ΣΤΑΤΙΚΕΣ ΜΕΘΟΔΟΙ

Στατικές μέθοδοι

- Τι σημαίνει το keyword static στον ορισμό της main μεθόδου? Τι είναι μια στατική μέθοδος?
- Μια στατική μέθοδος μπορεί να κληθεί χωρίς αντικείμενο της κλάσης, χρησιμοποιώντας κατευθείαν το όνομα της κλάσης
  - Η μέθοδος ανήκει στην κλάση και όχι σε κάποιο συγκεκριμένο αντικείμενο.
  - Όταν καλούμε την συνάρτηση main κατά την εκτέλεση του προγράμματος δεν δημιουργούμε κάποιο αντικείμενο της κλάσης
  - Χρήσιμο για τον ορισμό βοηθητικών μεθόδων

Συντακτικό

- Ορισμός

```
class myClass
{
        ...

        public static ReturnType methodName(arguments)
        { … }

        ...
}
```

- Κλήση

```
myClass.methodName(arguments)
```

Παράδειγμα

Ορισμός

```
class Auxiliary
{
    public static int max(int x, int y){
        if (x > y){
            return x;
        }
        return y;
    }
}
```

Κλήση

```
int m = Auxiliary.max(6,5);
```

Η κλήση της μεθόδου max δεν χρειάζεται τον ορισμό αντικείμενου
Γίνεται χρησιμοποιώντας κατευθείαν το όνομα της κλάσης

Παρένθεση

- Ένας άλλος τρόπος να υλοποιήσετε το max τελεστή

```
public static int max(int x, int y){
      return (x>y)? x: y;
}
```

Η έκφραση:

```
condition ? value_if_true: value_if_false
```

επιστέφει μια τιμή ανάλογα με την αποτίμηση του condition και είναι ένας γρήγορος τρόπος να υλοποιήσουμε ένα if το οποίο επιστρέφει μία τιμή

Στατικές μεταβλητές

- Παρόμοια με τις στατικές μεθόδους μπορούμε να ορίσουμε και στατικές μεταβλητές
  - Οι στατικές μεταβλητές ανήκουν στην κλάση και όχι σε κάποιο συγκεκριμένο αντικείμενο και, εφόσον είναι public μπορούμε να έχουμε πρόσβαση σε αυτές χρησιμοποιώντας το όνομα της κλάσης χωρίς να έχουμε ορίσει κάποιο αντικείμενο.

Συντακτικό

- Ορισμός

```
class myClass
{
        public static Type varName;

        public static ReturnType methodName(arguments)
        { … }

        ...
}
```

- Κλήση

```
…. myClass.varName…. ;
```

Παράδειγμα

Ορισμός

```
class Auxiliary
{
    public static int factor = 2.0;

    public static int max(int x, int y){
        if (x > y){
            return x;
        }
        return y;
    }
}
```

Κλήση

```
int m =
    Auxiliary.factor * Auxiliary.max(6,5);
```

Σταθερές

- Οι στατικές μεταβλητές πολλές φορές χρησιμοποιούνται για να ορίσουμε σταθερές.
  - Τις ορίζουμε σε μία κλάση και μπορούμε να τις χρησιμοποιούμε σε διάφορα σημεία στο πρόγραμμα.

- Για να προσδιορίσουμε ότι μία μεταβλητή είναι σταθερά μπορούμε να χρησιμοποιήσουμε το keyword final.

