();
                ex.setVisible(true);
            }
        });
    }
}
```

Στο παράδειγμα λοιπόν θέτουμε ένα tooltip για το frame και για το button.

```
btn.setToolTipText("A button component");
```

Για να ενεργοποιήσουμε ένα tooltip, καλούμε την μέθοδο `setTooltipText()`.

```
JPanel pane = (JPanel) getContentPane();
GroupLayout gl = new GroupLayout(pane);
pane.setLayout(gl);
```

Ένα τμήμα περιεχομένου του παραθύρου (content pane) είναι ένα στιγμιότυπο του component `JPanel`. Η μέθοδος `getContentPane()` επιστρέφει ένα τύπο δεδομένων τύπου `Container`. Επειδή το σετάρισμα ενός tooltip απαιτεί ένα στιγμιότυπο τύπου `JComponent`, κάνουμε cast το αντικείμενο σε ένα αντικείμενο τύπου `JPanel`.

```
pane.setToolTipText("Content pane");
```

Το tooltip σετάρεται για το content pane.

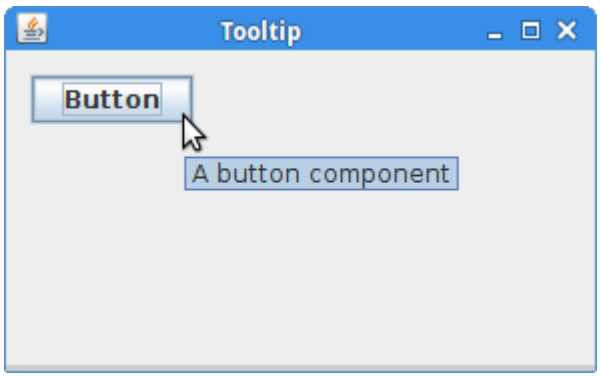
```
gl.setHorizontalGroup(gl.createSequentialGroup()
        .addComponent(arg[0])
        .addGap(200)
```

```
);

gl.setVerticalGroup(gl.createSequentialGroup()
        .addComponent(arg[0])
        .addGap(120)
);
```

Καλούμε την μέθοδο `addGap()` για οριζόντιες και κάθετες διαστάσεις. Αυτή δημιουργεί κάποιο χώρο στα δεξιά και στο κάτω μέρος από το button.

```
pack();
```

Η μέθοδος `pack()` αυτόματα δίνει διαστάσεις στο `JFrame` ανάλογα με τις διαστάσεις των components που βάζουμε μέσα του και ανάλογα την θέση στην οποία βάζουμε καθένα από τα components. Χρησιμοποιεί επομένως το πρότερα ορισμένο χώρο για τον υπολογισμό αυτό. Το παράθυρο επομένως θα δείξει το button και τα κενά τα οποία έχουμε ορίσει με την μέθοδο `addGap()`.



Διάγραμμα 3: Tooltip

## Η χρήση των Mnemonics

Τα *Mnemonics* είναι shortcut keys τα οποία ενεργοποιούν ένα component το οποίο υποστηρίζει mnemonics. Για παράδειγμα, μπορούν να χρησιμοποιηθούν με labels, buttons, ή κυρίως με menu items.

```
import java.awt.Container;
import java.awt.EventQueue;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import javax.swing.GroupLayout;
import javax.swing.JButton;
import javax.swing.JComponent;
import javax.swing.JFrame;


public class MnemonicEx extends JFrame {

    public MnemonicEx() {

        initUI();
    }
```

```
    private void initUI() {

        JButton btn = new JButton("Button");
        btn.addActionListener(new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent e) {
                System.out.println("Button pressed");
            }

        });

        btn.setMnemonic(KeyEvent.VK_B);

        createLayout(btn);

        setTitle("Mnemonics");
        setLocationRelativeTo(null);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }

    private void createLayout(JComponent... arg) {

        Container pane = getContentPane();
        GroupLayout gl = new GroupLayout(pane);
        pane.setLayout(gl);

        gl.setAutoCreateContainerGaps(true);

        gl.setHorizontalGroup(gl.createSequentialGroup()
                .addComponent(arg[0])
                .addGap(200)
        );

        gl.setVerticalGroup(gl.createParallelGroup()
                .addComponent(arg[0])
                .addGap(200)
        );

        pack();
    }

    public static void main(String[] args) {

        EventQueue.invokeLater(new Runnable() {

            @Override
            public void run() {
                MnemonicEx ex = new MnemonicEx();
                ex.setVisible(true);
            }
        });
    }
}
```

Έχουμε ένα button με ένα action listener. Σετάρουμε το mnemonic για αυτό το κουμπί. Μπορεί να ενεργοποιηθεί με το `Alt+B` keyboard shortcut.

