ΤΕΧΝΙΚΕΣ ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΟΥΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Κλάσεις και Αντικείμενα

Στην άσκηση αυτή θα υλοποιήσετε μια κλάση **RandomVector** η οποία διαχειρίζεται ένα τυχαίο διάνυσμα ακεραίων το οποίο μπορεί να έχει οποιοδήποτε μέγεθος. Η κλάση σας θα πρέπει να υποστηρίζει τις εξής λειτουργίες:

1. Ο **constructor** της κλάσης θα παίρνει σαν όρισμα την διάσταση του διανύσματος (τον αριθμό των συνιστωσών), και θα δίνει τυχαίες τιμές σε κάθε διάσταση μεταξύ 1 και 10.

2. Μια μέθοδο **add** η οποία παίρνει σαν όρισμα ένα άλλο διάνυσμα (ένα αντικείμενο τύπου Vector) και, εφόσον είναι δυνατόν, το προσθέτει στο παρόν διάνυσμα (προσθέτει τις τιμές ανά συνιστώσα) και αποθηκεύει το αποτέλεσμα στο παρόν διάνυσμα.
3. Μια μέθοδο **print**, η οποία θα τυπώνει τις τιμές του διανύσματος. Π.χ., για ένα τρισδιάστατο διάνυσμα με τιμές 2, 5, 4 θα τυπώνει «(2 5 4)».
4. Δημιουργήστε και ένα **constructor** ο οποίος δεν παίρνει ορίσματα και by default δημιουργεί τρισδιάστατα διανύσματα.

Για να τεστάρετε την κλάση σας δημιουργείστε μία άλλη κλάση **VectorTest** η οποία θα έχει τη main και θα δημιουργεί δύο διανύσματα, θα τυπώνει τα δύο διανύσματα, μετά θα τα προσθέτει και θα τυπώνει το αποτέλεσμα της πρόσθεσης. Έπειτα θα δημιουργήσετε δύο διανύσματα 3 διαστάσεων, όπου επίσης η κλάση σας θα τυπώνει τα δύο διανύσματα, μετά θα τα προσθέτει και θα τυπώνει το αποτέλεσμα της πρόσθεσης.

Μαθήματα από το lab

- Τι πληροφορία (δεδομένα) θέλουμε να κρατάει η κλάση μας?
  - Τη διάσταση του διανύσματος
  - Τις τιμές του διανύσματος.

- Η πληροφορία (τα δεδομένα) που θέλουμε να κρατάει η κλάση θα είναι τα πεδία της κλάσης
  - Ένα ακέραιο dimension για τη διάσταση του διανύσματος
  - Ένα πίνακα ακεραίων values μεγέθους dimension με τις τιμές του διανύσματος

```
import java.util.Random

class RandomVector
{
  public RandomVector(int dimension)
  {
          Random rndGen = new Random();
          int values[] = new int[dimension];
          for (int i=0; i < dimension; i++){
            values[i] = 1+rndGen.nextInt(10);
          }
  }

  public void print()
  {
          System.out.println("( ");
          for (int i = 0; i < dimension; i ++){
            System.out.println(values[i] + " ");
          }
          System.out.println(")");
  }
}

class VectorTest
{
  public static void main(String[] args){
          RandomVector v = new RandomVector(3);
          v.print();
  }
}
```

Σωστό ή λάθος?

Οι μεταβλητές dimension και values δεν είναι ορισμένες.

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

ΛΑΘΟΣ!

```
import java.util.Random

class RandomVector
{
  private int dimension=3;
  private int values[];

  public RandomVector(int dimension)
  {
        Random rndGen = new Random();
        int values[] = new int[dimension];
        for (int i=0; i < dimension; i++){
          values[i] = 1+rndGen.nextInt(10);
        }
  }

  public void print()
  {
        System.out.println("( ");
        for (int i = 0; i < dimension; i ++){
          System.out.println(values[i] + " ");
        }
        System.out.println(")");

  }
}

class VectorTest
{
  public static void main(String[] args){
        RandomVector v = new RandomVector(3);
        v.print();
  }
}
```

Σωστό?

Ο constructor δεν αρχικοποιεί τα πεδία της κλάσης .