Παράδειγμα

Ορισμός

```
class Circle
{
   public static final double PI = 3.14;

   public static double area(double r){
       return PI*r*r;
   }
}
```

Κλήση

```
int unitCircleArea = Circle.area(1);
System.out.println("PI value is" + Circle.PI);
```

Στατικές μέθοδοι

- Όταν ορίζουμε μια στατική μέθοδο μέσα σε μία κλάση, δεν μπορούμε να χρησιμοποιούμε μη στατικά πεδία, ή να καλούμε μη στατικές μεθόδους.
  - Μη στατικά πεδία και μη στατικές μέθοδοι συσχετίζονται με ένα αντικείμενο. Εφόσον μπορούμε να καλέσουμε μια στατική μέθοδο χωρίς αντικείμενο, δεν μπορούμε μέσα σε αυτή να χρησιμοποιούμε μη στατικά πεδία ή μεθόδους.
  - Σκεφτείτε ότι για κάθε χρήση μιας μεθόδου ή μιας μεταβλητής μπορούμε να βάλουμε το this μπροστά. Αν δεν υπάρχει αντικείμενο η αναφορά this δεν ορίζεται
- Αν θέλουμε να καλέσουμε μια μη στατική μέθοδο θα πρέπει να ορίσουμε ένα αντικείμενο μέσα στην στατική μέθοδο

Παράδειγμα

```
class Auxiliary2
{
        private int x;
        private int y;

        public Auxiliary2(int x, int y){
                this.x = x;
                this.y = y;
        }

        public int max(){
                return (x>y)? x: y;
        }

        public int min(){
                return (x>y)? y: x;
        }

        public static double maxToMin(int x, int y){
                Auxiliary2 aux = new Auxiliary2(x,y);
                return ((double)aux.max())/aux.min();
        }
}
```

Στατικές μεταβλητές

- Εκτός από σταθερές μπορούμε να ορίσουμε στατικές μεταβλητές όταν θέλουμε διαφορετικά αντικείμενα να επικοινωνούν μέσω μιας μεταβλητής
  - Υπάρχει μόνο ένα αντίγραφο μιας στατικής μεταβλητής, άρα όταν το αλλάζει ένα αντικείμενο την αλλαγή την βλέπουν και όλα τα άλλα αντικείμενα της κλάσης.
- Παράδειγμα: Στο πρόγραμμα TakeTurns δείχνουμε πως μπορούμε να χρησιμοποιήσουμε στατικές μεταβλητές για να επικοινωνούν μεταξύ τους τα αντικείμενα.

```
class TakeTurns
{
    private static int players = 0;
    private static int rounds = 0;
    private int id;

    public TakeTurns(int i){
        id = i;
        players ++;
    }

    public void play(){
        if (rounds%players == id){
            System.out.println("Round "+ rounds + " Player " + id + " played");
            rounds ++;
        }
    }

    public static void main(String args[]){
        TakeTurns player0 = new TakeTurns(0);
        TakeTurns player1 = new TakeTurns(1);

        for (int i = 0; i < 10; i ++){
            player0.play();
            player1.play();
        }
    }
}
```

Τα αντικείμενα player0 και player1 βλέπουν τις ίδιες μεταβλητές players και rounds, αλλά διαφορετική μεταβλητή id

Ο κάθε παίχτης παίζει μόνο όταν είναι η σειρά του

Στατικές μέθοδοι και μεταβλητές

- Έχετε ήδη χρησιμοποιήσει στατικές μεθόδους και μεταβλητές σε διάφορες περιπτώσεις

- Παραδείγματα
  - System.out: στατικό πεδίο της κλάσης System, το οποίο κρατάει ένα PrintStream με το οποίο μπορούμε γράψουμε στην οθόνη.
  - System.in: στατικό πεδίο της κλάσης System, το οποίο κρατάει ένα FileInputStream που συνδέεται με το πληκτρολόγιο.
  - System.exit(): στατική μέθοδος της κλάσης System

Περιβάλλουσες κλάσεις

- Οι wrapper classes Integer, Double, Boolean και Character έχουν πολλές στατικές μεθόδους και στατικά πεδία που μας βοηθάνε να χειριζόμαστε τους βασικούς τύπους.
  - Integer.parseInt(String): Μετατρέπει ένα String σε int.
    - Αντίστοιχα: Double.parseDouble(String), Boolean.parseBoolean(String)
  - Integer.MAX_VALUE, Integer.MIN_VALUE: Μέγιστη και ελάχιστη τιμή ενός ακεραίου
    - Αντίστοιχα: Double.MAX_VALUE, Double.MIN_VALUE
  - Character.isDigit(char): επιστρέφει true αν ο χαρακτήρας είναι ένα ψηφίο
    - Παρόμοια: Character.isLetter(char), Character.isLetterOrDigit(), Character.isWhiteSpace(char)
- Οι κλάσεις αυτές έχουν και μη στατικές μεθόδους.