```
btn.setMnemonic(KeyEvent.VK_B);
```

Η μέθοδος `setMnemonic()` σετάρει ένα keyboard mnemonic για ένα κουμπί. Το συγκεκριμένο κλειδί που θα βάλουμε το παίρνουμε μέσω ενός κωδικού από την κλάση `KeyEvent`. Συνήθως τα mnemonics ξεκινούν με τη χρήση του κλειδιού ALT = κάποιο άλλο κλειδί. Σε αυτό το σημείο μπορούμε να πούμε ότι υπάρχουν τρεις τρόποι για να

ενεργοποιήσουμε ένα κουμπί: το αριστερό mouse button click, το `Alt+B shortcut`, και το `Space` key (θεωρώντας ότι εκείνη τη στιγμή το κουμπί έχει το focus δηλαδή είναι εστιασμένο).

## Τα Menus και τα toolbars στη Java Swing

Ένα menu είναι μία ομάδα από εντολές (commands) οι οποίες βρίσκονται σε μία μπάρα μενού (menubar). Μία μπάρα εργαλείων (toolbar) έχει κάποια κουμπιά με κάποιες κοινές εντολές στην εφαρμογή. Για να δημιουργήσουμε ένα menubar στη Swing, χρησιμοποιούμε τρία αντικείμενα: ένα `JMenuBar`, ένα `JMenu` και ένα `JMenuItem`.

## Ένα απλό menu

```
import java.awt.EventQueue;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;

import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;

public class SimpleMenuEx extends JFrame {

    public SimpleMenuEx() {

        initUI();
    }

    private void initUI() {

        createMenuBar();

        setTitle("Simple menu");
        setSize(300, 200);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }

    private void createMenuBar() {

        JMenuBar menubar = new JMenuBar();
        ImageIcon icon = new ImageIcon("exit.png");

        JMenu file = new JMenu("File");
        file.setMnemonic(KeyEvent.VK F);

        JMenuItem eMenuItem = new JMenuItem("Exit", icon);
        eMenuItem.setMnemonic(KeyEvent.VK_E);
        eMenuItem.setToolTipText("Exit application");
        eMenuItem.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent event) {
                System.exit(0);
            }
        });

        file.add(eMenuItem);
        menubar.add(file);

        setJMenuBar(menubar);
```

```
    }

    public static void main(String[] args) {

        EventQueue.invokeLater(new Runnable() {

            @Override
            public void run() {
                SimpleMenuEx ex = new SimpleMenuEx();
                ex.setVisible(true);
            }
        });
    }
}
```

Στο παράδειγμά μας θα δείξουμε ένα μενού με μόνο ένα στοιχείο. Επιλέγοντας το exit η εφαρμογή θα κλείνει.

```
JMenuBar menubar = new JMenuBar();
```

Ένα menubar δημιουργείται με την κλάση `JMenuBar`.

```
ImageIcon icon = new ImageIcon("exit.png");
```

Το exit icon εμφανίζεται στο menu.

```
JMenu file = new JMenu("File");
file.setMnemonic(KeyEvent.VK_F);
```

Ένα αντικείμενο του μενού δημιουργείται με την κλάση `JMenu`. Στα μενού έχουμε πρόσβαση μέσω του πληκτρολογίου επίσης. Για να δεσμεύσουμε ένα μενού σε ένα συγκεκριμένο κλειδί, χρησιμοποιούμε τη μέθοδο `setMnemonic ()`. Στην περίπτωσή μας, το μενού μπορεί να ανοίξει με τη συντόμευση Alt + F.

```
JMenuItem eMenuItem = new JMenuItem("Exit", icon);
eMenuItem.setMnemonic(KeyEvent.VK_E);
```

Ένα αντικείμενο μενού αποτελείται από τα διάφορα στοιχεία του μενού. Ένα στοιχείο μενού έχει δημιουργηθεί με την κλάση `JMenuItem`. Ένα στοιχείο του μενού έχει το δικό του mnemonic. Μπορεί να ενεργοποιηθεί με το συνδυασμό πλήκτρων Alt + F + E.

```
eMenuItem.setToolTipText("Exit application");
```

Αυτή η γραμμή κώδικα δημιουργεί ένα tooltip για ένα στοιχείο μενού.

```
eMenuItem.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent event) {
        System.exit(0);
    }
});
```

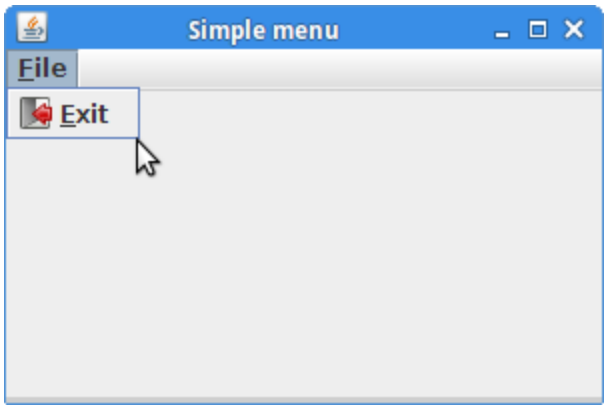
Το `JMenuItem` είναι ένα ειδικό button component. Προσθέτουμε ένα action listener σε αυτό το οποίο τερματίζει την εφαρμογή.

```
file.add(eMenuItem);
menubar.add(file);
```

Το στοιχείο μενού (menu item) προστίθεται στο αντικείμενο μενού (menu object) και το αντικείμενο μενού εισάγεται στην γραμμή μενού (`menubar`).

