ΤΕΧΝΙΚΕΣ ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΟΥΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Μαθήματα από τα εργαστήρια

ΕΡΓΑΣΤΗΡΙΟ 1

Μαθήματα από το πρώτο εργαστήριο

- Δημιουργία αντικειμένου Scanner
  - `Scanner input = new Scanner(System.in);`
  - Το αντικείμενο `input` είναι η σύνδεση του προγράμματος μας με το πληκτρολόγιο.
    - Έχουμε ένα πληκτρολόγιο θα δημιουργήσουμε ένα αντικείμενο Scanner το οποίο θα χρησιμοποιήσουμε για να διαβάσουμε οτιδήποτε πληκτρολογηθεί.
    - Δεν έχει νόημα να κάνουμε ένα αντικείμενο για κάθε μεταβλητή που διαβάζουμε.
  - Μέθοδοι της Scanner:
    - `next()`: επιστρέφει το επόμενο String από την είσοδο (όλοι οι χαρακτήρες από το σημείο που σταμάτησε την προηγούμενη φορά μέχρι να βρει white space: κενο, tab, αλλαγή γραμμής)
    - `nextInt()`: διαβάζει το επόμενο String και το μετατρέπει σε int και επιστρέφει ένα int αριθμό.
    - `nextDouble()`: διαβάζει το επόμενο String και το μετατρέπει σε double και επιστρέφει τον double αριθμό.
    - `nextLine()`: Διαβάζει ότι υπάρχει μέχρι να βρει newline και το επιστρέφει ως String.

Μαθήματα από το πρώτο εργαστήριο

- Διάβασμα από την είσοδο:
  - Θέλουμε να διαβάσουμε ένα πραγματικό αριθμό ακολουθούμενο από ένα string.

ΣΩΣΤΟ!

```
Scanner in = new Scanner(System.in);
double d = in.nextDouble();
String s = in.next();
```

ΛΑΘΟΣ!

```
Scanner in = new Scanner(System.in);
double d = in.nextDouble();
String s = in.nextLine();
```

Το nextLine() δεν μας κάνει γιατί διαβάζει ότι ακολουθεί τον αριθμό μέχρι να βρει “\n”
Αν πατήσουμε το enter μετά από τον ακέραιο, στην είσοδο μένει το κενό String και το “\n”
Το nextLine() επιστρέφει λοιπόν το κενό String.

Μαθήματα από το πρώτο εργαστήριο

- Ορισμός και χρήση μεταβλητών:
  - Μια μεταβλητή ορίζεται μόνο μία φορά μέσα σε ένα λογικό μπλοκ του κώδικα μας
  - Όταν θέλουμε να την χρησιμοποιήσουμε δεν χρειάζεται και δεν μπορούμε να την ορίσουμε ξανά.

```
import java.util.Scanner;

class VariableTest
{
        public static void main(String[] 
        {
                Scanner in = new Scanner(System.in);
                String s = in.nextLine();
                while (!s.equals("exit")){
                        System.out.println("You entered:"+s);
                        s = in.nextLine();
                }
        }
}
```

Η γραμμή αυτή κάνει δύο πράγματα:
1. Ορίζει την μεταβλητή s: String s
2. Εκχωρεί στην s το αποτέλεσμα της in.nextLine()

Ορισμός μεταβλητής:
<τύπος> <όνομα>
String s;

Εφόσον έχουμε ήδη ορίσει την μεταβλητή String s, δεν μπορούμε να την ορίσουμε ξανά. Εδώ απλά την χρησιμοποιούμε για να εκχωρήσουμε νέα τιμή

Μαθήματα από το πρώτο εργαστήριο

- Στοίχιση κώδικα:
  - Κάθε φορά που ανοίγετε ένα καινούριο μπλοκ οι εντολές θα πρέπει να πηγαίνουν ένα tab πιο μέσα
    - Χρησιμοποιείτε τα tabs και όχι κενά.
  - Τα άγκιστρα που σηματοδοτούν την αρχή και το τέλος του μπλοκ είναι στοιχισμένα με τις προηγούμενες εντολές.

```
import java.util.Scanner;

class VariableTest
{
← 1 tab →public static void main(String[] args)
         {
←——2 tabs——→Scanner in = new Scanner(System.in);
             String s = in.nextLine();
             while (!s.equals("exit"))
             {
←————— 3 tabs —————→System.out.println("You entered:"+s);
                    s = in.nextLine();
             }
         }
}
```