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

ΛΑΘΟΣ!

```
import java.util.Random

class RandomVector
{
  private int dimension;
  private int values[];

  public RandomVector(int dimension)
  {
        Random rndGen = new Random();
        this.dimension = dimension;
        for (int i=0; i < dimension; i++){
          values[i] = 1+rndGen.nextInt(10);
        }
  }

  public void print()
  {
        System.out.println("( ");
        for (int i = 0; i < dimension; i ++){
          System.out.println(values[i] + " ");
        }
        System.out.println(")");
  }
}

class VectorTest
{
  public static void main(String[] args){
        RandomVector v = new RandomVector(3);
        v.print();
  }
}
```

Σωστό?

Η dimensions αρχικοποιείται σωστά.

Ο πίνακας values όμως όχι.

Τον έχουμε ορίσει σωστά αλλά δεν του έχουμε δώσει χώρο! Δεν έχουμε προσδιορίσει το μέγεθος του

ΛΑΘΟΣ!

```
import java.util.Random

class RandomVector
{
  private int dimension;
  private int values[] = new int[dimension];

  public RandomVector(int dimension)
  {
        Random rndGen = new Random();
        this.dimension = dimension;
        for (int i=0; i < dimension; i++){
          values[i] = 1+rndGen.nextInt(10);
        }
  }

  public void print()
  {
        System.out.println("( ");
        for (int i = 0; i < dimension; i ++){
          System.out.println(values[i] + " ");
        }
  }
}

class VectorTest
{
  public static void main(String[] args){
        RandomVector v = new RandomVector(3);
        v.print();
  }
}
```

Σωστό?

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

ΛΑΘΟΣ!

```
import java.util.Random

class RandomVector
{
  private int dimension;
  private int values[];

  public RandomVector(int dimension)
  {
          Random rndGen = new Random();
          values = new int[dimension];
          for (int i=0; i < dimension; i++){
            values[i] = 1+rndGen.nextInt(10);
          }
  }

  public void print()
  {
          System.out.println("( ");
          for (int i = 0; i < dimension; i ++){
            System.out.println(values[i] + " ");
          }
  }
}

class VectorTest
{
  public static void main(String[] args){
          RandomVector v = new RandomVector(3);
          v.print();
  }
}
```

Σωστό?

Ο Constructor θα αρχικοποιήσει σωστά τον πίνακα values, αλλά δεν θα αλλάξει το πεδίο dimension μιας και χρησιμοποιεί την τοπική μεταβλητή

Η dimension εδώ αναφέρεται στο πεδίο και έχει τιμή μηδέν.

ΛΑΘΟΣ!

```
import java.util.Random

class RandomVector
{
  private int dimension;
  private int values[];

  public RandomVector(int dimension)
  {
        Random rndGen = new Random();
        this.dimension = dimension;
        values = new int[dimension];
        for (int i=0; i < dimension; i++){
          values[i] = 1+rndGen.nextInt(10);
        }
  }

  public void print()
  {
        System.out.println("( ");
        for (int i = 0; i < dimension; i ++){
          System.out.println(values[i] + " ");
        }
  }
}

class VectorTest
{
  public static void main(String[] args){
        RandomVector v = new RandomVector(3);
        v.print();
  }
}
```

Σωστό?

Πρώτα δηλώνουμε τα πεδία μέσα στην κλάση

Μετά δίνουμε τιμή στη διάσταση και αφού πλέον ξέρουμε τη διάσταση δίνουμε χώρο στον πίνακα που θα κρατάει τις τιμές.

Τώρα μπορούμε και να κάνουμε και την αρχικοποίηση

ΣΩΣΤΟ!

```
import java.util.Random

class RandomVector
{
  private int dimension;
  private int values[];

  public RandomVector(int dimension)
  {
          Random rndGen = new Random();
          this.dimension = dimension;
          values = new int[dimension];
          for (int i=0; i < dimension; i++){
            values[i] = 1+rndGen.nextInt(10);
          }
  }

  public RandomVector()
  {
          Random rndGen = new Random();
          this.dimension = 3;
          values = new int[dimension];
          for (int i=0; i < dimension; i++){
            values[i] = 1+rndGen.nextInt(10);
          }
  }
}

class VectorTest
{
  public static void main(String[] args){
          RandomVector v = new RandomVector();
          v.print();
  }
}
```

