ΤΕΧΝΙΚΕΣ ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΟΥΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Κλάσεις και Αντικείμενα

Αναφορές

Μαθήματα από το lab

- Υπενθύμιση: Η άσκηση ζητούσε να υλοποιήσετε μία κλάση vector που να διαχειρίζεται διανύσματα οποιουδήποτε μεγέθους.

- Τι πληροφορία πρέπει να κρατάει η κλάση μας?
  - Τη διάσταση του διανύσματος
  - Τις τιμές του διανύσματος.

```
class Vector
{
  public Vector(int dimension)
  {
        int values[] = new int[dimension];
        for (int i=0; i < dimension; i++){
          values[i] = 0;
        }
  }

  public String toString()
  {
        String output = "";
        for (int i = 0; i < dimension; i ++){
          output = output + values[i] + " ";
        }
        return output;
  }
}

class TestVector
{
  public static void main(String[] args){
        Vector v = new Vector(3);
        System.out.println(v);
  }
}
```

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!

```
class Vector
{
  private int dimension=2;
  private int values[];

  public Vector(int dimension)
  {
        int values[] = new int[dimension];
        for (int i=0; i < dimension; i++){
          values[i] = 0;
        }
  }

  public String toString()
  {
        String output = "";
        for (int i = 0; i < dimension; i ++){
          output = output + values[i] + " ";
        }
        return output;
  }
}

class TestVector
{
  public static void main(String[] args){
        Vector v = new Vector(3);
        System.out.println(v);
  }
}
```

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!

```
class Vector
{
  private int dimension;
  private int values[];

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

  public String toString()
  {
        String output = "";
        for (int i = 0; i < dimension; i ++){
          output = output + values[i] + " ";
        }
        return output;
  }
}

class TestVector
{
  public static void main(String[] args){
        Vector v = new Vector(3);
        System.out.println(v);
  }
}
```

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!

```
class Vector
{
  private int dimension;
  private int values[] = new int[dimension];

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

  public String toString()
  {
        String output = "";
        for (int i = 0; i < dimension; i ++){
          output = output + values[i] + " ";
        }
        return output;
  }
}

class TestVector
{
  public static void main(String[] args){
        Vector v = new Vector(3);
        System.out.println(v);
  }
}
```

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!

```
class Vector
{
  private int dimension;
  private int values[];

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

  public String toString()
  {
        String output = "";
        for (int i = 0; i < dimension; i ++){
          output = output + values[i] + " ";
        }
        return output;
  }
}

class TestVector
{
  public static void main(String[] args){
        Vector v = new Vector(3);
        System.out.println(v);
  }
}
```

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!

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 Vector(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

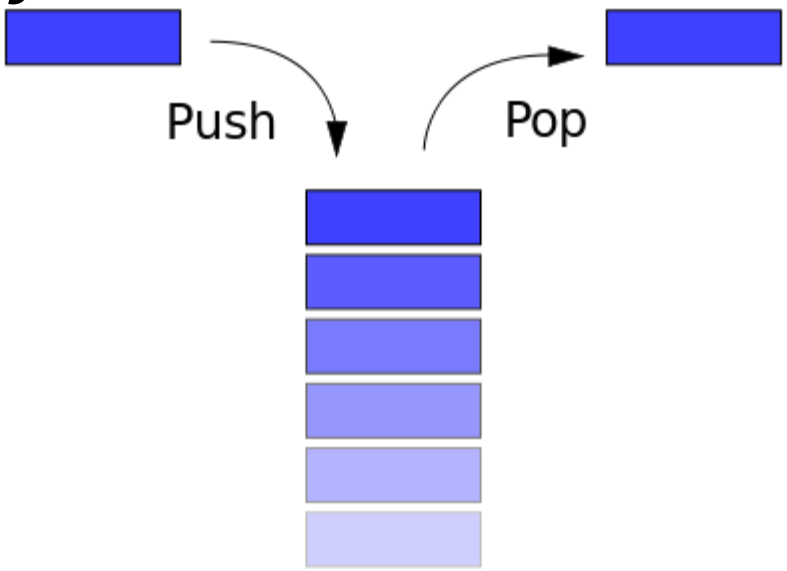
```
public Vector(int dimension)
{
      this.dimension = dimension;
      int values[] = new int[dimension];
      for (int i=0; i < dimension; i++){
        values[i] = 0;
      }
}
```

Οι παράμετροι είναι σαν τοπικές μεταβλητές

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

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?

ΑΝΑΦΟΡΕΣ

new

- Όπως είδαμε για να δημιουργήσουμε ένα αντικείμενο χρειάζεται να καλέσουμε τη new.
  - Για τον πίνακα είπαμε ότι έτσι δίνουμε χώρο στον πίνακα και δεσμεύουμε την απαιτούμενη μνήμη.

- Τι ακριβώς συμβαίνει όταν καλούμε την new?

Η μνήμη του υπολογιστή

- Η κύρια μνήμη (main memory) του υπολογιστή κρατάει τα δεδομένα (και τις εντολές) για την εκτέλεση των προγραμμάτων.
  - Η μνήμη είναι προσωρινή, τα δεδομένα χάνονται όταν ολοκληρωθεί το πρόγραμμα.
- Η μνήμη είναι χωρισμένη σε bytes (8 bits)
  - Ο χώρος που χρειάζεται για ένα χαρακτήρα ASCII.
- Το κάθε byte έχει μια διεύθυνση, με την οποία μπορούμε να προσπελάσουμε τη συγκεκριμένη θέση μνήμης
  - Random Access Memory (RAM)
  - Σε 32-bit συστήματα μια διεύθυνση είναι 32 bits, σε 64-bit συστήματα μια διεύθυνση είναι 64 bits.