Η κλάση Math

- Μία κλάση με πολλές στατικές μεθόδους και στατικά πεδία για μαθηματικούς υπολογισμούς
- Παραδείγματα
  - min: επιστρέφει το ελάχιστο δύο αριθμών
  - max: επιστρέφει το μέγιστο δύο αριθμών
  - abs: επιστρέφει την απόλυτη τιμή
  - pow(x,y): υψώνει το x στην y δυναμη
  - floor/ceil: επιστρέφει τον μεγαλύτερο/μικρότερο ακέραιο που είναι μικρότερος/μεγαλυτερος από το όρισμα
  - sqrt: επιστρέφει την τετραγωνική ρίζα ενός αριθμού
  - PI: ο αριθμός π
  - E: Η βάση των φυσικών λογαρίθμων

Συμπερασματικά

- Στατικές μεθόδους και πεδία συνήθως ορίζουμε όταν θέλουμε μια βοηθητική συλλογή από σταθερές και μεθόδους (παρόμοια με την κλάση Math της Java).
- Μια στατική μέθοδο που μπορείτε να ορίσετε για κάθε κλάση είναι η main, ώστε να τεστάρετε μια συγκεκριμένη κλάση.

ΕΣΩΤΕΡΙΚΕΣ ΚΛΑΣΕΙΣ

Εσωτερικές κλάσεις

- Μπορούμε να ορίσουμε μια κλάση μέσα στον ορισμό μιας άλλης κλάσης

```
class Shape
{
        private class Point
        {

                <Code for Point>
        }

        <Code for Shape>
}
```

Γιατί να το κάνουμε αυτό?

- Η κλάση Point μπορεί να είναι χρήσιμη μόνο για την Shape
- Μας επιτρέπει να ορίσουμε άλλη Point σε άλλο σημείο

- Η Point και η Shape έχουν η μία πρόσβαση στα ιδιωτικά πεδία και μεθόδους της άλλης

ENUM

Τι είναι το enum

- Ένα enum είναι μια ειδική "κλάση" που αντιπροσωπεύει μια ομάδα σταθερών (αμετάβλητες μεταβλητές, όπως οι final μεταβλητές).
- Για να δημιουργήσετε ένα enum, χρησιμοποιούμε την λέξη-κλειδί enum (αντί class ή interface) και διαχωρίστε τις σταθερές με κόμμα. Σημειώστε ότι πρέπει να είναι με κεφαλαία γράμματα:

```
enum Level {
  LOW,
  MEDIUM,
  HIGH
};
```

Πρόσβαση στο enum

- Γράφουμε το όνομα του enum και χρησιμοποιούμε την τελεία.

```
enum Level {
  LOW,
  MEDIUM,
  HIGH
};

public class MyClass {
  public static void main(String[] args) {
    Level myVar = Level.MEDIUM;
    System.out.println(myVar);
  }
}
```

Enum μέσα σε κλάση

```
public class MyClass {
  enum Level {
    LOW,
    MEDIUM,
    HIGH
  };

  public static void main(String[] args) {
    Level myVar = Level.MEDIUM;
    System.out.println(myVar);
  }
}
```