```
setJMenuBar(menubar);
```

Η μέθοδος `setJMenuBar()` σετάρει το menubar για το `JFrame` container.



Διάγραμμα 4: Ένα απλό μενού

## Υπομενού (Submenu)

Κάθε μενού μπορεί επίσης να έχει ένα υπομενού. Με αυτό τον τρόπο μπορούμε να βάλουμε παρόμοιες εντολές σε ομάδες εντολών. Για παράδειγμα, μπορούμε να τοποθετήσουμε τις εντολές που κρύβουν και δείχνουν διάφορες γραμμές εργαλείων, όπως η προσωπική γραμμή, η γραμμή διευθύνσεων, η γραμμή κατάστασης ή η μπάρα πλοήγησης σε ένα υπομενού που ονομάζεται γραμμές εργαλείων. Μέσα σε ένα μενού, μπορούμε να χωρίσουμε τις εντολές με ένα διαχωριστικό. Ο διαχωριστής είναι μια απλή γραμμή. Είναι κοινή πρακτική να διαχωρίζουμε εντολές όπως είναι το New, το Open το Print μαζί με το Print View κλπ. Επιπροσθέτως με τα mnemonics, οι εντολές μενού μπορούν να εκτελεστούν με τη χρήση των επιταχυντών (accelerators).

```
import java.awt.EventQueue;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
```

```
public class SubmenuEx extends JFrame {

    public SubmenuEx() {

        initUI();
    }

    private void initUI() {

        createMenuBar();

        setTitle("Submenu");
        setSize(360, 250);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }

    private void createMenuBar() {

        JMenuBar menubar = new JMenuBar();

        ImageIcon iconNew = new ImageIcon("new.png");
        ImageIcon iconOpen = new ImageIcon("open.png");
        ImageIcon iconSave = new ImageIcon("save.png");
        ImageIcon iconExit = new ImageIcon("exit.png");

        JMenu fileMenu = new JMenu("File");

        JMenu impMenu = new JMenu("Import");

        JMenuItem newsfMi = new JMenuItem("Import newsfeed list...");
        JMenuItem bookmMi = new JMenuItem("Import bookmarks...");
        JMenuItem mailMi = new JMenuItem("Import mail...");

        impMenu.add(newsfMi);
        impMenu.add(bookmMi);
        impMenu.add(mailMi);

        JMenuItem newMi = new JMenuItem("New", iconNew);
        JMenuItem openMi = new JMenuItem("Open", iconOpen);
        JMenuItem saveMi = new JMenuItem("Save", iconSave);

        JMenuItem exitMi = new JMenuItem("Exit", iconExit);
        exitMi.setToolTipText("Exit application");

        exitMi.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent event) {
                System.exit(0);
            }
        });

        fileMenu.add(newMi);
        fileMenu.add(openMi);
        fileMenu.add(saveMi);
        fileMenu.addSeparator();
        fileMenu.add(impMenu);
        fileMenu.addSeparator();
        fileMenu.add(exitMi);

        menubar.add(fileMenu);

        setJMenuBar(menubar);
    }

    public static void main(String[] args) {

        EventQueue.invokeLater(new Runnable() {
            @Override
            public void run() {

                SubmenuEx ex = new SubmenuEx();
                ex.setVisible(true);
            }
        });
    }
```

```
}
```

Αυτό το παράδειγμα δημιουργεί ένα υπομενού και διαχωρίζει τις ομάδες των στοιχείων του μενού με ένα διαχωριστικό μενού (separator).

```
JMenu impMenu = new JMenu("Import");
...
fileMenu.add(impMenu);
```

Ένα υπομενού είναι ακριβώς όπως οποιοδήποτε άλλο κανονικό μενού. Δημιουργείται με τον ίδιο τρόπο. Εμείς απλά προσθέτουμε ένα μενού στο υπάρχον μενού.

```
exitMi.setToolTipText("Exit application");
```

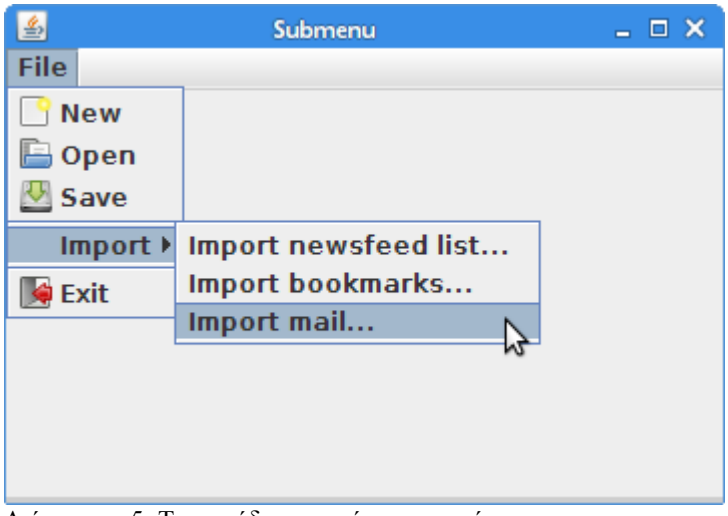
Ένα tooltip σετάρεται με το μενού Exit με τη μέθοδο `setToolTipText()`.