Διεύθυνση μνήμης	Περιεχόμενο μνήμης
0000	‘a’
0001	‘b’
0010	‘c’
0011	‘d’
0100	‘e’
0101	‘f’
0110	‘g’
0111	‘h’

Αποθήκευση μεταβλητών

- Η κύρια μνήμη (main memory) του υπολογιστή κρατάει τις μεταβλητές ενός προγράμματος
- Μια μεταβλητή μπορεί να απαιτεί χώρο περισσότερο από 1 byte.
  - Π.χ., οι μεταβλητές τύπου double χρειάζονται 8 bytes.
  - Η μεταβλητή τότε αποθηκεύεται σε συνεχόμενα bytes στη μνήμη.
- Η θέση μνήμης (διεύθυνση) της μεταβλητής θεωρείται το πρώτο byte από το οποίο ξεκινάει η αποθήκευση του της μεταβλητής.
  - Στο παράδειγμα μας η μεταβλητή βρίσκεται στη θέση 0000
  - Αν ξέρουμε την αρχή και το μέγεθος της μεταβλητής μπορούμε να τη διαβάσουμε.
- Άρα μία θέση μνήμης αποτελείται από μία διεύθυνση και το μέγεθος.

Διεύθυνση μνήμης	Περιεχόμενο μνήμης
0000	8.5
0001	
0010	
0011	
0100	
0101	
0110	
0111	

Αποθήκευση μεταβλητών πρωταρχικού τύπου

- Για τις μεταβλητές πρωταρχικού τύπου (char, int, double,…) ξέρουμε εκ των προτέρων το μέγεθος της μνήμης που χρειαζόμαστε.
- Όταν ο μεταγλωττιστής δει τη δήλωση μιας μεταβλητής πρωταρχικού τύπου δεσμεύει μια θέση μνήμης αντίστοιχου μεγέθους
  - Η δήλωση μιας μεταβλητής ουσιαστικά δίνει ένα όνομα σε μία θέση μνήμης
  - Συχνά λέμε η θέση μνήμης x για τη μεταβλητή x.

```
int x = 5;
int y = 3;
```

	Διεύθυνση μνήμης	Περιεχόμενο μνήμης
x	0000	5
	0001	
	0010	
	0011	
y	0100	3
	0101	
	0110	
	0111	

Αποθήκευση αντικειμένων

- Για τα αντικείμενα δεν ξέρουμε πάντα εκ των προτέρων το μέγεθος της μνήμης που θα πρέπει να δεσμεύσουμε.

```
String s; // δεν ξερουμε το μέγεθος του s
s = "ab"; // το s έχει μέγεθος 2 χαρακτήρες
s = "abc"; //  το s έχει μέγεθος 3 χαρακτήρες
```

- Παρομοίως αν δηλώσουμε

```
int[] A;
```

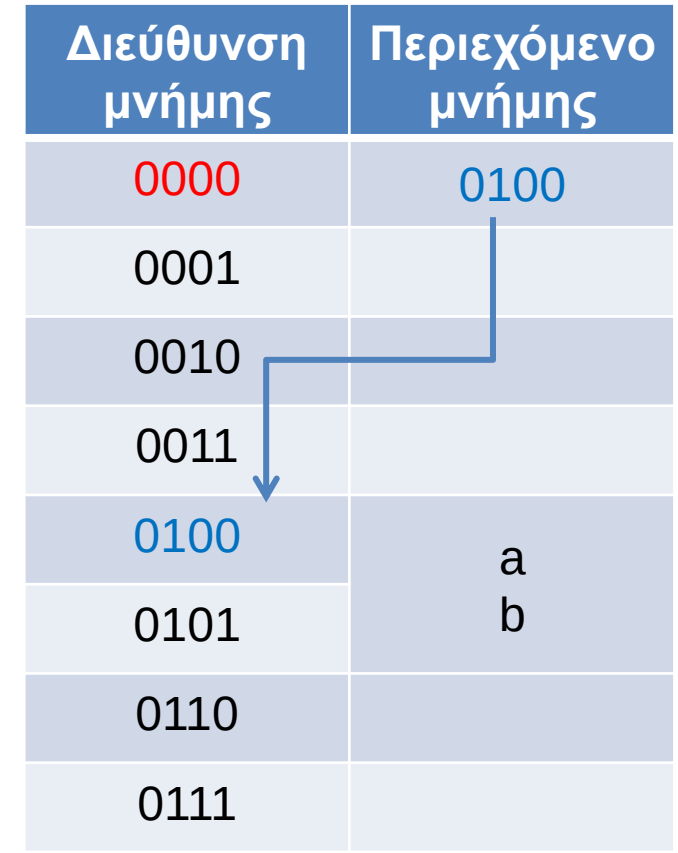
μας λέει ότι έχουμε ένα πίνακα από ακέραιους αλλά δεν μας λέει πόσο μεγάλος θα είναι αυτός ο πίνακας.

```
A = new int[2];
A = new int[3];
```

Αποθήκευση αντικειμένων

- Οι θέσεις μνήμης των αντικειμένων κρατάνε μια διεύθυνση στο χώρο στον οποίο αποθηκεύεται το αντικείμενο
- Η διεύθυνση αυτή λέγεται αναφορά.
- Οι αναφορές είναι παρόμοιες με τους δείκτες σε άλλες γλώσσες προγραμματισμού με τη διαφορά ότι η Java δεν μας αφήνει να πειράξουμε τις διευθύνσεις.
  - Εμείς χρησιμοποιούμε μόνο τη μεταβλητή του αντικειμένου, όχι το περιεχόμενο της
  - Το dereferencing το κάνει η Java αυτόματα.

```
String s = "ab";
```



Παράδειγμα - πινάκες

```
int[] A;
A = new int[2];
A = new int[3];
```

Διεύθυνση μνήμης	Περιεχόμενο μνήμης
0000	
0001	
0010	
0011	
0100	
0101	
0110	
0111	

