ΤΕΧΝΙΚΕΣ ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΟΥΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Αντικείμενα με πίνακες.

Constructors.

Υλοποίηση Στοίβας

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

Ορισμός μεταβλητής πίνακα

Δημιουργία πίνακα

```
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] ++;
         }
   }
}
```

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

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

Οι κόκκινες μεταβλητές υπάρχουν μόνο μέσα στο μπλοκ της μεθόδου

Οι μπλε μεταβλητές είναι πεδία

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

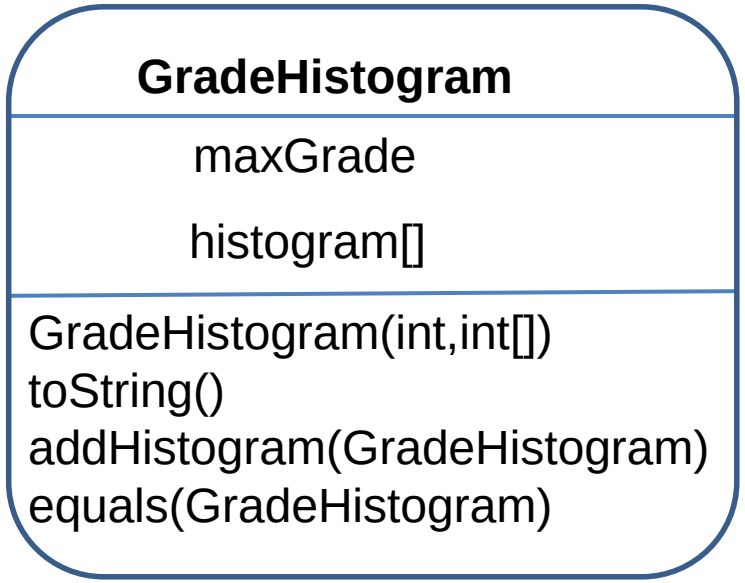
Η μέθοδος παίρνει σαν όρισμα ένα αντικείμενο GradeHistogram το οποίο θα προσθέσει

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

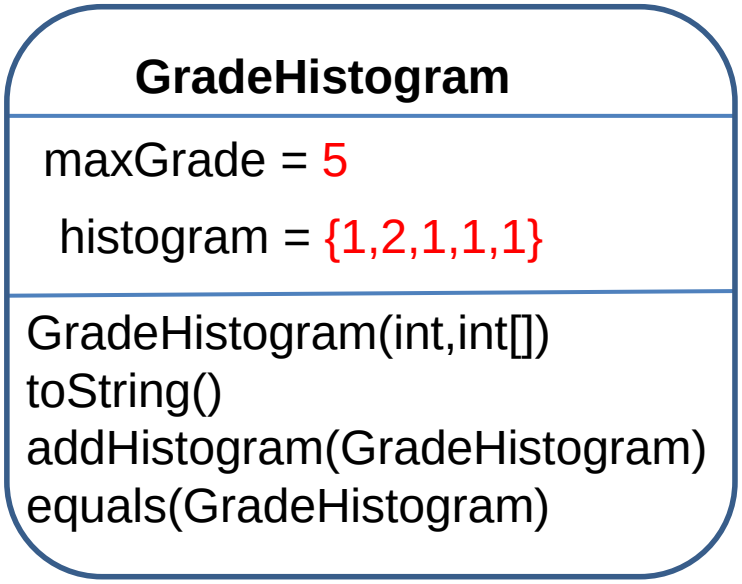
Έχουμε πρόσβαση στα πεδία του other γιατί είναι της ίδιας κλάσης με το αντικείμενο που καλεί την addHistogram

Κλάσεις και αντικείμενα

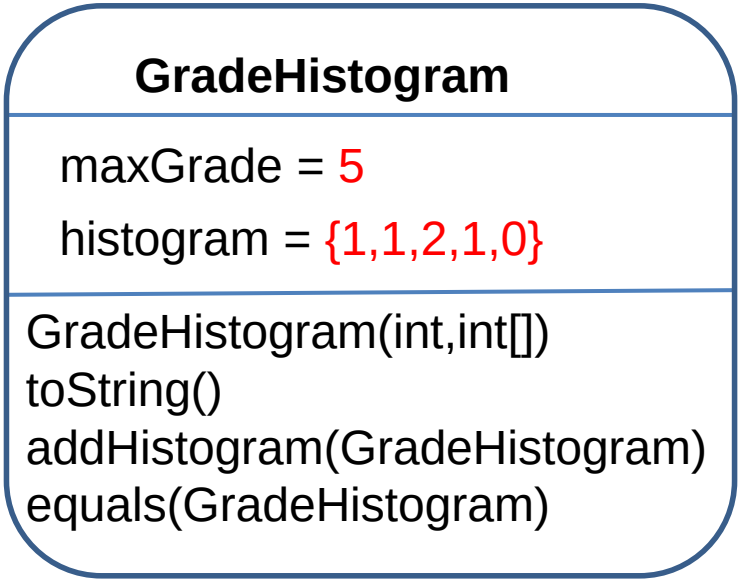
Ορισμός της κλάσης



hist2 =
new GradeHistogram(5,grades2)