ΕΡΓΑΣΤΗΡΙΟ 2

Μαθήματα από το lab

- Όταν ορίζουμε ένα πίνακα, για να μπορούμε να τον χρησιμοποιήσουμε θα πρέπει να του δώσουμε χώρο, δηλαδή να δεσμεύσουμε μνήμη για τις θέσεις που χρειαζόμαστε
  - Αυτό γίνεται με την new.
  - Δεσμεύουμε τόσες θέσεις μνήμης όσο το μέγεθος του πίνακα
  - Η κάθε θέση μνήμης αποθηκεύει μια μεταβλητή τύπου τον τύπο του πίνακα
    - `int[] array = new int[3]`, δεσμεύει τρεις θέσεις που κρατάνε ακεραίους
    - `Car[] array = new Car[n]`, δεσμεύει `n` θέσεις που κρατάνε αντικείμενα τύπου `Car`.
- Αν δεν καλέσουμε την new για τον πίνακα τότε η μεταβλητή μας είναι κενή (φανταστείτε το σαν ένα πίνακα με μηδέν θέσεις).

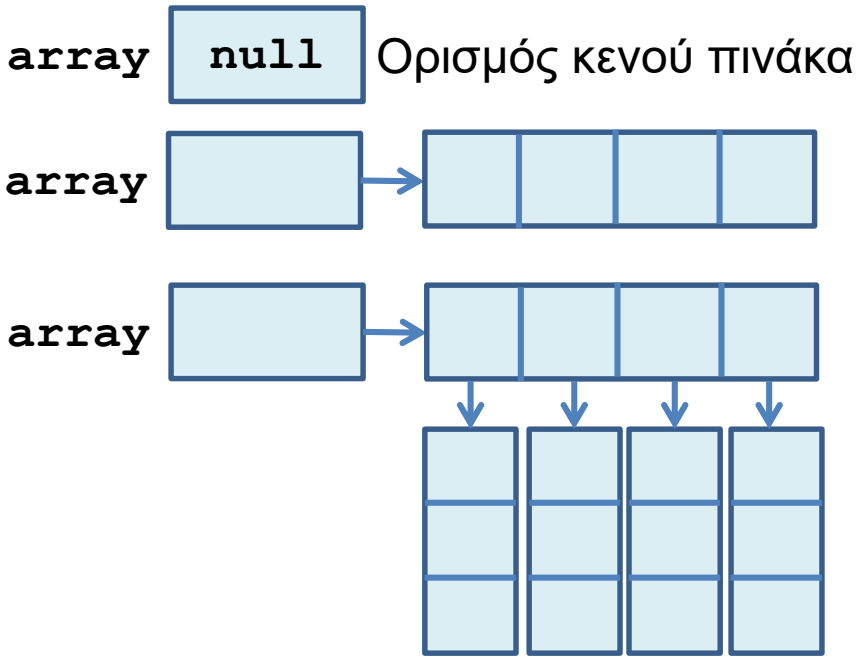
Μαθήματα από το lab

- Ένας δισδιάστατος πίνακας είναι ένας πίνακας από πίνακες.
- Για να του δώσουμε χώρο το κάνουμε ανά μία διάσταση.
  - Π.χ., για ένα $n \times m$ πίνακα με ακεραίους,
  - πρώτα θα δεσμεύσουμε n θέσεις μνήμης που η κάθε μία θα κρατάει πίνακες από ακεραίους,
  - μετά για κάθε θέση θα δεσμεύσουμε m θέσεις που θα κρατάνε ακεραίους.

```
int[][] array;

array = new int[n][];

for(int i = 0; i < n; i ++){
    array[i] = new int[m];
}
```