Παράδειγμα - πινάκες

```
int[] A;
A = new int[2];
A = new int[3];
```

Η δεσμευμένη λέξη null σημαίνει μια κενή αναφορά (δεν δείχνει πουθενά)

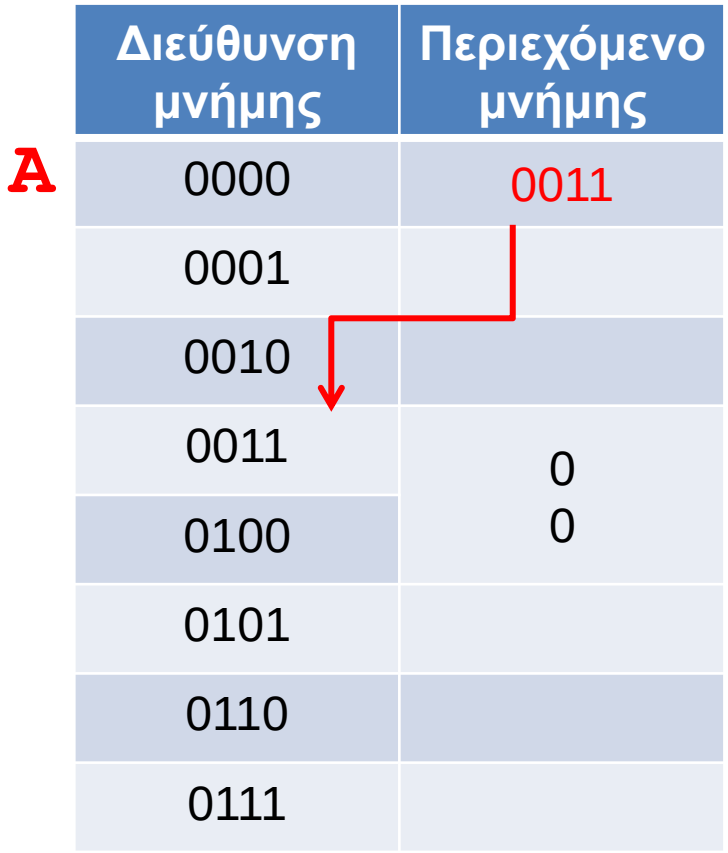
A

Διεύθυνση μνήμης	Περιεχόμενο μνήμης
0000	null
0001	
0010	
0011	
0100	
0101	
0110	
0111	

Παράδειγμα - πινάκες

```
int[] A;
A = new int[2];
A = new int[3];
```

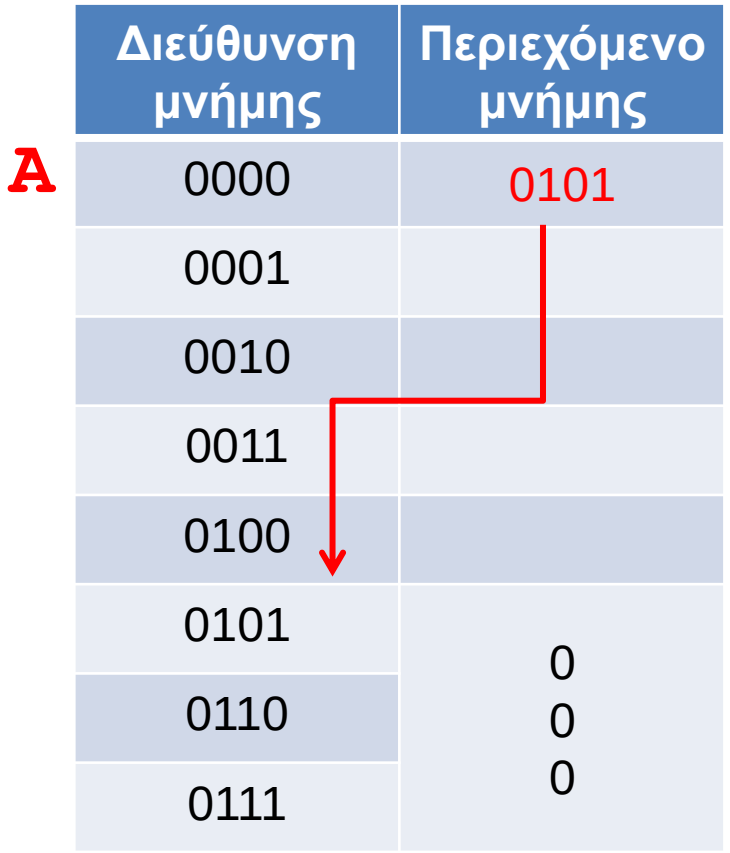
Με την εντολή new δεσμεύουμε δύο θέσεις ακεραίων και η αναφορά του A δείχνει σε αυτό το χώρο που δεσμεύσαμε



Παράδειγμα - πινάκες

```
int[] A;
A = new int[2];
A = new int[3];
```

Με νέα κλήση της new δεσμεύουμε νέο χώρο για το Α, και αν δεν έχουμε κρατήσει την προηγούμενη αναφορά σε κάποια άλλη μεταβλητή τότε χάνεται (garbage collection)