Enum σε χρήση με εντολή Switch

```
enum Level {
  LOW,
  MEDIUM,
  HIGH
};

public class MyClass {
  public static void main(String[] args) {
    Level myVar = Level.MEDIUM;

    switch(myVar) {
      case LOW:
        System.out.println("Low level");
        break;
      case MEDIUM:
         System.out.println("Medium level");
        break;
      case HIGH:
        System.out.println("High level");
        break;
    }
  }
}
```

Πως διατρέχουμε ένα enum

```
for (Level myVar : Level.values()) {
  System.out.println(myVar);
}
```

Διαφορές ανάμεσα σε Enums και κλάσεις

- Ένα enum μπορεί, ακριβώς όπως και οι κλάσεις, να έχει πεδία και μεθόδους. Η διαφορά είναι ότι στο enum οι σταθερές είναι public, static και final (δεν αλλάξουν – δεν μπορούν να γίνουν overridden).
- Σε ένα enum δεν μπορούμε να δημιουργήσουμε αντικείμενα και δεν μπορεί να κληρονομήσει από άλλη κλάση.
- Μπορεί όμως ένα enum υλοποιήσει ένα interface.

```
public interface Operator {
    int apply (int a, int b);
}

public enum SimpleOperators implements Operator {
    PLUS {     int apply(int a, int b) { return a + b; }     },
    MINUS {    int apply(int a, int b) { return a - b; }     },
    PRODUCT {  int apply(int a, int b) { return a * b; }   },
    DIVISION {  int apply(int a, int b) { return a/b; }    };
}

public enum ComplexOperators implements Operator {
    POWER {  int apply(int a, int b) { return a^b; } },
    SQRTA {  int apply(int a, int b) { return Math.sqrt(a*b);}  };
}
……..
public static void main(string args[]){    ArrayList<Operator> operators = new ArrayList<Operator>();
SimpleOperators  s1=SimpleOperators.PLUS;
ComplexOperatots c1=ComplexOperators.POWER;
operators.add(s1);
operators.add(c1);

For(i=0;i<operators.length();i++)
            System.out.println(“the result is “+operators[i](10,i));
```

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

Καθορίζει τι κάνει το παράθυρο όταν πατάμε το κουμπί για κλείσιμο

Κάνει το παράθυρο ορατό

JFrame

- Επιλογές για το setDefaultCloseOperation:
  - EXIT_ON_CLOSE: Καλεί την Sytem.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.