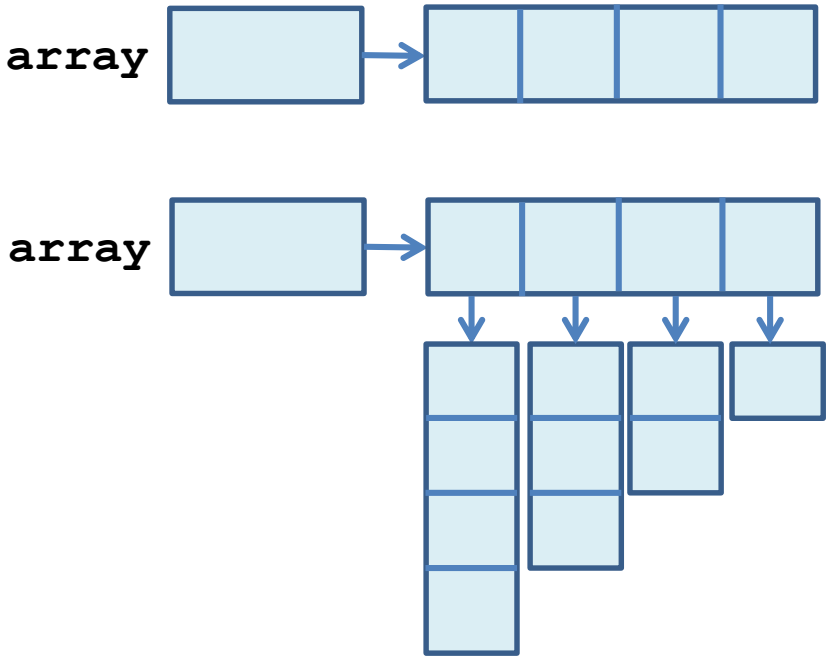


Μαθήματα από το lab

- Στην περίπτωση σας η κάθε γραμμή είχε μεταβλητή διάσταση

```
n = 4;
int[][] array = new int[n][];

for(int i = 0; i < n; i ++){
    array[i] = new int[n-i];
}
```



ΕΡΓΑΣΤΗΡΙΟ 3

Ένα *ιστόγραμμα* τιμών μετράει για ένα σύνολο από τιμές πόσες φορές εμφανίστηκε η κάθε τιμή. Για παράδειγμα αν έχω τις τιμές: 1,2,1,2,4,5,3,3,3,2,4 το ιστόγραμμα τους είναι 2,3,3,2,1, και είναι ο αριθμός εμφανίσεων των τιμών 1,2,3,4,5 αντίστοιχα (η τιμή 1 εμφανίζεται 2 φορές, η τιμή 2, 3 φορές, κοκ).

Στην άσκηση αυτή θα υλοποιήσετε μια κλάση **GradeHistogram** η οποία κρατάει ένα ιστόγραμμα για τους βαθμούς ενός μαθήματος. Η κλάση σας θα πρέπει να κρατάει το μέγιστο βαθμό για το μάθημα, και ένα πίνακα τον αριθμό εμφανίσεων του κάθε βαθμού. Αν ο μέγιστος βαθμός είναι maxGrade τότε οι πιθανοί βαθμοί θα είναι όλοι οι ακέραιοι στο διάστημα [1,maxGrade]. Η κλάση θα πρέπει να έχει και τις εξής μεθόδους:

1. Ένα **constructor,** ο οποίος θα παίρνει σαν όρισμα τον μέγιστο βαθμό και ένα πίνακα με βαθμούς και δημιουργεί το ιστόγραμμα των βαθμών.
2. Μια μέθοδο **toString**, η οποία θα επιστρέφει ένα String που αναπαριστά το ιστόγραμμα. Για το ιστόγραμμα στο παραπάνω παράδειγμα θα επιστρέφει το String: «1:2 2:3 3:3 4:2 5:1».
3. Την μέθοδο **equals**, η οποία θα συγκρίνει αν δύο ιστογράμματα είναι ίδια.
4. Μια μέθοδο **addHistogram** η οποία παίρνει σαν όρισμα ένα άλλο ιστόγραμμα (ένα αντικείμενο τύπου **GradeHistogram**) και, εφόσον έχουν τον ίδιο μέγιστο βαθμό, το προσθέτει στο υπάρχον ιστόγραμμα.

Σας δίνεται η κλάση **GradeHistogramTest,** για να τεστάρετε την κλάση σας. Όταν υλοποιήσετε τις μεθόδους που καλούνται στην main, βγάλετε τα σχόλια από τις αντίστοιχες εντολές για να τεστάρετε τις μεθόδους. Τεστάρετε τον κώδικα σας σταδιακά όπως φαίνεται στα σχόλια.

Μαθήματα από το lab

- Τι πληροφορία (δεδομένα) θέλουμε να κρατάει η κλάση μας?
  - Το μέγιστο βαθμό
  - Τις τιμές του ιστογράμματος

- Η πληροφορία (τα δεδομένα) που θέλουμε να κρατάει η κλάση θα είναι τα πεδία της κλάσης
  - Έναν ακέραιο maxGrade με το μέγιστο βαθμό που θα είναι και το μήκος του πίνακα
  - Ένα πίνακα ακεραίων histogram με τις συχνότητες για τον κάθε βαθμό

Κατασκευή ιστογράμματος

- Πως φτιάχνουμε ένα ιστόγραμμα από ένα πίνακα με βαθμούς?
  - Κάθε φορά που βλέπουμε τον βαθμό x θα πρέπει να αυξήσουμε την x-θέση του ιστογράμματος κατά ένα.

```
for (int i = 0; i < grades.length; i ++){
      int x = grades[i];
      histogram[x-1] ++;
}
```

Η μέθοδος toString

- Στην μέθοδο toString πρέπει να δημιουργήσουμε το String το οποίο θα αναπαριστά το ιστόγραμμα. Για να το κάνουμε αυτό θα πρέπει να διατρέξουμε τον πίνακα με τις τιμές και να φτιάξουμε το String αυξητικά.