Αντικείμενα κλάσεων

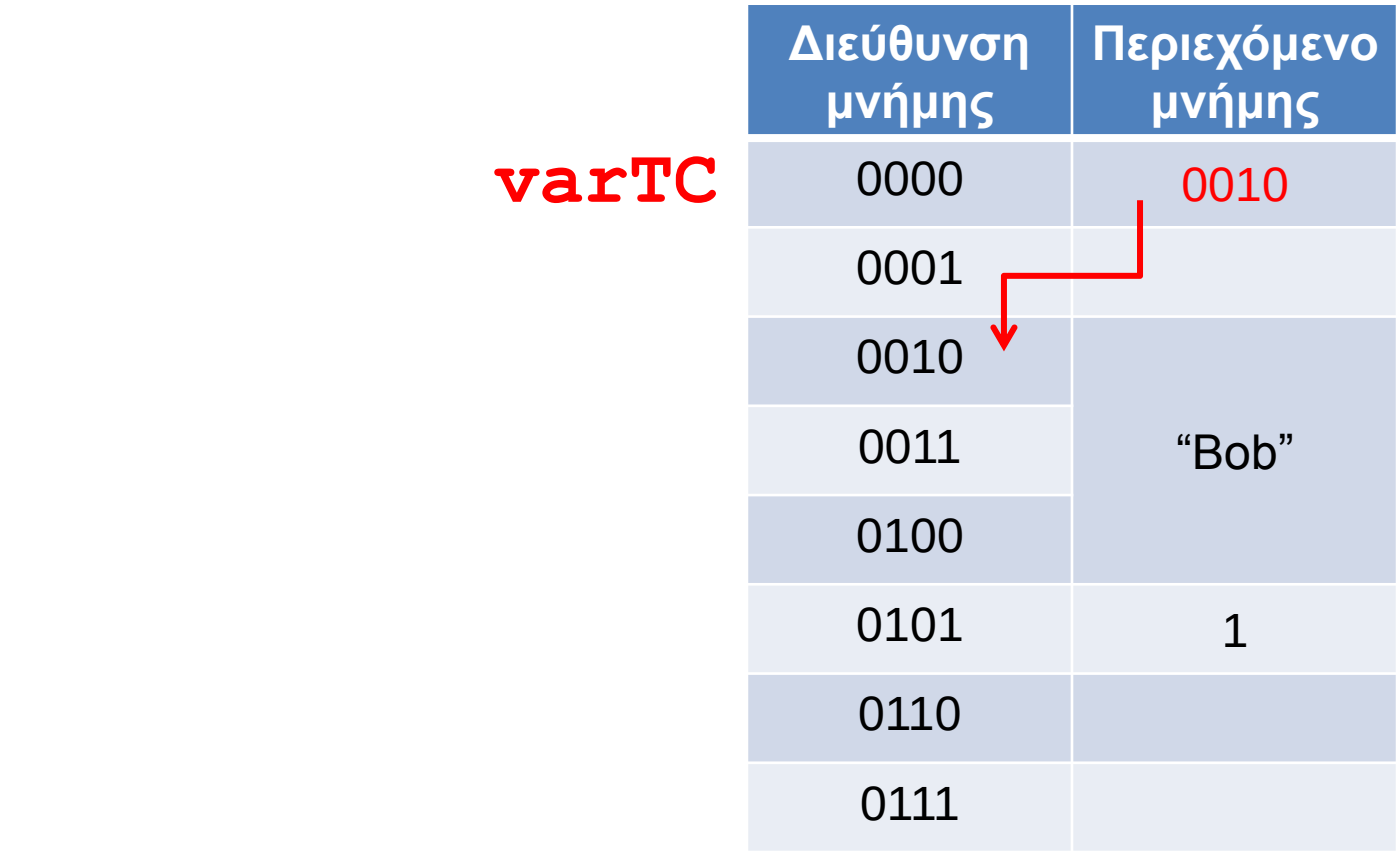
- Τι γίνεται με τα αντικείμενα κλάσεων που ορίσαμε εμείς?
- Παράδειγμα: ToyClass από το βιβλίο.

```
public class ToyClass
{
    private String name;
    private int number;

    public ToyClass(String initialName, int initialNumber){
        name = initialName;
        number = initialNumber;
    }

    public ToyClass( ){
        name = "No name yet.";  number = 0;
    }

    public void set(String newName, int newNumber){
        name = newName;
        number = newNumber;
    }

    public String toString( ){
        return (name + " " + number);
    }

     public void copier(ToyClass aParameter) {
        aParameter.name = name;
        aParameter.number = number;
    }

    public boolean equals(ToyClass otherObject) {
        return ( (name.equals(otherObject.name)) && (number == otherObject.number) );
    }
}
```

Παράδειγμα

```
ToyClass varTC = new ToyClass("Bob", 1);
```

varTC



Διεύθυνση μνήμης	Περιεχόμενο μνήμης
0000	0010
0001	
0010	"Bob"
0011	
0100	
0101	1
0110	
0111	

