Αντικειμενοστρεφής Προγραμματισμός
Διάλεξη – 7 :
ΣΥΝΘΕΤΕΣ ΚΛΑΣΕΙΣ ΚΑΙ ΜΕΘΟΔΟΙ
Pass-by-value και φαινομενικό pass-by-reference
Η κλαση string
h κλαση stringbuffer

Pass-by-value και φαινομενικό pass-by-reference 1/6

- Γενικά, στις γλώσσες προγραμματισμού, οι όροι "pass-by-value" (πέρασμα με τιμή) και "pass-by-reference" (πέρασμα με αναφορά) αναφέρονται στον τρόπο με τον οποίο περνάει μία μέθοδος (που καλεί κάποια άλλη) τα ορίσματα εισόδου στην καλούμενη μέθοδο.
- Ένα τέτοιο πέρασμα (pass) μπορεί να γίνει είτε περνώντας τις τιμές των ορισμάτων αυτών στην καλούμενη μέθοδο (pass-by-value), είτε περνώντας αναφορές (στη θέση μνήμης) των ορισμάτων αυτών (pass-by-reference).

Pass-by-value και φαινομενικό pass-by-reference 2/6

Έστω μια κλάση υποστήριξης ClassB:

```
public class ClassB {
        private int x, y;
        public ClassB(int x, int y) {
                this.x = x;
                this.y = y;
        }
        public void setValues(int a, int b) {
                x = a; y = b;
        }
        public int getX()  {
                return x;
        }
        public int getY() {
                return y;
        }
}
```

Pass-by-value και φαινομενικό pass-by-reference 3/6

Έστω η κλάση εφαρμογής Program που περιέχει τη main και μία μέθοδο add5:

```
public class Program {
        public static void main (String[] args) {
                ClassB obj = new ClassB(5,10);
                int z = 100;
                System.out.println("Τιμές των x, y και z πριν την κλήση της μεθόδου:");
                System.out.println("x = " + obj.getX());
                System.out.println("y = " + obj.getY());
                System.out.println("z = " + z);
                add5(obj,z); // Κλήση της add5 που ορίζεται παρακάτω
                System.out.println("Τιμές των x, y και z μετά την κλήση της μεθόδου:");
                System.out.println("x = " + obj.getX());
                System.out.println("y = " + obj.getY());
                System.out.println("z = " + z);
        }
        public static void add5(ClassB obj, int z) {
                obj.setValues(obj.getX()+5, obj.getY()+5);
                z+=5;
        }
}
```

Pass-by-value και φαινομενικό pass-by-reference 4/6

Το πρόγραμμα αυτό θα εκτυπώσει τα εξής:

Τιμές των x, y και z πριν την κλήση της μεθόδου:

x = 5

y = 10

z = 100

Τιμές των x, y και z μετά την κλήση της μεθόδου:

x = 10

y = 15

z = 100

Δηλαδή, η τιμή της μεταβλητής z δεν άλλαξε μετά την κλήση της μεθόδου add5, ενώ αντιθέτως οι τιμές των μεταβλητών x και y του αντικειμένου obj της κλάσης ClassB άλλαξαν.

Στη Java όλες οι παράμετροι περνιούνται με τιμή (pass-by-value) από τη μέθοδο που κάνει την κλήση στη μέθοδο που καλείται. Αυτό ισχύει *και* για τις μεταβλητές *και* για τα αντικείμενα. Δηλαδή, και τα αντικείμενα περνιούνται με τιμή (pass-by-value).

Θεωρητικά στη Java δεν υπάρχει πέρασμα με αναφορά (pass-by-reference) όπως υπάρχει σε άλλες γλώσσες προγραμματισμού. Τότε γιατί στο παραπάνω παράδειγμα φαίνεται να γίνεται pass-by-reference στην περίπτωση του αντικειμένου; Αυτό συμβαίνει γιατί στη Java το κάθε αντικείμενο ουσιαστικά είναι μια αναφορά στη θέση μνήμης που περιέχονται τα στοιχεία (δεδομένα) του αντικειμένου.