Η μέθοδος toString ορίζεται πάντα έτσι

```
public String toString(){
    String output = "";
    for (int i = 0; i < maxGrade; i ++){
       output = output + (i+1) + ":" + histogram[i] + " ";
    }
    return output;
}
```

```
class GradeHistogram
{

  public GradeHistogram(int maxGrade, int[] grades)
  {
        int[] histogram = new int[maxGrade];
        for (int i = 0; i < grades.length; i ++){
              int x = grades[i];
              histogram[x-1] ++;
        }
  }

  public String toString(){
  {
        String output = "";
        for (int i = 0; i < maxGrade; i ++){
              output = output + (i+1) +
                    ":" + histogram[i] + " ";
        }
        return output;
  }
}
```

Σωστό ή λάθος?

Οι μεταβλητές maxGrade και histogram δεν είναι ορισμένες.

Για να μπορεί να τις βλέπει η μέθοδος print (ή οποιαδήποτε άλλη μέθοδος) θα πρέπει να είναι ορισμένες ως πεδία της κλάσης

ΛΑΘΟΣ!

```
class GradeHistogram
{
  private int maxGrade;
  private int[] histogram;

  public GradeHistogram(int maxGrade, int[] grades)
  {
        int[] histogram = new int[maxGrade];
        for (int i = 0; i < grades.length; i ++){
              int x = grades[i];
              histogram[x-1] ++;
        }
  }

  public String toString(){
  {
        String output = "";
        for (int i = 0; i < maxGrade; i ++){
              output = output + (i+1) +
                    ":" + histogram[i] + " ";
        }
        return output;
  }
}
```

Σωστό?

Ο constructor δεν αρχικοποιεί τα πεδία της κλάσης .

Οι μεταβλητές maxGrade και histogram που ορίζονται μέσα στον constructor είναι τοπικές μεταβλητές και δεν αλλάζουν την τιμή των πεδίων.

ΛΑΘΟΣ!

```
class GradeHistogram
{
  private int maxGrade;
  private int[] histogram;

  public GradeHistogram(int maxGrade, int[]
  {
        this.maxGrade = maxGrade;
        for (int i = 0; i < grades.length; i ++){
              int x = grades[i];
              histogram[x-1] ++;
        }
  }

  public String toString(){
  {
        String output = "";
        for (int i = 0; i < maxGrade; i ++){
              output = output + (i+1) +
                    ":" + histogram[i] + " ";
        }
        return output;
  }
}
```

Σωστό?

Η μεταβλητή maxGrade αρχικοποιείται σωστά.

Ο πίνακας histogram όμως όχι.

Τον έχουμε ορίσει σωστά αλλά δεν τον έχουμε δημιουργήσει (δεν του έχουμε δώσει χώρο)! Δεν έχουμε προσδιορίσει το μέγεθος του

ΛΑΘΟΣ!

```
class GradeHistogram
{
   private int maxGrade;
   private int[] histogram = new int[maxGrade];

   public GradeHistogram(int maxGrade, int[] gra
   {
         this.maxGrade = maxGrade;
         for (int i = 0; i < grades.length; i ++){
               int x = grades[i];
               histogram[x-1] ++;
         }
   }

   public String toString(){
   {
         String output = "";
         for (int i = 0; i < maxGrade; i ++){
               output = output + (i+1) +
                     ":" + histogram[i] + " ";
         }
         return output;
   }
}
```

Σωστό?

Θυμηθείτε ότι οι εντολές αυτές θα εκτελεστούν πριν από τις εντολές του constructor. Εκείνη τη στιγμή δεν ξέρουμε το μέγιστο βαθμό και άρα δημιουργούμε ένα πίνακα μηδενικού μεγέθους!

ΛΑΘΟΣ!

```
class GradeHistogram
{
   private int maxGrade;
   private int[] histogram;

   public GradeHistogram(int maxGrade, int[] grade
   {
         histogram = new int[maxGrade];
         for (int i = 0; i < grades.length; i ++){
                int x = grades[i];
                histogram[x-1] ++;
         }
   }

   public String toString(){
   {
         String output = "";
         for (int i = 0; i < maxGrade; i ++){
                output = output + (i+1) +
                       ":" + histogram[i] + " ";
         }
         return output;
   }
}
```

Σωστό?