Default constructor και η κλήση του

```
import java.util.Random

class RandomVector
{
  private int dimension = 3;
  private int values[] = new int[dimension];

  public RandomVector(int dimension)
  {
        Random rndGen = new Random();
        this.dimension = dimension;
        values = new int[dimension];
        for (int i=0; i < dimension; i++){
          values[i] = 1+rndGen.nextInt(10);
        }
  }

  public RandomVector()
  {
        Random rndGen = new Random();
        for (int i=0; i < dimension; i++){
          values[i] = 1+rndGen.nextInt(10);
        }
  }
}

class VectorTest
{
  public static void main(String[] args){
        RandomVector v = new RandomVector();
        v.print();
  }
}
```

Η αρχικοποίηση των πεδίων θα γίνει στις default τιμές

Ο constructor με όρισμα θα ξανα-ορίσει την διάσταση και θα δώσει νέο χώρο για τον πίνακα

Ο default constructor με όρισμα θα κρατήσει τις default τιμές

Όχι και τόσο καλή υλοποίηση

```
import java.util.Random

class RandomVector
{
  private int dimension;
  private int values[];

  public RandomVector(int dimension)
  {
        this.dimension = dimension;
        values = new int[dimension];
        fillValues();
  }

  public RandomVector()
  {
        this.dimension = 3;
        values = new int[dimension];
        fillValues();
  }

  private void fillValues()
  {
        Random rndGen = new Random();
        for (int i=0; i < dimension; i++){
          values[i] = 1+rndGen.nextInt(10);
        }
  }
}
```

Η διαδικασία του γεμίσματος του πίνακα επαναλαμβάνεται σε δύο μέρη. Μπορούμε λοιπόν να ορίσουμε μία βοηθητική μέθοδο που θα την υλοποιεί και θα την καλούμε στον constructor

Πλεονεκτήματα:

- Κάνει τον κώδικα πιο απλό και κατανοητό
- Το αντικείμενο Random ορίζεται μόνο εκεί που το χρειαζόμαστε.

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

```
public RandomVector(int dimension)
{
      this.dimension = dimension;
      int values[] = new int[dimension];
      for (int i=0; i < dimension; i++){
        values[i] = 0;
      }
}
```

Οι κόκκινες μεταβλητές υπάρχουν μόνο μέσα στο μπλοκ της μεθόδου

Παράμετρος

```
public RandomVector(int dimension)
{
      this.dimension = dimension;
      int values[] = new int[dimension];
      for (int i=0; i < dimension; i++){
        values[i] = 0;
      }
}
```

Οι παράμετροι είναι σαν τοπικές μεταβλητές

```
public RandomVector(όρισμα)
{
      int dimension = <τιμή ορίσματος>
      this.dimension = dimension;
      int values[] = new int[dimension];
      for (int i=0; i < dimension; i++){
        values[i] = 0;
      }
}
```

Η μέθοδος add

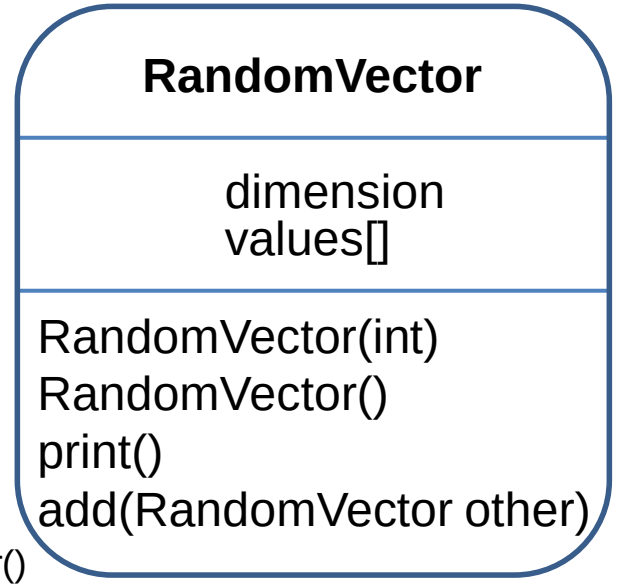
Η μέθοδος δεν επιστρέφει κάτι μιας και το αποτέλεσμα της πρόσθεσης θα αποθηκευτεί στο αντικείμενο

```
public void add(RandomVector other)
{
      if (this.dimension != other.dimension){
            return;
      }
      for (int i=0; i < dimension; i++){
         this.values[i] += other.values[i];
      }
}
```