Pass-by-value και φαινομενικό pass-by-reference 5/6

- Για να κατανοήσει κανείς τι ακριβώς συμβαίνει με τα αντικείμενα, πρέπει καταρχάς να κατανοήσει τι συμβαίνει όταν μια απλή μεταβλητή περνιέται με τιμή κατά την κλήση μιας μεθόδου.
- Όταν καλείται μια μέθοδος που δέχεται μια παράμετρο εισόδου, η μεταβλητή που δίνεται από τη μέθοδο που κάνει την κλήση στη θέση της παραμέτρου, αντιγράφεται σε μία νέα προσωρινή μεταβλητή που θα χρησιμοποιηθεί ως παράμετρος για την κλήση.
- Το ίδιο συμβαίνει και στην περίπτωση που αντί για μεταβλητή περνιέται ένα αντικείμενο σαν παράμετρος εισόδου στην καλούμενη μέθοδο: το αντικείμενο αντιγράφεται σε ένα προσωρινό αντικείμενο που αποστέλλεται στη μέθοδο που καλείται.

Pass-by-value και φαινομενικό pass-by-reference 6/6

- *Όμως, αυτό το προσωρινό αντικείμενο δεν είναι τίποτε άλλο από μια διαφορετική αναφορά στο ίδιο αρχικό αντικείμενο* (αφού τα αντικείμενα είναι αναφορές στις θέσεις μνήμης που περιέχονται τα αντικείμενα). Επομένως, ό,τι αλλαγές γίνουν στα δεδομένα του αντικειμένου από την κληθείσα μέθοδο, θα ισχύουν και για το αντικείμενο που περάστηκε ως παράμετρος στην καλούσα μέθοδο, αφού πρόκειται για το ίδιο αντικείμενο.
- Για αυτό το λόγο, στο παραπάνω παράδειγμα οι μεταβλητές του αντικειμένου obj παραμένουν αλλαγμένες μετά την επαναφορά της ροής του κώδικα από τη μέθοδο add5 στη μέθοδο main (μετά την κλήση της μεθόδου add5), σε αντίθεση με την απλή μεταβλητή z, η οποία παραμένει αμετάβλητη στην αρχική της τιμή.

ΣΗΜΕΙΩΣΕΙΣ ΓΙΑ Pass-by-value και το pass-by-reference 1/4

- Αν μέσα στη μέθοδο add5 ή σε κάποια άλλη αντίστοιχη μέθοδο δεν άλλαζαν οι τιμές μεταβλητών του αντικειμένου obj, αλλά άλλαζε *ολόκληρο* το αντικείμενο (σε αντιστοιχία με την αλλαγή μιας απλής μεταβλητής), τότε η αλλαγή αυτή θα έμενε στη μέθοδο και δε θα "πέρναγε" στη μέθοδο main, δηλαδή θα συνέβαινε ό,τι ακριβώς συμβαίνει και με τις απλές μεταβλητές.
- Αυτό είναι μια "απόδειξη" ότι *και τα αντικείμενα ουσιαστικά γίνονται pass-by-value στη Java*. Π.χ., αν στο παραπάνω παράδειγμα, στην κλάση Program υπήρχε αντί της μεθόδου add5 η μέθοδος change, οπότε η Program γινόταν ως εξής:

ΣΗΜΕΙΩΣΕΙΣ ΓΙΑ Pass-by-value και το pass-by-reference 2/4

```
public class Program {
        public static void main (String[] args) {
                ClassB obj = new ClassB(5,10);
                int z = 100;
                System.out.println("Τιμές των x, y και z πριν την κλήση της μεθόδου:");
                System.out.println("x = " + obj.getX());
                System.out.println("y = " + obj.getY());
                System.out.println("z = " + z);
                change(obj,z); // Κλήση της change που ορίζεται παρακάτω
                System.out.println("Τιμές των x, y και z μετά την κλήση της μεθόδου:");
                System.out.println("x = " + obj.getX());
                System.out.println("y = " + obj.getY());
                System.out.println("z = " + z);
        }
        public static void change(ClassB obj, int z) {
                ClassB newObj = new ClassB(50,55);
                obj = newObj;
                z+=5;
        }
}
```