Ο Constructor θα αρχικοποιήσει σωστά τον πίνακα histogram, αλλά δεν θα αλλάξει το πεδίο maxGrade μιας και χρησιμοποιεί την τοπική μεταβλητή - παράμετρο

Το maxGrade εδώ αναφέρεται στο πεδίο και έχει τιμή μηδέν.

ΛΑΘΟΣ!

```
class GradeHistogram
{
   private int maxGrade;
   private int[] histogram;

   public GradeHistogram(int maxGrade, int[] gr
   {
          this.maxGrade = maxGrade;
          histogram = new int[maxGrade];
          for (int i = 0; i < grades.length; i ++){
                 int x = grades[i];
                 histogram[x-1] ++;
          }
   }

   public String toString()
   {
          String output = "";
          for (int i = 0; i < maxGrade; i ++){
                 output = output + (i+1) +
                        ":" + histogram[i] + " ";
          }
          return output;
   }
}
```

Σωστό?

Πρώτα δηλώνουμε τα πεδία μέσα στην κλάση

Στον Constructor δίνουμε τιμή στο maxGrade και αφού πλέον ξέρουμε το μήκος του πίνακα τον δημιουργούμε και του δίνουμε χώρο για να κρατάει τις τιμές.

Τώρα μπορούμε και να κάνουμε και την αρχικοποίηση του πίνακα

ΣΩΣΤΟ!

Ορισμός και δημιουργία μεταβλητών

- Τι σημαίνει ορίζω μια μεταβλητή?
  - Οπουδήποτε έχουμε κώδικα της μορφής

    <τυπος> <όνομα μεταβλητής>

    ορίζουμε μια καινούρια μεταβλητή με αυτό το όνομα. Π.χ.,
    - `int maxGrade`
    - `int[] histogram`
    - `GradeHistogram hist`
- Τι σημαίνει δημιουργώ μια μεταβλητή/αντικείμενο
  - Δημιουργώ σημαίνει ότι δίνω χώρο στην μνήμη και αυτό γίνεται με την new. Χωρίς την κλήση της new το αντικείμενο δεν υπάρχει. Εξαίρεση, οι βασικοί τύποι (int, double, boolean).
    - `histogram = new int[maxGrade];`
    - `hist = new GradeHistogram(maxGrade, grades);`

Εμβέλεια μεταβλητών

- Η κάθε μεταβλητή έχει εμβέλεια μέσα στο block στο οποίο ορίζεται.
  - Τις μεταβλητές-πεδία της κλάσης μπορούν να τις χρησιμοποιήσουν όλες οι μέθοδοι της κλάσης
    - Οι μεταβλητές έχουν ζωή όσο υπάρχει το αντίστοιχο αντικείμενο της κλάσης
  - Οι μεταβλητές που ορίζονται μέσα σε μία μέθοδο μπορούν να χρησιμοποιηθούν μόνο μέσα στη μέθοδο.
    - Οι μεταβλητές χάνονται όταν βγούμε από τη μέθοδο.
  - Οι παράμετροι μιας μεθόδου είναι σαν τοπικές μεταβλητές της μεθόδου.

Παράδειγμα

```
class GradeHistogram
{
   private int maxGrade;
   private int[] histogram;

   public GradeHistogram(int maxGrade, int[] grades)
   {
         this.maxGrade = maxGrade;
         histogram = new int[maxGrade];
         for (int i = 0; i < grades.length; i ++){
               int x = grades[i];
               histogram[x-1] ++;
         }
   }
}
```

Ορισμός μεταβλητής πίνακα

Δημιουργία πίνακα

Οι κόκκινες μεταβλητές υπάρχουν μόνο μέσα στο μπλοκ της μεθόδου

Οι μπλε μεταβλητές είναι πεδία

Παράδειγμα (λάθος)

```
class GradeHistogram
{
   private int maxGrade;
   private int[] histogram;

   public GradeHistogram(int maxGrade, int[] grades)
   {
         this.maxGrade = maxGrade;
         int[] histogram = new int[maxGrade];
         for (int i = 0; i < grades.length; i ++){
                int x = grades[i];
                histogram[x-1] ++;
         }
   }
}
```

Το πεδίο-πίνακας **histogram** έχει οριστεί αλλά δεν έχει δημιουργηθεί

Ορισμός τοπικής μεταβλητής και δημιουργία της

Οι κόκκινες μεταβλητές υπάρχουν μόνο μέσα στο μπλοκ της μεθόδου

Οι μπλε μεταβλητές είναι πεδία

Η τοπική μεταβλητή **histogram** χάνεται μόλις βγούμε από τον constructor

Η μέθοδος equals

Η μέθοδος equals ορίζεται πάντα έτσι

```
public boolean equals(GradeHistogram other)
{
      if (this.maxGrade != other.maxGrade){
            return false;
      }
      for (int i = 0; i < maxGrade; i ++){
            if (this.histogram[i] != other.histogram[i]){
                  return false;
            }
      }
      return true;
 }
```