```
JMenuItem newMi = new JMenuItem("New", iconNew);
```

Αυτός ο δημιουργός `JMenuItem` δημιουργεί ένα στοιχείο μενού που περιέχει μία ετικέτα και ένα εικονίδιο.

```
fileMenu.addSeparator();
```

Ο διαχωριστής (separator) είναι μία οριζόντια γραμμή που οπτικά διαχωρίζει κάποια στοιχεία μενού από κάποια άλλα. Με αυτό τον τρόπο μπορούμε να οργανώσουμε τα στοιχεία του μενού σε ομάδες με βάση κάποια λογική και με βάση κάποια σειρά.



Διάγραμμα 5: Το παράδειγμα ενός υπομενού

## Mnemonics και accelerators

Τα Mnemonics και τα accelerators είναι κλειδιά συντόμευσης (shortcut keys) τα οποία ενεργοποιούν εντολές μέσω πληκτρολογίου. Τα Mnemonics κάνουν περιήγηση στην ιεραρχία

του μενού για να επιλέξουμε ένα συγκεκριμένο στοιχείο μενού, ενώ οι επιταχυντές κάνουν παράκαμψη την ιεραρχία μενού και ενεργοποιούν άμεσα το στοιχείο του μενού.

Το ακόλουθο παράδειγμα χρησιμοποιεί ενέργειες, οι οποίες είναι αντικείμενα που μπορούν να μοιραστούν σε διάφορα components που χρειάζονται την ίδια λειτουργικότητα

```
import java.awt.EventQueue;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import javax.swing.AbstractAction;
import static javax.swing.Action.MNEMONIC_KEY;
import static javax.swing.Action.SMALL_ICON;
import javax.swing.ImageIcon;
import javax.swing.JFrame;
import static javax.swing.JFrame.EXIT ON CLOSE;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.KeyStroke;


public class ShortCutsEx extends JFrame {

    public ShortCutsEx() {

        initUI();
    }

    private void initUI() {

        createMenuBar();

        setTitle("Mnemonics and accelerators");
        setSize(360, 250);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(EXIT ON CLOSE);
    }

    private void createMenuBar() {

        JMenuBar menubar = new JMenuBar();

        ImageIcon iconNew = new ImageIcon("new.png");
        ImageIcon iconOpen = new ImageIcon("open.png");
        ImageIcon iconSave = new ImageIcon("save.png");
        ImageIcon iconExit = new ImageIcon("exit.png");

        JMenu fileMenu = new JMenu("File");
        fileMenu.setMnemonic(KeyEvent.VK_F);

        JMenuItem newMi = new JMenuItem(new MenuItemAction("New", iconNew,
                KeyEvent.VK N));

        JMenuItem openMi = new JMenuItem(new MenuItemAction("Open", iconOpen,
                KeyEvent.VK_O));

        JMenuItem saveMi = new JMenuItem(new MenuItemAction("Save", iconSave,
                KeyEvent.VK S));

        JMenuItem exitMi = new JMenuItem("Exit", iconExit);
        exitMi.setMnemonic(KeyEvent.VK_E);
        exitMi.setToolTipText("Exit application");
        exitMi.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK W,
            ActionEvent.CTRL MASK));

        exitMi.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent event) {
                System.exit(0);
```

```
            }
        });

        fileMenu.add(newMi);
        fileMenu.add(openMi);
        fileMenu.add(saveMi);
        fileMenu.addSeparator();
        fileMenu.add(exitMi);

        menubar.add(fileMenu);

        setJMenuBar(menubar);
    }

    private class MenuItemAction extends AbstractAction {

        public MenuItemAction(String text, ImageIcon icon,
                Integer mnemonic) {
            super(text);

            putValue(SMALL_ICON, icon);
            putValue(MNEMONIC KEY, mnemonic);
        }

        @Override
        public void actionPerformed(ActionEvent e) {

            System.out.println(e.getActionCommand());
        }
    }

    public static void main(String[] args) {

        EventQueue.invokeLater(new Runnable() {
            @Override
            public void run() {

                ShortCutsEx ex = new ShortCutsEx();
                ex.setVisible(true);
            }
        });
    }
}
```

Το παράδειγμα έχει αρκετά μνημονικά και έναν επιταχυντή. Τρία στοιχεία μενού μοιράζονται ένα συγκεκριμένο αντικείμενο δράσης. Επιλέγοντας αυτά τα τρία στοιχεία του μενού προκαλεί τη εντολή της δράσης τους να εκτυπωθεί στην κονσόλα.

```
JMenu fileMenu = new JMenu("File");
fileMenu.setMnemonic(KeyEvent.VK_F);
```

Ένα μνημονικό σετάρεται στο File menu. Το μενού μπορεί να ενεργοποιηθεί πλέον με το κλειδί `Alt+F`.

```
JMenuItem newMi = new JMenuItem(new MenuItemAction("New", iconNew,
        KeyEvent.VK_N));
```

Το μενού New παίρνει ένα action object σαν παράμετρο. Ο δημιουργός του παίρνει μία ετικέτα, ένα icon, και ένα mnemonic ως παραμέτρους.