ΣΗΜΕΙΩΣΕΙΣ ΓΙΑ Pass-by-value και το pass-by-reference 3/4

τότε το πρόγραμμα θα εκτύπωνε τα εξής:

Τιμές των x, y και z πριν την κλήση της μεθόδου:

x = 5

y = 10

z = 100

Τιμές των x, y και z μετά την κλήση της μεθόδου:

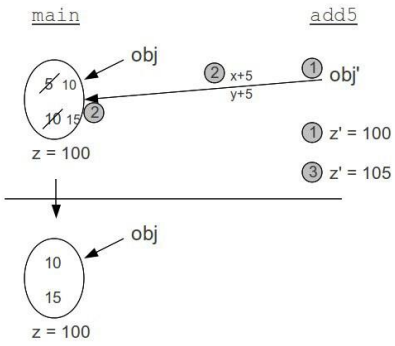
x = 5

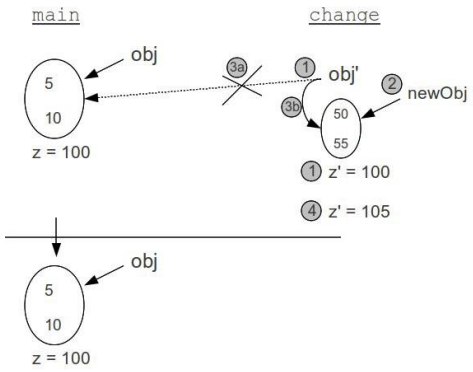
y = 10

z = 100

Πλέον η συμπεριφορά των τιμών των x και y του αντικειμένου obj είναι η ίδια με αυτή μιας κανονικής μεταβλητής, δηλαδή παραμένουν στις προηγούμενες τιμές τους (5 και 10 αντίστοιχα), πράγμα που σημαίνει ότι όντως το αντικείμενο obj περάστηκε με τιμή (pass-by-value) στη μέθοδο change και δε μεταβλήθηκε από αυτή. Το μόνο που άλλαξε σε σχέση με πριν είναι ότι πλέον η μέθοδος δεν τροποποιεί μεμονωμένα τις τιμές των μεταβλητών του αντικειμένου που δέχεται, αλλά τροποποιεί "ολόκληρο" το αντικείμενο, άρα το μόνο που τροποποιεί είναι την αναφορά στη θέση μνήμης του προσωρινού αντικειμένου που δημιουργείται, το οποίο τώρα απλά αναφέρεται στη θέση μνήμης ενός νέου αντικειμένου (με τιμές μεταβλητών 50 και 55). Οι μεταβλητές του αρχικού αντικειμένου obj φυσικά παραμένουν αμετάβλητες μέσα στη μέθοδο main, όπως ακριβώς συμβαίνει και με τη μεταβλητή z.

ΣΗΜΕΙΩΣΕΙΣ ΓΙΑ Pass-by-value και το pass-by-reference 4/4





Η Κλάση String 1/2

- Κάθε συμβολοσειρά (ή αλφαριθμητικό) (string) είναι αντικείμενο της κλάσης String.
- Βασική ιδιότητα: Τα string, δηλ. τα αντικείμενα της κλάσης String, δεν τροποποιούνται.
- Τρόποι δημιουργίας συμβολοσειρών :
  - Με τη χρήση εισαγωγικών: String str = "Hello "
  - Με τη χρήση των + ή += πάνω σε υπάρχοντα strings

    "Hello " + "world" → νέο string → Hello world
  - Με κανονική δημιουργία αντικειμένου μέσω της new.
- Η String έχει δύο κατασκευαστές (constructors):
  - public String() και public String(String value)
    - π.χ. String str1 = new String();
  - String str2 = new String("Hello");