Είναι πιο εύκολο να ελέγξουμε για την περίπτωση της ανισότητας παρά της ισότητας. Μόλις μία από τις συνθήκες δεν ικανοποιείται επιστρέφουμε false. Αν φτάσουμε μέχρι τέλους ικανοποιούνται όλες και άρα επιστρέφουμε true

Δεν κάνουμε έλεγχο ισότητας χρησιμοποιώντας την toString!
Το String που επιστρέφουμε μπορεί να μην ικανοποιεί τον έλεγχο ισότητας

Η μέθοδος addHistogram

Η μέθοδος δεν επιστρέφει κάτι μιας και το αποτέλεσμα της πρόσθεσης θα αποθηκευτεί στο αντικείμενο

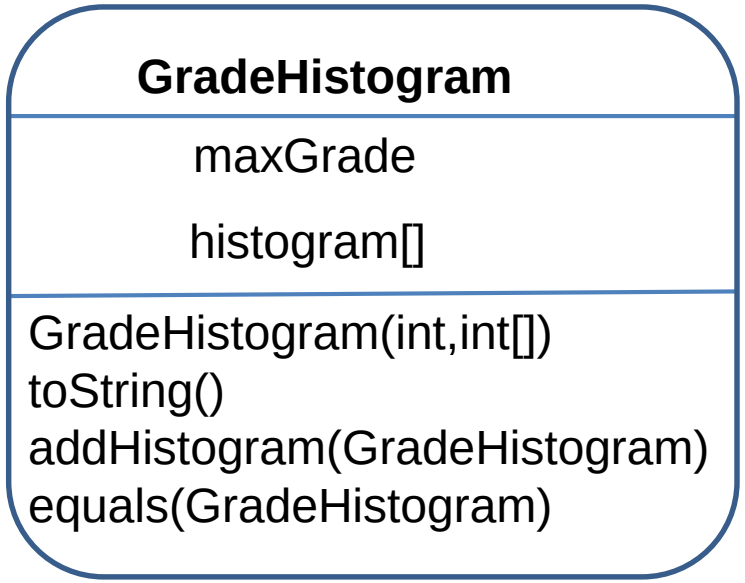
Η μέθοδος παίρνει σαν όρισμα ένα αντικείμενο GradeHistogram το οποίο θα προσθέσει

```
public void addHistogram(GradeHistogram other)
{
      if (this.maxGrade != other.maxGrade){
            return;
      }
      for (int i=0; i < maxGrade; i++){
        this.histogram[i] += other.histogam[i];
      }
}
```

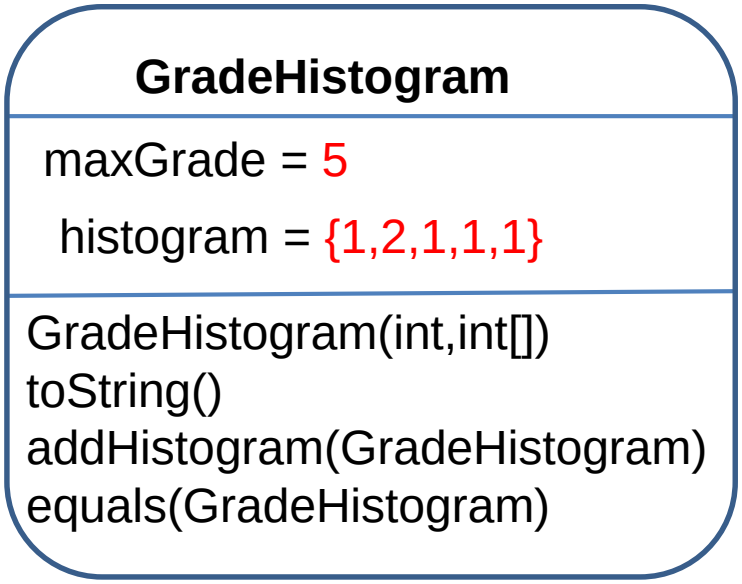
Έχουμε πρόσβαση στα πεδία του other γιατί είναι της ίδιας κλάσης με το αντικείμενο που καλεί την addHistogram