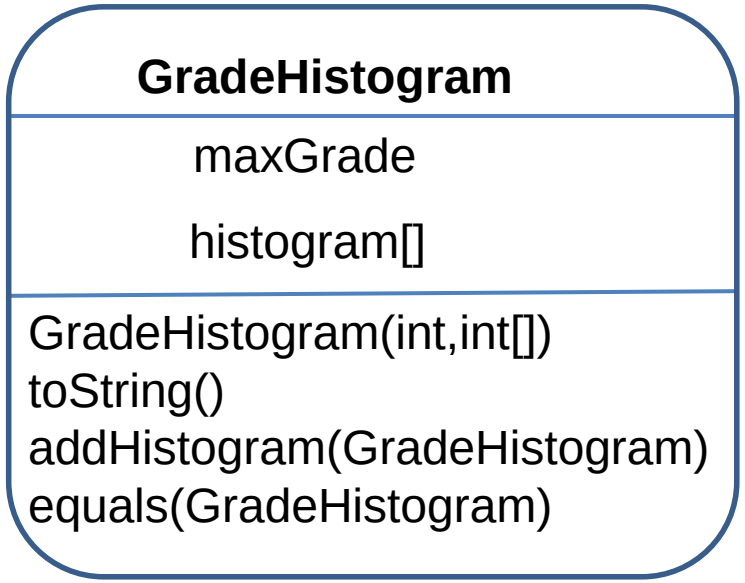


hist3 =
new GradeHistogram(5,grades3)



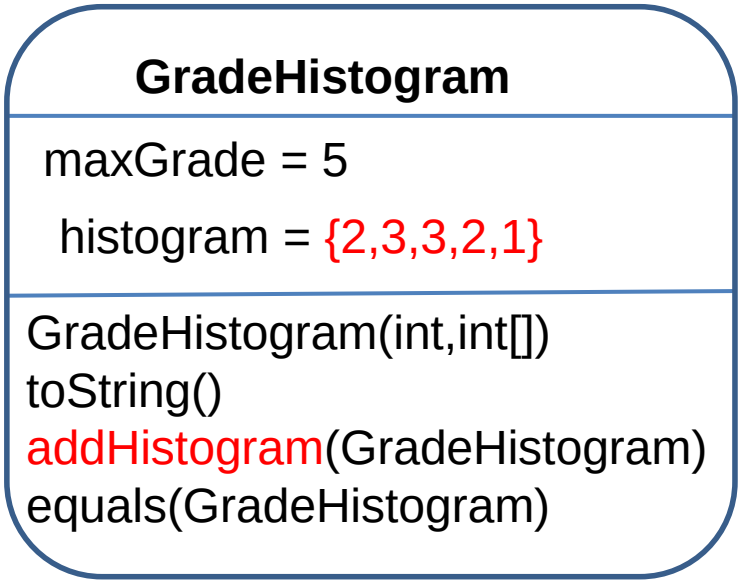
Κλάσεις και αντικείμενα

Ορισμός της κλάσης

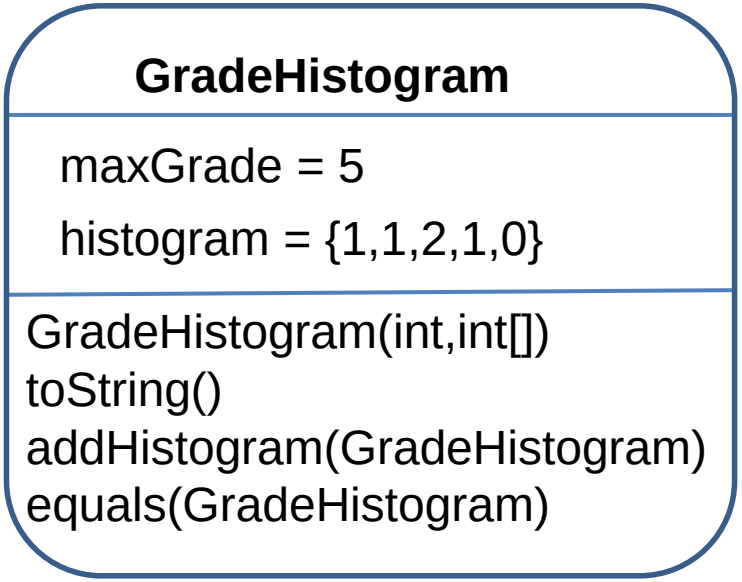


hist2.addHistogram(hist3);

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?