Αναθέσεις μεταξύ αντικειμένων

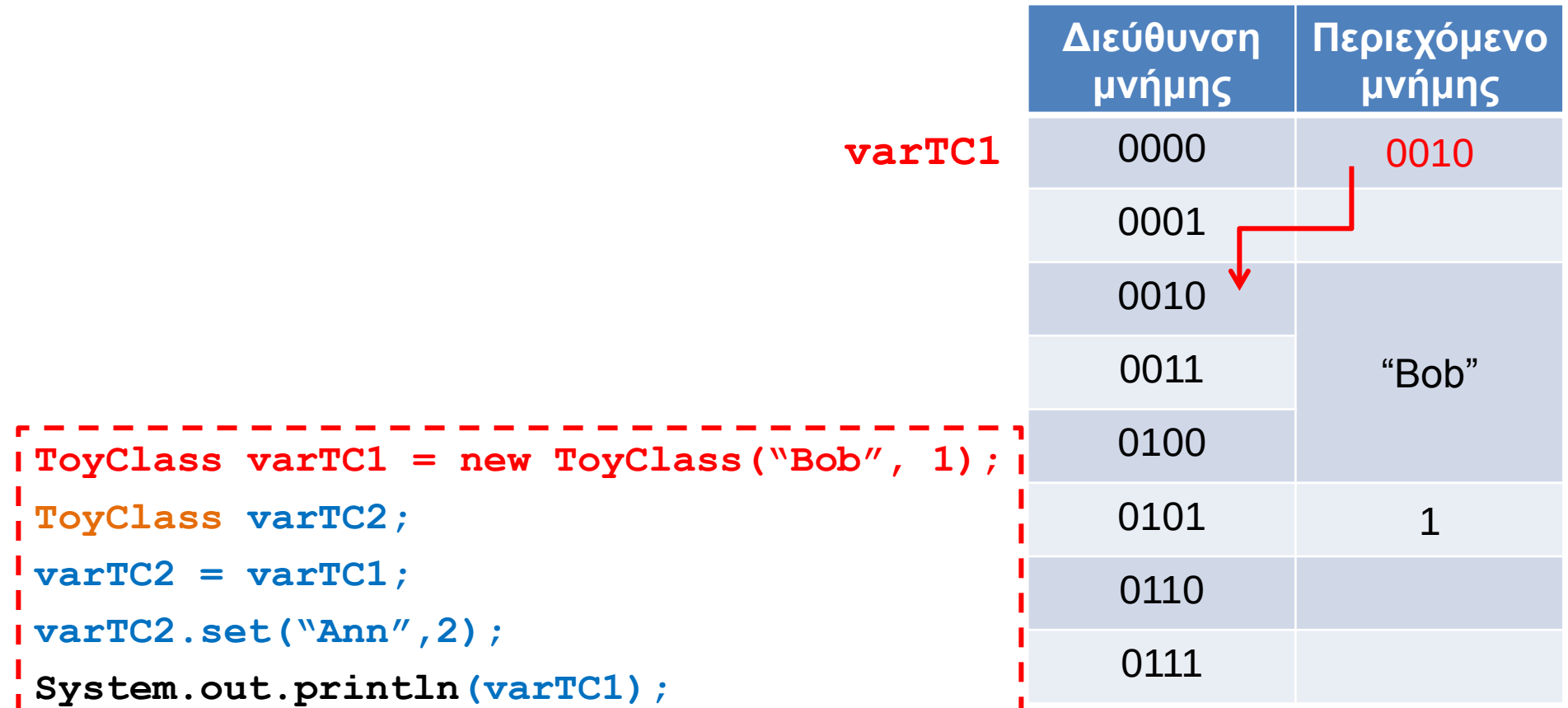
Τι θα τυπώσει το παρακάτω πρόγραμμα?

```
ToyClass varTC1 = new ToyClass("Bob", 1);
ToyClass varTC2;
varTC2 = varTC1;
varTC2.set("Ann",2);
System.out.println(varTC1);
```

Διεύθυνση μνήμης	Περιεχόμενο μνήμης
0000	
0001	
0010	
0011	
0100	
0101	
0110	
0111	

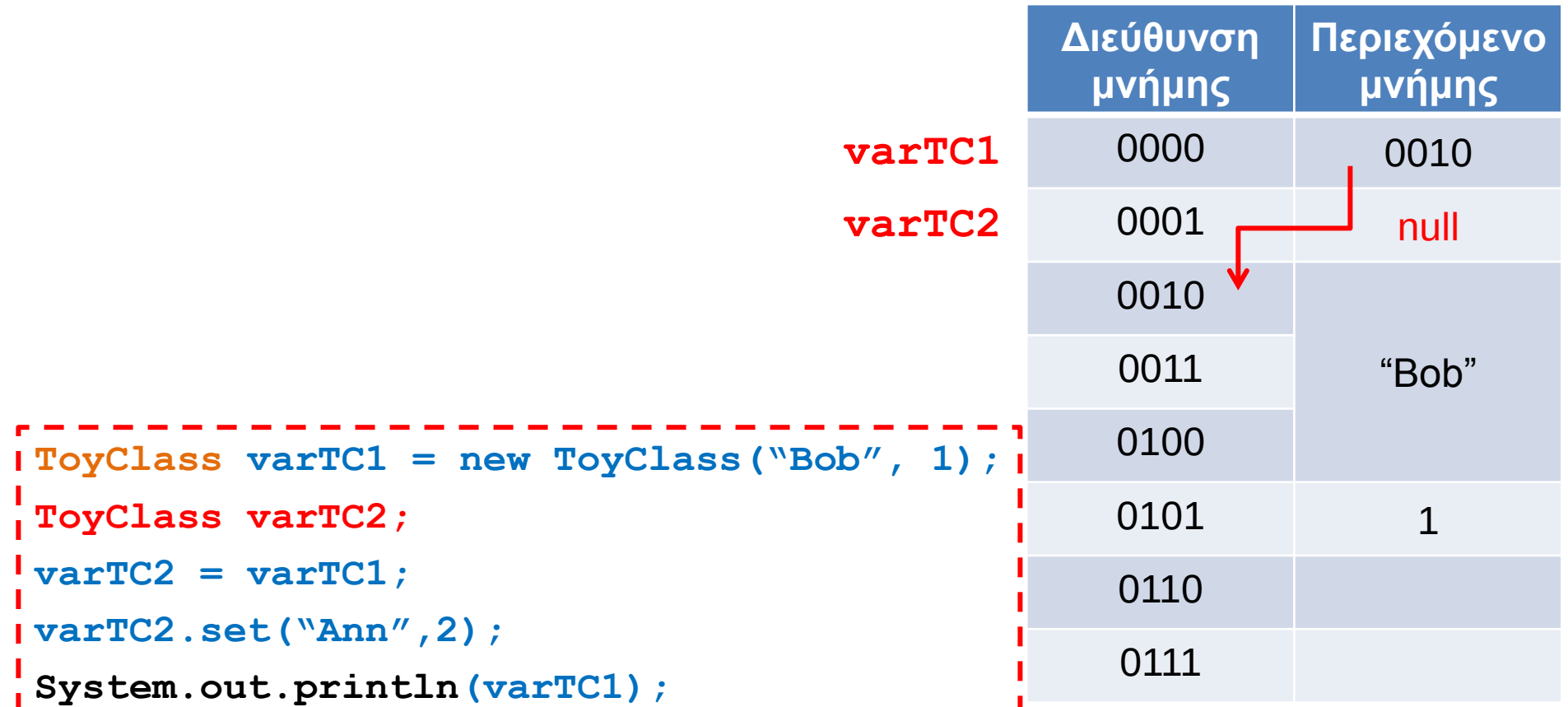
Αναθέσεις μεταξύ αντικειμένων

```
ToyClass varTC1 = new ToyClass("Bob", 1);
ToyClass varTC2;
varTC2 = varTC1;
varTC2.set("Ann",2);
System.out.println(varTC1);
```