Κλάσεις και αντικείμενα

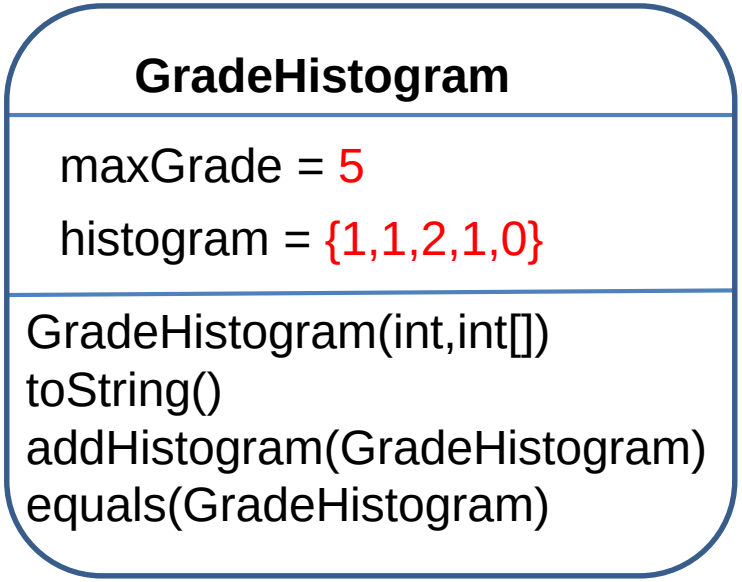
Ορισμός της κλάσης



hist2 =
new GradeHistogram(5,grades2)

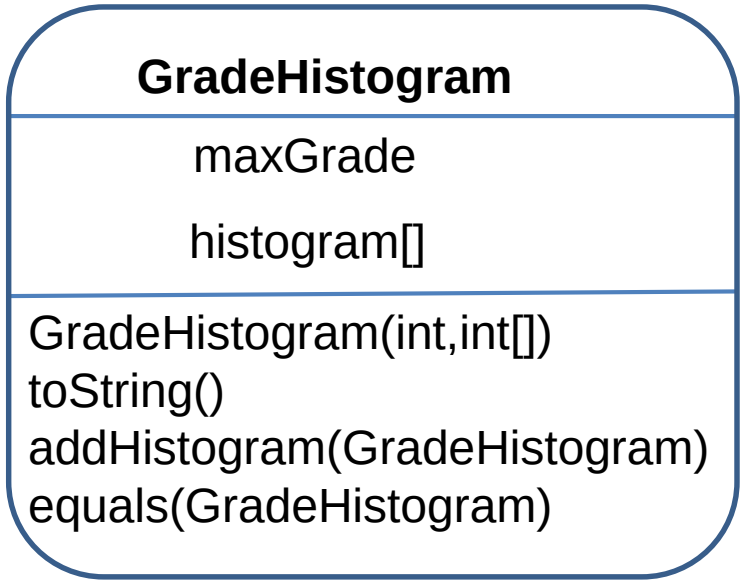


hist3 =
new GradeHistogram(5,grades3)