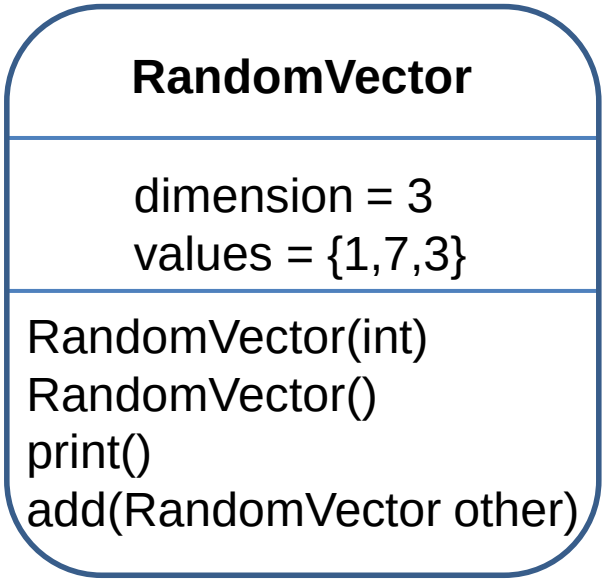
Έχουμε πρόσβαση στα πεδία του other γιατί είναι της ίδιας κλάσης με το αντικείμενο που καλεί την add

Κλάσεις και αντικείμενα

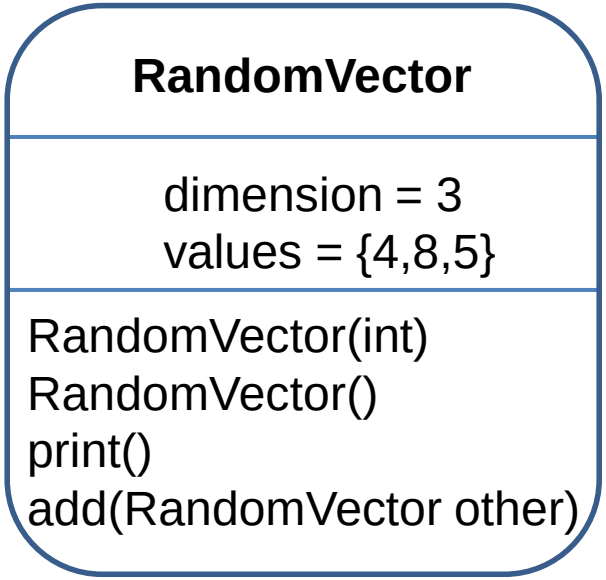
Ορισμός της κλάσης



vector1 = new RandomVector()

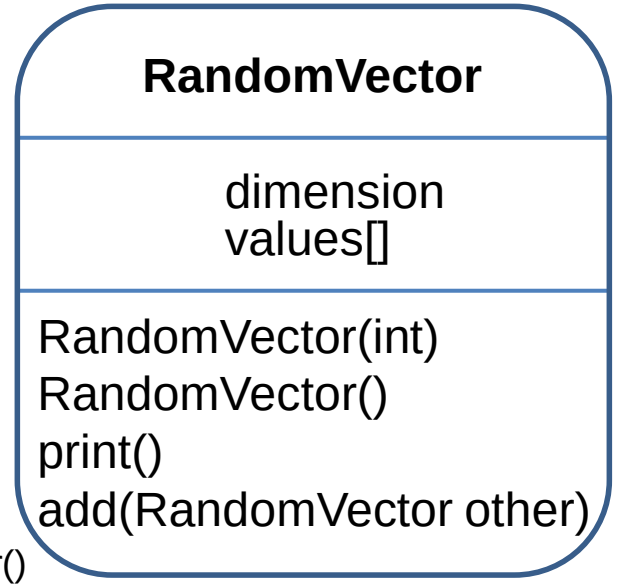


vector2 = new RandomVector()