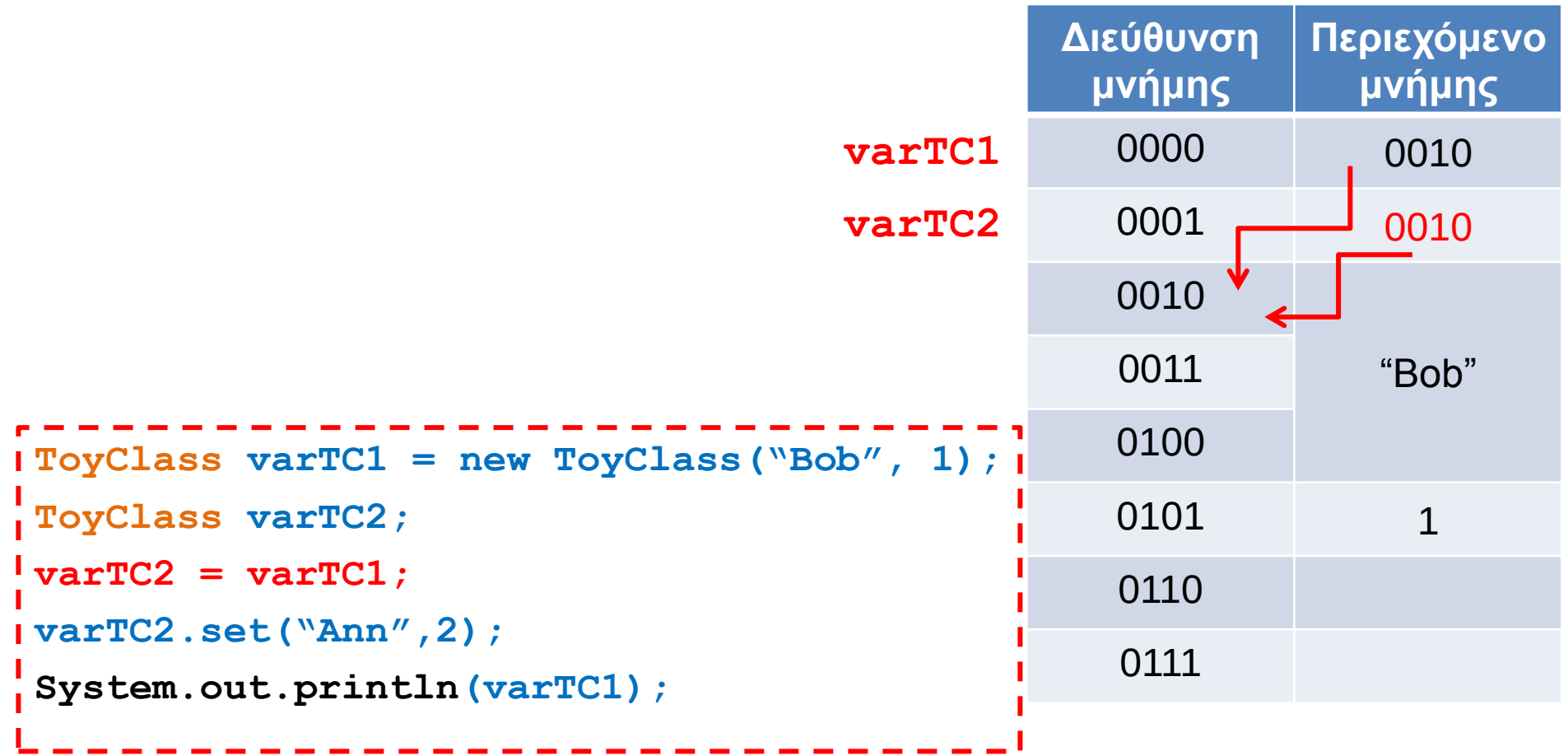
Αναθέσεις μεταξύ αντικειμένων

```
ToyClass varTC1 = new ToyClass("Bob", 1);
ToyClass varTC2;
varTC2 = varTC1;
varTC2.set("Ann",2);
System.out.println(varTC1);
```