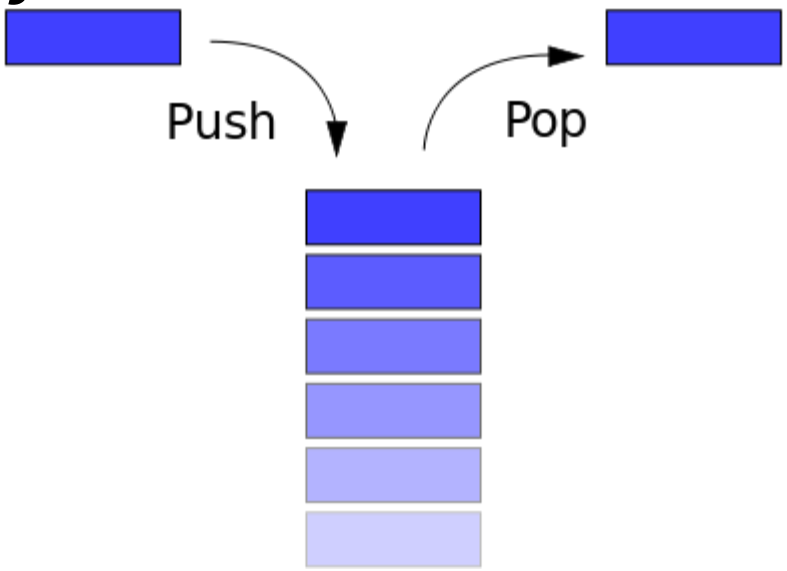
Η μεταβλητή output πλέον είναι πεδίο. Οι αλλαγές της τιμής της παραμένουν στο αντικείμενο

Τι γίνεται αν κάνουμε πολλαπλές κλήσεις της μεθόδου toString?

ΛΑΘΟΣ!

Παράδειγμα ADT: Στοίβα (Stack)

- Η Στοίβα είναι μια συλλογή δεδομένων η οποία επιτρέπει τις εξής λειτουργίες:
  - push(element): προσθέτει ένα νέο στοιχείο στην κορυφή της στοίβας
  - pop(): αφαιρεί και επιστρέφει το στοιχείο το οποίο βρίσκεται στην κορυφή της στοίβας.
  - isEmpty(): ελέγχει αν η στοίβα είναι άδεια και επιστρέφει true ή false
- Η Στοίβα υλοποιεί την πολιτική Last-In-First-Out (LIFO) στη σειρά που μας δίνει τα στοιχεία
  - Χρήσιμο σε διάφορες εφαρμογές, π.χ., για τη δέσμευση μνήμης στην κλήση συναρτήσεων



Υλοποίηση

- Θα υλοποιήσουμε μια Στοίβα ακεραίων χρησιμοποιώντας ένα πίνακα (Στοιβα συγκεκριμένης χωρητικότητας)
  - Τι πεδία πρέπει να ορίσουμε?
  - Τι μεθόδους?

```
class Stack
{
        private int capacity;
        private int size = 0;
        private int[] elements;

        public Stack(int capacity){
                this.capacity = capacity;
                elements = new int[capacity];
        }

        public void push(int element){
                if (size == capacity){
                        System.out.println("Cannot enter any more elements");
                        return;
                }
                elements[size] = element;
                size ++;
        }

        public int pop(){
                if (size == 0){
                        System.out.println("No elements to pop");
                        return -1;
                }
                size -- ;
                return elements[size];
        }

        public boolean isEmpty(){
                return (size == 0);
        }
}
```

Εφαρμογές

- Υπολόγισε την δυαδική μορφή ενός ακεραίου.

```
class Binary
{
	public static void main(String[] args)
	{
		Stack myStack = new Stack(100);
		int number = 1973;

		while (number > 0){
			myStack.push(number%2);
			number = number/2;
		}

		while (!myStack.isEmpty()){
			System.out.print(myStack.pop());
		}
	}
}
```

Επεκτάσεις

- Πως θα ορίσουμε την μέθοδο equals?
- Πως θα ορίσουμε τη μέθοδο toString?

```
public String toString(){
        String returnString = "":
        for (int i = 0; i < size; i ++){
                returnString = returnString + elements[i] + " ";
        }
        return returnString;
}

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