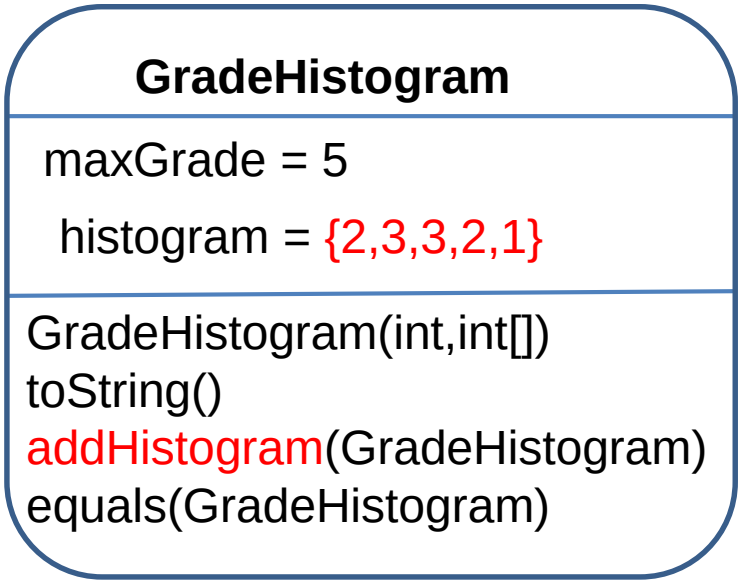
Κλάσεις και αντικείμενα

Ορισμός της κλάσης

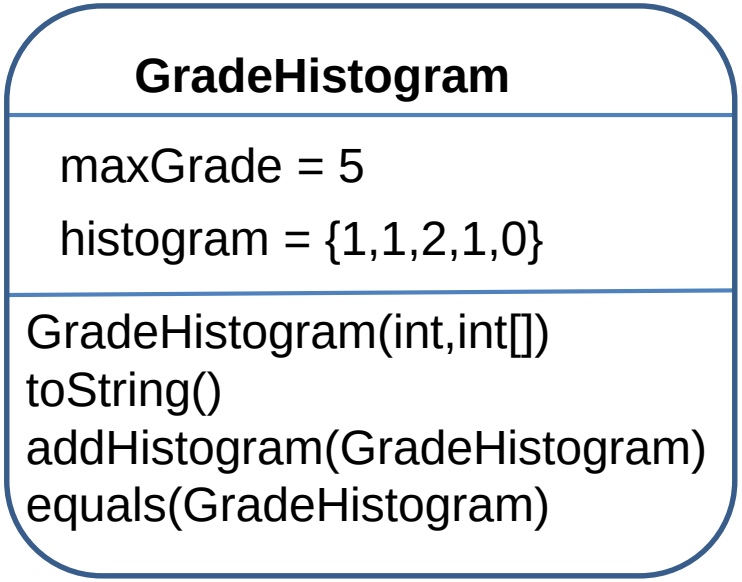


hist2.addHistogram(hist3);

hist2 =
new GradeHistogram(5,grades2)



hist3 =
new GradeHistogram(5,grades3)



```
class GradeHistogram
{
  private int maxGrade;
  private int[] histogram;
  private String output = "";

  public Geometric(int maxGrade, int[] grades)
  {
        this.maxGrade = maxGrade;
        histogram = new int[maxGrade];
        for (int i = 0; i < grades.length; i ++){
              x = grades[i];
              histogram[x-1] ++;
        }
  }

  public String toString(){
  {
        for (int i = 0; i < maxGrade; i ++){
              output = output + (i+1) +
                    ":" + histogram[i] + " ";
        }
        return output;
  }
}
```

Σωστό?

Η μεταβλητή output πλέον είναι πεδίο. Οι αλλαγές της τιμής της παραμένουν στο αντικείμενο

Τι γίνεται αν κάνουμε πολλαπλές κλήσεις της μεθόδου toString?

ΛΑΘΟΣ!

ΕΡΓΑΣΤΗΡΙΟ 4

Μαθήματα από το Εργαστήριο 4

Υπερφορτώστε την **deposit** ώστε να παίρνει όρισμα μια επιταγή και να προσθέτει το ποσό της επιταγής στον λογαριασμό, εφόσον η επιταγή έχει παραλήπτη τον κάτοχο του λογαριασμού. Επιστρέφει boolean τιμή αν έγινε σωστά η κατάθεση.

- Στην άσκηση μας έχουμε φτιάξει μια κλάση Check η οποία κρατάει πληροφορίες για μία επιταγή
- Όταν σας ζητάει μια άσκηση να περάσετε σαν όρισμα μια επιταγή αυτό σημαίνει ότι θα περάσετε σαν όρισμα ένα αντικείμενο Check.
  - Όχι String, όχι το όνομα και το ποσό αλλά το αντικείμενο Check.
  - Το αντικείμενο αυτό περιέχει όλη την πληροφορία που χρειαζόμαστε
- Δεν μπορούμε να προσπελάσουμε αυτή την πληροφορία άμεσα όμως γιατί η κλάση Check είναι διαφορετική από την κλάση Account που έχει την μέθοδο.
  - Γι αυτό ορίζουμε της μεθόδους προσπέλασης στην κλάση Check ώστε να διαβάσουμε τα πεδία που χρειαζόμαστε.
  - Μια μέθοδος πρόσβασης έχει το όνομα get<Όνομα Πεδίου> όπου το πεδίο είναι αυτό που θέλουμε να διαβάσουμε.
    - getName()
    - getAmmount()

Η μέθοδος deposit

Η παράμετρος στην μέθοδο είναι ένα αντικείμενο Check.

Μην χρησιμοποιείτε συνέχεια το other για το όνομα της παραμέτρου. Η ιδέα του other είναι να το χρησιμοποιούμε όταν η παράμετρος είναι αντικείμενο της ίδιας κλάσης.

```
public boolean deposit(Check depositCheck)
{
        if (depositCheck.getName().equals(this.name)){
                this.amount += depositCheck.getAmount();
                return true;
        }
        return false;
}
```

Έλεγχος ισότητας με equals

Το getName() και getAmmount() είναι μέθοδοι της κλάσης Check, όχι πεδία. Άρα χρειαζόμαστε να τις καλέσουμε με τα ορίσματα τους. Στην περίπτωση αυτή δεν έχουν ορίσματα άρα απλά βάζουμε κενές παρενθέσεις.