Η Κλάση String 2/2

- Προσοχή στον τελεστή + :

```
double x=15, y=25;
System.out.println(x+y);
System.out.println("Το άθροισμα είναι " + x + y);
System.out.println("Το άθροισμα είναι " + (x + y));
```

- Ο κώδικας αυτός θα εκτυπώσει:
- `40.0`
- `Το άθροισμα είναι 15.025.0`
- `Το άθροισμα είναι 40.0`
- Δηλαδή, όταν ο τελεστής + βρει κάποιο string, προσθέτει απλά τις τιμές των μεταβλητών που βρίσκει μετά σαν συνέχεια στο string αυτό. Άρα, για να γίνει η πράξη της πρόσθεσης των δύο double μεταβλητών, θα πρέπει να μπει σε παρένθεση για να προηγηθεί της συνένωσης με το string.

Βασικές μέθοδοι της κλάσης String

- public int length()→ επιστρέφει το πλήθος των χαρακτήρων του αλφ/κού
  - π.χ. String a = "blah blah";
  - int b = a.length(); // → b = 9
- public char charAt(int index) → επιστρέφει τον χαρακτήρα στη θέση index
  - char d = a.charAt(3); // → d = 'h' *(μετράμε από το 0!)*
- public int indexOf(char ch) → επιστρέφει την πρώτη θέση του ch
  - a.indexOf('a'); → 2
- public int indexOf(char ch, int start) → επιστρέφει την πρώτη θέση του ch από τη θέση start και μετά.
  - a.indexOf('a',3); → 7
- public int indexOf(String str) → επιστρέφει την πρώτη θέση του str
- public int indexOf(String str, int start) →επιστρέφει την πρώτη θέση του str από τη θέση start και μετά.
- public int lastIndexOf(char ch) → επιστρέφει την τελευταία θέση του ch
- public int lastIndexOf(char ch, int start) → επιστρέφει την τελευταία μέχρι και το start θέση του ch
- Όλες αυτές οι μέθοδοι αν δεν βρουν αυτό που ψάχνουν στο string (το ch ή το str), επιστρέφουν -1.

Μέθοδοι σύγκρισης String **1/2**

- public boolean equals(String str)
  - true, αν βρει ίδιο μήκος και ακριβώς ίδιους χαρακτήρες
  - false, σε διαφορετική περίπτωση
  - π.χ. String e = "blah blah";
  - boolean f = a.equals(e); // → f = true
  - *Ο τελεστής == όταν εφαρμοστεί σε δύο string συγκρίνει τις θέσεις μνήμης τους και όχι το περιεχόμενό τους.*
- public boolean equalsIgnoreCase(String str)
  - Αγνοεί κεφαλαία ή πεζά γράμματα
- public int compareTo(String str)
  - < 0 αν κάποιο string έχει < μήκος του str
  - = 0 αν κάποιο string έχει = μήκος με το str
  - > 0 αν κάποιο string έχει > μήκος του str
  - String h = "blah";
  - int i = h.compareTo(a); // → i<0

Μέθοδοι σύγκρισης String **2/2**

- public boolean regionMatches(int start, String str, int strStart, int len)
  - h.regionMatches(1,a,6,3); // → true
  - h → b l a h
    0 1 2 3
  - a → b l a h b l a h
    0 1 2 3 4 5 6 7 8
  - boolean b = a.regionMatches(2,"Ah",0,2); // → false (ah ≠ Ah)
- public boolean regionMatches(boolean ignoreCase, int start, String str, int strStart, int len)
  - boolean b = a.regionMatches(true,2,"Ah",0,2); // → true

Μέθοδοι έλεγχου αρχής τέλους

- public boolean startsWith(String prefix)
- public boolean endsWith(String suffix)
  - boolean b = a.endsWith("ah") → true

