

# ΤΕΧΝΙΚΕΣ ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΟΥΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

---

Στατικές μέθοδοι και μεταβλητές  
Εσωτερικές κλάσεις

# 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**

# ΣΥΝΤΑΚΤΙΚΟ

- 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
```

επιστρέφει 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**.

# ΣΥΝΤΑΚΤΙΚΟ

- 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/μεγαλύτερος 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

ΕΠΙΣΚΟΠΗΣΗ

---

# Θέματα που καλύψαμε

- Γενικές έννοιες αντικειμενοστραφούς προγραμματισμού
- Βασικά στοιχεία Java
- Κλάσεις και αντικείμενα
  - Πεδία, μέθοδοι, δημιουργοί, αναφορές
- Σύνθεση και συνάθροιση αντικειμένων
  - Πώς να φτιάχνουμε μεγαλύτερες κλάσεις με μικρότερα αντικείμενα - σχεδίαση
- Κληρονομικότητα, Πολυμορφισμός
- Συλλογές δεδομένων
- Εξαιρέσεις, I/O με αρχεία
- Γραφικά περιβάλλοντα

# Αντικειμενοστραφής Προγραμματισμός

- Αν και το μάθημα έγινε σε Java, οι **βασικές αρχές** είναι οι ίδιες και για άλλες αντικειμενοστραφείς γλώσσες, και μπορείτε να μάθετε πολύ γρήγορα μια οποιαδήποτε **άλλη γλώσσα προγραμματισμού**
  - Μπορείτε να μάθετε **C#** σε μια βδομάδα
  - Η **C++** είναι λίγο πιο μπερδεμένη γιατί πρέπει να κάνετε μόνοι σας τη διαχείριση μνήμης αλλά με τις βασικές αρχές που ξέρετε μπορείτε να την μάθετε γρήγορα.

# Εξετάσεις

- Οι εξετάσεις θα είναι με ανοιχτά βιβλία και σημειώσεις
- Οι ερωτήσεις θα είναι στο πνεύμα των εργαστηρίων και των ασκήσεων
  - Κατά κύριο λόγο θα είναι προγραμματιστικές, αλλά μπορεί να ζητηθεί να ονομάσετε ένα μηχανισμό, ή να εξηγήσετε γιατί συμβαίνει κάτι
- Καλή επιτυχία!