```
exitMi.setMnemonic(KeyEvent.VK_E);
```

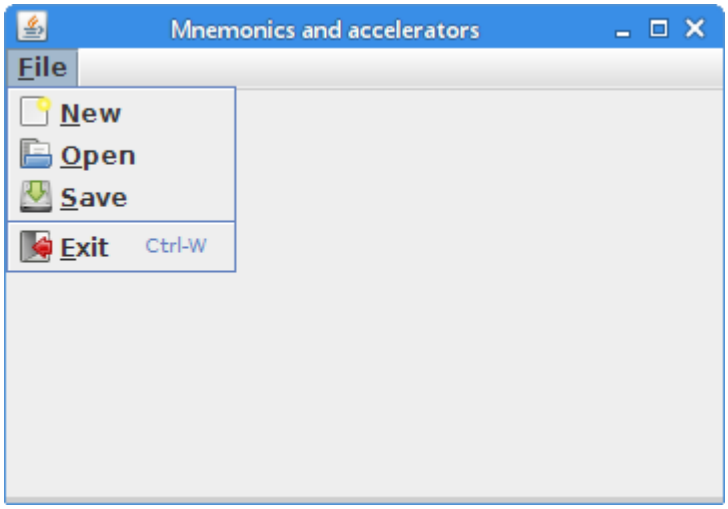
Το Exit menu δεν χρησιμοποιεί ένα action object. Η λειτουργία τους δημιουργείται με διαφορετικό τρόπο. Καλούμε την μέθοδο `setMnemonic()` για να σετάρουμε το κλειδί μνημονικού. Για να χρησιμοποιήσουμε ένα μνημονικό, το component πρέπει να γίνεται ορατό στην οθόνη. Επομένως πρέπει πρώτα να ενεργοποιήσουμε το αντικείμενο του μενού το οποίο κάνει ορατό το μενού Exit και μετά να ενεργοποιήσουμε αυτό το συγκεκριμένο στοιχείο μενού. Αυτό σημαίνει ότι αυτό το στοιχείο μενού ενεργοποιείται με τον συνδυασμό `Alt+F+E`.

```
exitMi.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_W,
    ActionEvent.CTRL_MASK));
```

Ένας επιταχυντής (accelerator) είναι ένα κλειδί συντόμευσης (shortcut) το οποίο κατευθείαν ξεκινά την εντολής εκτέλεσης ενός στοιχείου μενού. Στην δική μας περίπτωση, πληκτρολογώντας το κλειδί `Ctrl+W` κατευθείαν κλείνουμε την εφαρμογή. Ο accelerator σετάρεται με την μέθοδο `setAccelerator()`.

```
private class MenuItemAction extends AbstractAction {

    public MenuItemAction(String text, ImageIcon icon,
            Integer mnemonic) {
        super(text);

        putValue(SMALL_ICON, icon);
        putValue(MNEMONIC_KEY, mnemonic);
    }

    @Override
    public void actionPerformed(ActionEvent e) {

        System.out.println(e.getActionCommand());
    }
}
```

Ένα στιγμιότυπο αυτής της κλάσης δράσης (action class) μοιράζεται σε τρία στοιχεία μενού. Οι δράσεις χρησιμοποιούν διάφορα κλειδιά για να ορίσουν την λειτουργικότητά τους. Η μέθοδος `putValue()` αντιστοιχεί τις τιμές των συμβολοσειρών με τα αντίστοιχα κλειδιά.




Διάγραμμα 6: Το παράδειγμα για τη χρήση Mnemonics και accelerators

Τα μνημονικά είναι «κρυμμένα» σαν υπογραμμισμένοι χαρακτήρες ενώ οι επιταχυντές έχουν διπλά στην ετικέτα τους το κλειδί της συντόμευσης.

## Το component JCheckBoxMenuItem

Το `JCheckBoxMenuItem` είναι ένα στοιχείο μενού που μπορούμε να διαλέγουμε (να επιλέγουμε με check) ή να από-επιλέγουμε (να αποδεσμεύουμε την επιλογή αυτή). Αν επιλεχθεί, το στοιχείο μενού αυτό εμφανίζεται με ένα checkmark δίπλα του. Αν «ξετσεκάρουμε» την επιλογή εμφανίζεται χωρίς το checkmark.

```
import java.awt.BorderLayout;
import java.awt.EventQueue;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import java.awt.event.KeyEvent;
import javax.swing.BorderFactory;
import javax.swing.JCheckBoxMenuItem;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JMenu;
import javax.swing.JMenuBar;

public class CheckBoxMenuItemEx extends JFrame {

    private JLabel statusbar;

    public CheckBoxMenuItemEx() {

        initUI();
    }

    private void initUI() {

        createMenuBar();

        statusbar = new JLabel("Ready");
        statusbar.setBorder(BorderFactory.createEtchedBorder());
        add(statusbar, BorderLayout.SOUTH);

        setTitle("JCheckBoxMenuItem");
        setSize(360, 250);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }

    private void createMenuBar() {

        JMenuBar menubar = new JMenuBar();
        JMenu fileMenu = new JMenu("File");
        fileMenu.setMnemonic(KeyEvent.VK F);

        JMenu viewMenu = new JMenu("View");
        viewMenu.setMnemonic(KeyEvent.VK_V);