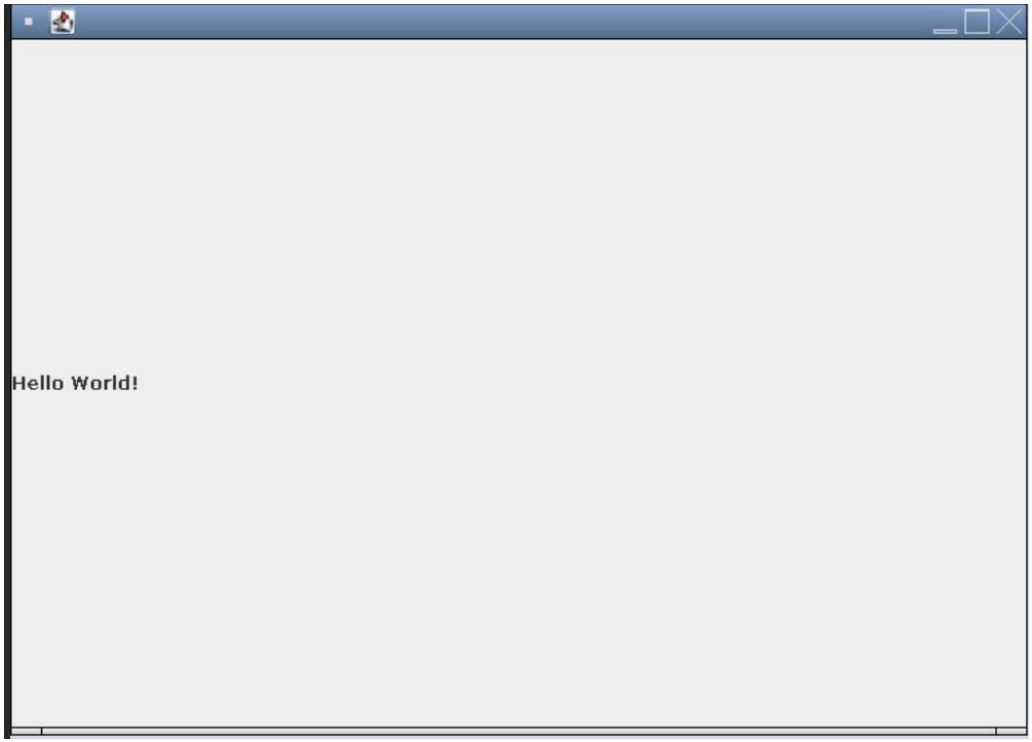
Ετικέτες

- 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


Hello World!

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 actionPerfomed 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.


Click to end program.

```
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);
    }
}
```

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

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 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.

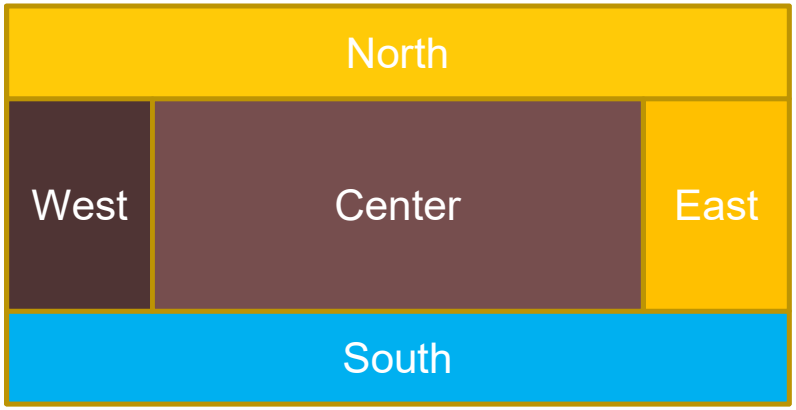
BorderLayout

- 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

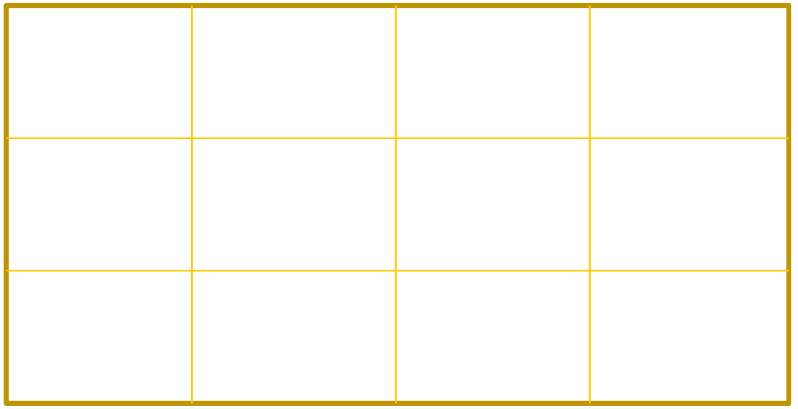
- Στην περίπτωση αυτή ορίζουμε ένα πλέγμα με n γραμμές και m στήλες και αυτό γεμίζει από τα αριστερά προς τα δεξιά και από πάνω προς τα κάτω
- Καλούμε την εντολή

```
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;

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);
    }
```

Η κλάση υλοποιεί τον ακροατή και την actionPerformed μεθοδο

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);
        }
    }

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

}
```

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

Το αποτέλεσμα του κάθε διαφορετικού κουμπιού.

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

Δημιουργία του παραθύρου


Multi-Color
Pick A Color
Blue
Red
Exit


Multi-Color
Pick A Color
Blue
Red
Exit

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