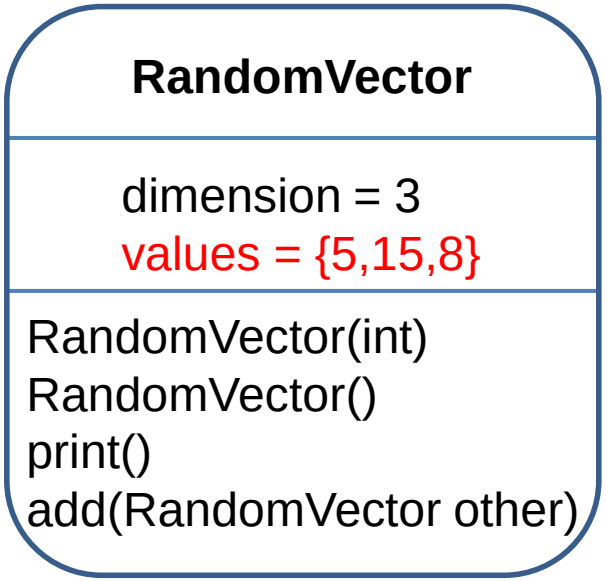
Κλάσεις και αντικείμενα

Ορισμός της κλάσης

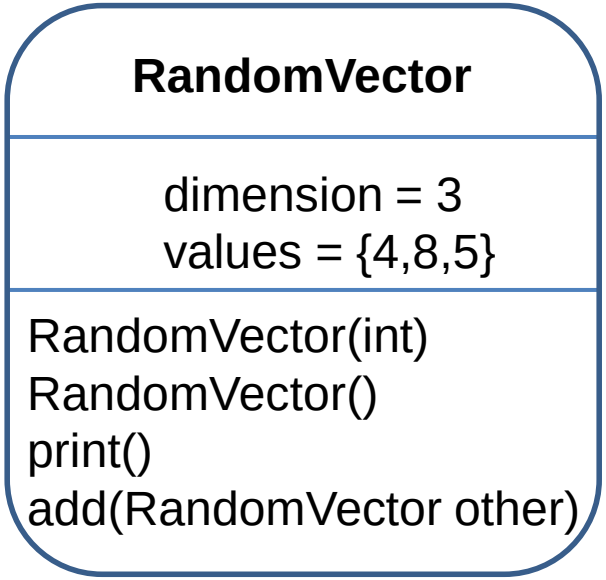


vector1.add(vector2)

vector1 = new RandomVector()



vector2 = new RandomVector()



Η εντολή exit

Χρησιμοποιείται για σοβαρά λάθη για να σταματάει την εκτέλεση του προγράμματος.

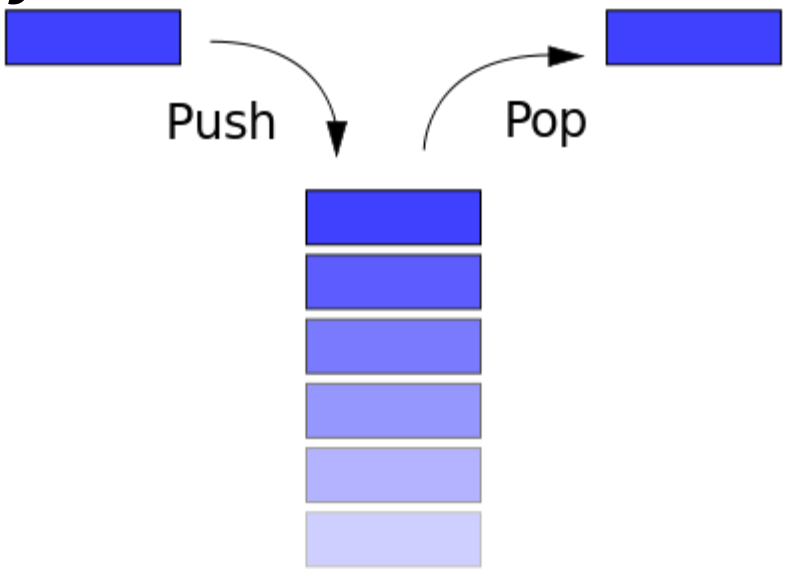
```
public RandomVector(int dimension)
{
      if (dimension < 0){
            System.out.println("Illegal dimension");
            System.exit(-1);
      }
      this.dimension = dimension;
      values = new int[dimension];
      fillValues();
}
```

Αν δώσουμε αρνητική διάσταση το πρόγραμμα μας θα σταματήσει.

Το -1 εξυπηρετεί σαν κωδικός λάθους, μπορείτε να βάλετε όποια τιμή θέλετε.

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.
- Υπολογίστε την συνάρτηση:

$f(x) = 2f(x-1) + 2x + 1,\ f(0) = 1,$

για x=5

```
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?