        JCheckBoxMenuItem sbarMi = new JCheckBoxMenuItem("Show statubar");
        sbarMi.setMnemonic(KeyEvent.VK S);
        sbarMi.setDisplayedMnemonicIndex(5);
        sbarMi.setSelected(true);

        sbarMi.addItemListener(new ItemListener() {

            @Override
            public void itemStateChanged(ItemEvent e) {

                if (e.getStateChange() == ItemEvent.SELECTED) {
```

```
                    statusbar.setVisible(true);
                } else {
                    statusbar.setVisible(false);
                }

            }

        });

        viewMenu.add(sbarMi);

        menubar.add(fileMenu);
        menubar.add(viewMenu);

        setJMenuBar(menubar);
    }

    public static void main(String[] args) {

        EventQueue.invokeLater(new Runnable() {

            @Override
            public void run() {
                CheckBoxMenuItemEx ex = new CheckBoxMenuItemEx();
                ex.setVisible(true);
            }
        });
    }
}
```

Το παράδειγμα αυτό χρησιμοποιεί ένα `JCheckBoxMenuItem` για εναλλαγή της προβολής μιας γραμμής κατάστασης (statusbar).

```
statusbar = new JLabel("Ready");
statusbar.setBorder(BorderFactory.createEtchedBorder());
add(statusbar, BorderLayout.SOUTH);
```

Η γραμμή κατάστασης είναι ένα απλό component τύπου `JLabel`. Απλώς βάλαμε ένα ανυψωμένο `EtchedBorder` γύρω από την ετικέτα έτσι ώστε να είναι ορατό.

```
JCheckBoxMenuItem sbarMi = new JCheckBoxMenuItem("Show statubar");
sbarMi.setMnemonic(KeyEvent.VK_S);
sbarMi.setDisplayedMnemonicIndex(5);
```

`JCheckBoxMenuItem` creates a check box menu item. There are two s letters in the label, therefore, we use the `setDisplayedMnemonicIndex()` method to choose which one is going to be underlined. We chose the second one.

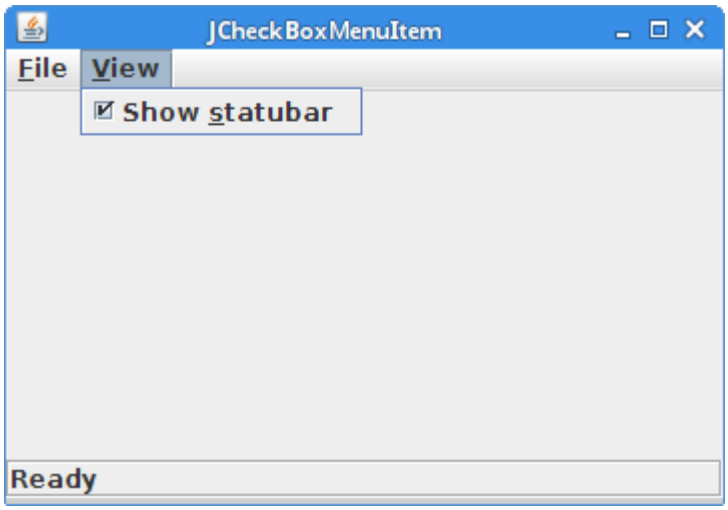
```
sbarMi.setSelected(true);
```

Επειδή η γραμμή κατάστασης είναι αρχικά ορατή, καλούμε τη μέθοδο `JCheckBoxMenuItem setSelected()` για να το επιλέξουμε.

```
sbarMi.addItemListener(new ItemListener() {

    @Override
    public void itemStateChanged(ItemEvent e) {

        if (e.getStateChange() == ItemEvent.SELECTED) {
            statusbar.setVisible(true);
```

```
        } else {
            statusbar.setVisible(false);
        }

    }

});
```

Το `JCheckBoxMenuItem` είναι ένα ειδικό button component. Εφαρμόζει το `ItemSelectable` interface. Για αυτό το λόγο χρησιμοποιείται ένας `ItemListener` για να «ακούει» της αλλαγές κατάστασης του. Ανάλογα με την κατάστασή του, δείχνουμε ή κρύβουμε την μπάρα κατάστασης στο κάτω μέρος του Frame.



Διάγραμμα 7: Το στοιχείο μενού CheckBox

## Menu στα δεξιά της γραμμής

Μερικές εφαρμογές περιλαμβάνουν ένα στοιχείο μενού τελείως δεξιά της γραμμής μενού. Συνήθως αυτό το στοιχείο μενού αντιστοιχεί στο Help menu.