Αναθέσεις μεταξύ αντικειμένων

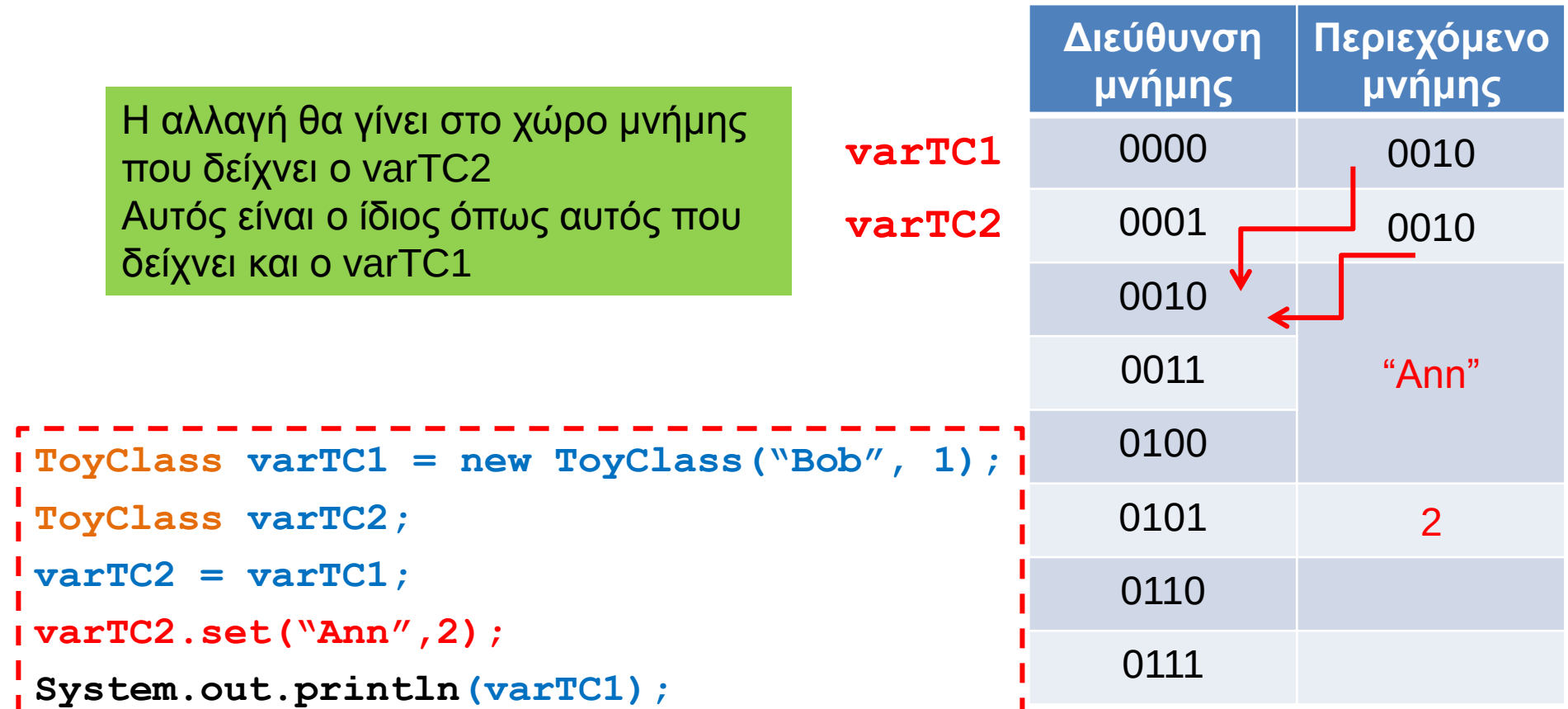
```
ToyClass varTC1 = new ToyClass("Bob", 1);
ToyClass varTC2;
varTC2 = varTC1;
varTC2.set("Ann",2);
System.out.println(varTC1);
```



Αναθέσεις μεταξύ αντικειμένων

Η αλλαγή θα γίνει στο χώρο μνήμης που δείχνει ο varTC2
Αυτός είναι ο ίδιος όπως αυτός που δείχνει και ο varTC1

```
ToyClass varTC1 = new ToyClass("Bob", 1);
ToyClass varTC2;
varTC2 = varTC1;
varTC2.set("Ann",2);
System.out.println(varTC1);
```