```
import java.awt.EventQueue;
import javax.swing.Box;
import javax.swing.JFrame;
import javax.swing.JMenu;
import javax.swing.JMenuBar;


public class RightMenuEx extends JFrame {

    public RightMenuEx() {

        initUI();
    }

    private void initUI() {

        createMenuBar();

        setTitle("Right menu");
        setSize(300, 200);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }
```

```
    private void createMenuBar() {

        JMenuBar menubar = new JMenuBar();

        JMenu fileMenu = new JMenu("File");
        JMenu viewMenu = new JMenu("View");
        JMenu toolsMenu = new JMenu("Tools");
        JMenu helpMenu = new JMenu("Help");

        menubar.add(fileMenu);
        menubar.add(viewMenu);
        menubar.add(toolsMenu);
        menubar.add(Box.createHorizontalGlue());
        menubar.add(helpMenu);

        setJMenuBar(menubar);
    }

    public static void main(String[] args) {

        EventQueue.invokeLater(new Runnable() {
            @Override
            public void run() {
                RightMenuEx ex = new RightMenuEx();
                ex.setVisible(true);
            }
        });
    }
}
```

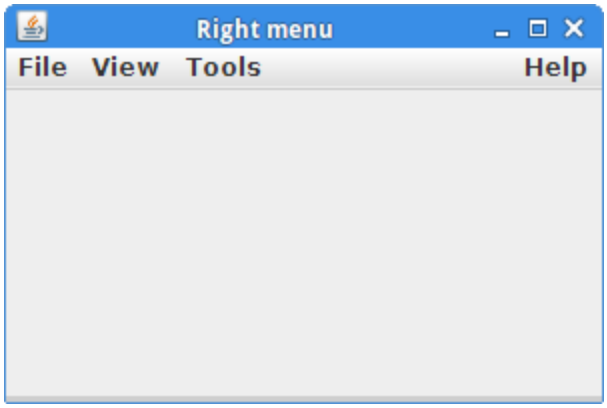
Το παράδειγμα δείχνει τρία μενού στα αριστερά και ένα μενού στα δεξιά.

```
JMenuBar menubar = new JMenuBar();

JMenu fileMenu = new JMenu("File");
JMenu viewMenu = new JMenu("View");
JMenu toolsMenu = new JMenu("Tools");
JMenu helpMenu = new JMenu("Help");
```

Εδώ δημιουργείται η menubar και τέσσερα menu objects.

```
menubar.add(fileMenu);
menubar.add(viewMenu);
menubar.add(toolsMenu);
menubar.add(Box.createHorizontalGlue());
menubar.add(helpMenu);
```

Αφού προστεθούν τα τρία μενού, προσθέτουμε την «οριζόντια κόλλα» (horizontal glue) στο menubar. Η κόλλα απορροφά όλο τον επιπρόσθετο χώρο που υπάρχει. Αυτό κάνει το μενού help να σπρώχνεται στα δεξιά του menubar.



Διάγραμμα 8: Το Help menu στα δεξιά

## Το στοιχείο εργαλειοθήκη (JToolbar)

Τα μενού ομαδοποιούν εντολές οι οποίες είναι συναφείς και οι οποίες χρησιμοποιούνται σε μία εφαρμογή. Οι εργαλειοθήκες (Toolbars) πραγματικά δίνουν ένα γρήγορο τρόπο πρόσβασης στις πιο χρησιμοποιούμενες εντολές. Για να δημιουργήσουμε μία εργαλειοθήκη χρησιμοποιούμε στη Swing το component `JToolBar`.

```
import java.awt.BorderLayout;
import java.awt.EventQueue;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JToolBar;


public class ToolbarEx extends JFrame {

    public ToolbarEx() {

        initUI();
    }

    private void initUI() {

        createMenuBar();
        createToolBar();

        setTitle("Simple toolbar");
        setSize(300, 200);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(EXIT ON CLOSE);
    }

    private void createMenuBar() {

        JMenuBar menubar = new JMenuBar();
        JMenu file = new JMenu("File");
        menubar.add(file);
        setJMenuBar(menubar);
    }

    private void createToolBar() {

        JToolBar toolbar = new JToolBar();
```

```
        ImageIcon icon = new ImageIcon("exit.png");

        JButton exitButton = new JButton(icon);
        toolbar.add(exitButton);

        exitButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent event) {
                System.exit(0);
            }
        });

        add(toolbar, BorderLayout.NORTH);
    }

    public static void main(String[] args) {

        EventQueue.invokeLater(new Runnable() {

            @Override
            public void run() {
                ToolbarEx ex = new ToolbarEx();
                ex.setVisible(true);
            }
        });
    }
}
```

Το παράδειγμα δημιουργεί μία εργαλειοθήκη η οποία περιέχει ένα exit button.

```
JToolBar toolbar = new JToolBar();
```

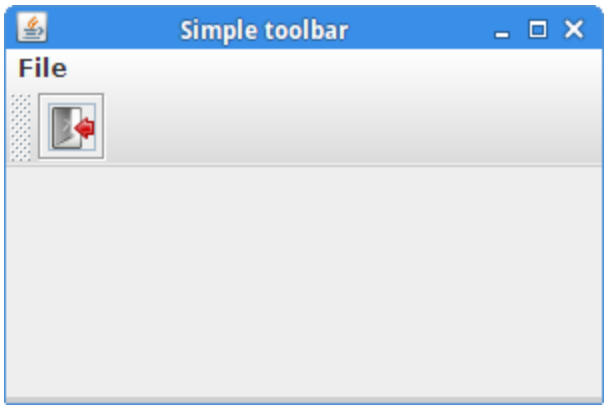
Η εργαλειοθήκη δημιουργείται με το component `JToolBar`.