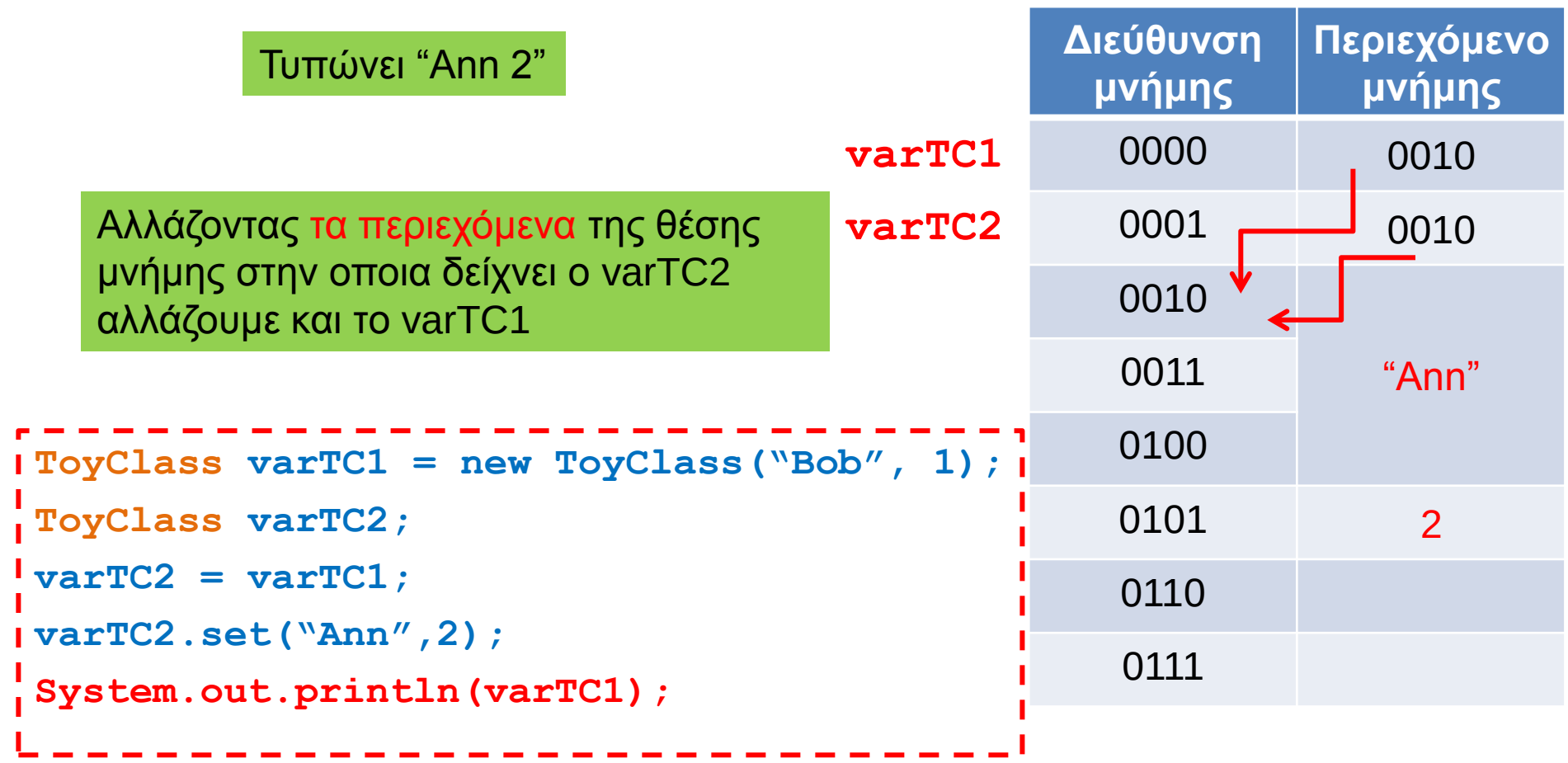


Αναθέσεις μεταξύ αντικειμένων

Τυπώνει “Ann 2”

Αλλάζοντας τα περιεχόμενα της θέσης μνήμης στην οποια δείχνει ο varTC2 αλλάζουμε και το varTC1

```
ToyClass varTC1 = new ToyClass(“Bob”, 1);
ToyClass varTC2;
varTC2 = varTC1;
varTC2.set(“Ann”,2);
System.out.println(varTC1);
```



Equals

- Έχουμε πει ότι όταν ελέγχουμε ισότητα μεταξύ αντικειμένων (π.χ., Strings) πρέπει να γίνεται μέσω της μεθόδου Equals και όχι με το ==
- Η συζήτηση με τις αναφορές εξηγεί γιατί η σύγκριση με == δε δουλεύει

- Η σύγκριση με == συγκρίνει αν δύο αναφορές είναι ίδιες και όχι αν τα περιεχόμενα των θέσεων μνήμης στις οποίες δείχνουν οι αναφορές είναι ίδια.

Αντικείμενα ως παράμετροι

- Όταν περνάμε παραμέτρους σε μία μέθοδο το πέρασμα γίνεται πάντα δια τιμής (call-by-value)
  - Δηλαδή απλά περνάμε τα περιεχόμενα της θέσης μνήμης της συγκεκριμένης μεταβλητής.
  - Για μεταβλητές πρωταρχικού τύπου, αλλαγές στην τιμή της παραμέτρου δεν αλλάζουν την μεταβλητή που περάσαμε σαν όρισμα.

- Τι γίνεται όμως αν η παράμετρος είναι ένα αντικείμενο?
  - Τα περιεχόμενα της θέσης μνήμης μιας μεταβλητής-αντικείμενο είναι μια αναφορά.
  - Αν μέσα στην μέθοδο αλλάξουν τα περιεχόμενα του αντικειμένου (εκεί που δείχνει η αναφορά) τότε αλλάζει και η μεταβλητή που περάσαμε.

Παράδειγμα

```
public class ClassParameterDemo
{
    public static void main(String[] args)
    {
        ToyClass anObject = new ToyClass("Mr. Cellophane", 0);
        System.out.println(anObject);
        ToyClass anotherObject = new ToyClass("Hot Shot", 42);
        System.out.println(
                "Now we call changer with anObject as argument.");
        anotherObject.copier(anObject);
        System.out.println(anObject);
    }
}
```

Τι θα τυπώσει?

```
public class ToyClass
{
    private String name;
    private int number;

    public void copier(ToyClass aParameter) {
        aParameter.name = name;
        aParameter.number = number;
    }

}
```

Hot Shot 42

Εξήγηση

```
ToyClass anObject = new
   ToyClass("Mr. Cellophane", 0);
ToyClass anotherObject = new
   ToyClass("Hot Shot", 42);
```

Διεύθυνση μνήμης	Περιεχόμενο μνήμης
0010 (anObject)	0200
0011 (anotherObject)	0300
0100	
…	
0200	“Mr Cellophane” 0
0300	“Hot Shot” 42
0110	
0111	

Εξήγηση

Διεύθυνση μνήμης	Περιεχόμενο μνήμης
0010	0200
0011	0300
0100	0200
…	
0200	“Mr Cellophane” 0
0300	“Hot Shot” 42
0110	
0111	

anObject

anotherObject

aParameter

```
anotherObject.copier(anObject);
```

```
public class ToyClass
{
    private String name;
    private int number;

    public void copier(ToyClass aParameter)
    {
        aParameter.name = name;
        aParameter.number = number;
    }

}
```

aParameter = anObject

Εξήγηση

anObject

anotherObject

aParameter

```
anotherObject.copier(anObject);
```

```
public class ToyClass
{
    private String name;
    private int number;

    public void copier(ToyClass aParameter)
    {
        aParameter.name = name;
        aParameter.number = number;
    }

}
```

Διεύθυνση μνήμης	Περιεχόμενο μνήμης
0010	0200
0011	0300
0100	0200
…	
0200	“Hot Shot” 42
0300	“Hot Shot” 42
0110	
0111	