```
JButton exitButton = new JButton(icon);
toolbar.add(exitButton);
```

Δημιουργούμε ένα button και το προσθέτουμε στο toolbar.

```
add(toolbar, BorderLayout.NORTH);
```

Η εργαλειοθήκη τοποθετείται στη «βόρεια περιοχή» (north area) του `BorderLayout`.



Διάγραμμα 9: Μία απλή εργαλειοθήκη

## Πολλαπλές Εργαλειοθήκες (Multiple Toolbars)

Συχνά μπορεί να χρειάζεται να χρησιμοποιηθούν πολλαπλές εργαλειοθήκες σε ένα παράθυρο.

```
import java.awt.Container;
import java.awt.EventQueue;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.GroupLayout;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JComponent;
import javax.swing.JFrame;
import javax.swing.JToolBar;


public class ToolbarsEx extends JFrame {

    public ToolbarsEx() {

        initUI();
    }

    public final void initUI() {

        createToolBars();

        setTitle("Toolbars");
        setSize(360, 250);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(EXIT ON CLOSE);
    }

    private void createToolBars() {

        JToolBar toolbar1 = new JToolBar();
        JToolBar toolbar2 = new JToolBar();

        ImageIcon newi = new ImageIcon("new.png");
        ImageIcon open = new ImageIcon("open.png");
        ImageIcon save = new ImageIcon("save.png");
        ImageIcon exit = new ImageIcon("exit.png");

        JButton newb = new JButton(newi);
        JButton openb = new JButton(open);
        JButton saveb = new JButton(save);

        toolbar1.add(newb);
        toolbar1.add(openb);
        toolbar1.add(saveb);

        JButton exitb = new JButton(exit);
        toolbar2.add(exitb);

        exitb.addActionListener(new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent event) {
                System.exit(0);
            }

        });

        createLayout(toolbar1, toolbar2);
    }

    private void createLayout(JComponent... arg) {

        Container pane = getContentPane();
        GroupLayout gl = new GroupLayout(pane);
        pane.setLayout(gl);

        gl.setHorizontalGroup(gl.createParallelGroup()
                .addComponent(arg[0], GroupLayout.DEFAULT SIZE,
                        GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(arg[1], GroupLayout.DEFAULT_SIZE,
```

```
                        GroupLayout.DEFAULT SIZE, Short.MAX VALUE)
        );

        gl.setVerticalGroup(gl.createSequentialGroup()
                .addComponent(arg[0])
                .addComponent(arg[1])
        );
    }

    public static void main(String[] args) {

        EventQueue.invokeLater(new Runnable() {

            @Override
            public void run() {
                ToolbarsEx ex = new ToolbarsEx();
                ex.setVisible(true);
            }
        });
    }
}
```

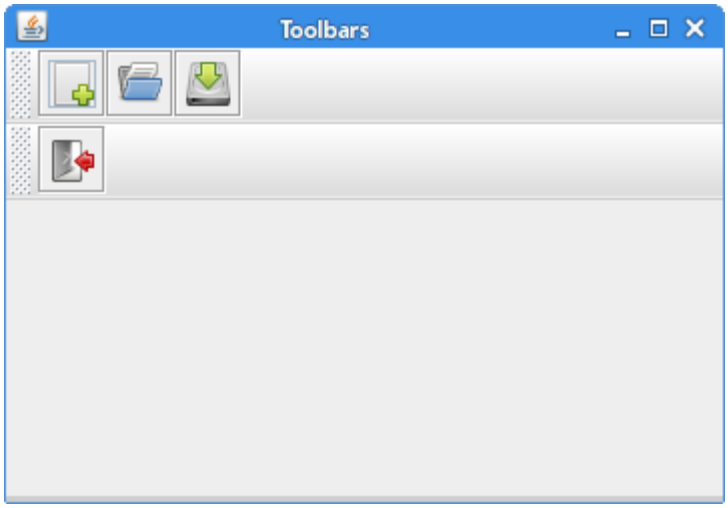
Εδώ εμφανίζονται δύο εργαλειοθήκες στο πάνω μέρος του παραθύρου.

```
JToolBar toolbar1 = new JToolBar();
JToolBar toolbar2 = new JToolBar();
```

Εδώ δημιουργούνται δύο αντικείμενα τύπου `Jtoolbar`

```
private void createLayout(JComponent... arg) {

    Container pane = getContentPane();
    GroupLayout gl = new GroupLayout(pane);
    pane.setLayout(gl);

    gl.setHorizontalGroup(gl.createParallelGroup()
            .addComponent(arg[0], GroupLayout.DEFAULT_SIZE,
                    GroupLayout.DEFAULT SIZE, Short.MAX VALUE)
            .addComponent(arg[1], GroupLayout.DEFAULT SIZE,
                    GroupLayout.DEFAULT SIZE, Short.MAX VALUE)
    );

    gl.setVerticalGroup(gl.createSequentialGroup()
            .addComponent(arg[0])
            .addComponent(arg[1])
    );
}
```

Ο `GroupLayout` manager χρησιμοποιείται για να τοποθετήσουμε τις εργαλειοθήκες στον container.




Διάγραμμα 10: Πολλαπλές εργαλειοθήκες.