Μέθοδοι δημιουργίας νέων αντικειμένων String

- public String replace(char oldChar, char newChar)
  - String j = a.replace('a','i'); // → blih blih
- Δεν αλλάζει το string a. Φτιάχνει νέο string (αντικείμενο)
- public String toLowerCase()
- public String toUpperCase()
- public String trim() → κόβει τα κενά σε αρχή και τέλος
- public String concat(String str) → το ίδιο με το +
  - // Από πριν: a=blah blah και h=blah
  - String s = a.concat(h); // → blah blahblah (ίδιο με το a+h;)

Μετατροπές από/σε String 1/2

- Από μεταβλητή τύπου boolean, int, long, float, double, char σε String:
  - String.valueOf(<μεταβλητή>);
- π.χ.,
  - int n = 10;
  - String p = String.valueOf(n); // → p = "10"
  - Από String σε boolean → Boolean.parseBoolean(str)
  - σε int → Integer.parseInt(str)
  - σε long → Long.parseLong(str)
  - σε float → new Float(str).floatValue
  - σε double → Double.parseDouble(str)

Μετατροπές από/σε String 2/2

- Ο χαρακτήρας \ χρησιμοποιείται σε ειδικές περιπτώσεις, όπως για την εκτύπωση των εισαγωγικών:
  - System.out.println("Say \"Hi\"!"); → τυπώνει: Say "Hi"!
  - (το παραπάνω string έχει 9 χαρακτήρες και όχι 11. Το \ δε μετράει σαν χαρακτήρας)
- Για εκτύπωση του \: \\
  - System.out.println("abc\\def"); → τυπώνει: abc\def
- \n → νέα γραμμή (ENTER)
- \t → tab

Η Κλάση StringBuffer 1/3

- Είναι η κλάση για τη δημιουργία συμβολοσειρών που μπορούν να τροποποιηθούν.
- Οι βασικοί της κατασκευαστές είναι οι εξής:
  - **public StringBuffer()** - δημιουργία κενής συμβολοσειράς 16 θέσεων
  - **public StringBuffer(String str)** - δημιουργία τροποποιήσιμης συμβολοσειράς με αρχική τιμή την τιμή str
  - **public StringBuffer(int capacity)** - δημιουργία κενής συμβολοσειράς δεδομένου αριθμού θέσεων (capacity)
- Στην ουσία τα αντικείμενα της StringBuffer είναι διανύσματα χαρακτήρων μεταβλητού μεγέθους. Μια μέθοδος που χρησιμοποιείται συχνά για τροποποίηση μιας συμβολοσειράς (όχι δημιουργία καινούριας), είναι η setCharAt:
  - **public void setCharAt(int index, char newChar)**

η οποία αντικαθιστά τον υπάρχοντα χαρακτήρα στη θέση index μιας συμβολοσειράς με το χαρακτήρα newChar.

Η Κλάση StringBuffer 2/3

- Η βασικότερη μέθοδος της κλάσης StringBuffer είναι η μέθοδος append η οποία υπάρχει σε πολλές *υπερφορτωμένες* μορφές. Η κύρια μορφή της είναι αυτή της συνένωσης συμβολοσειρών:
  - **public void append(String str)**
- η οποία προσθέτει τη συμβολοσειρά str σε κάποιο συγκεκριμένο StringBuffer.
- Έστω ότι έχουν δηλωθεί και έχουν πάρει τιμές τρεις μεταβλητές τύπου String, οι: title, firstName και lastName.

Η Κλάση StringBuffer 3/3

- Εάν προσπαθούσε κάποιος να ενώσει αυτές τις τρεις συμβολοσειρές συνενώνοντας String και αποθηκεύοντας το αποτέλεσμα σε ένα νέο String με την εντολή:

```
String name = title + " " + firstName + " " + lastName;
```

τότε στην ουσία θα είχε δημιουργήσει 4 ενδιάμεσα String μέχρι να επιτευχθεί η τελική συμβολοσειρά. Αντιθέτως, με τη χρήση της μεθόδου append της StringBuffer, η συνένωση των συμβολοσειρών θα γινόταν με τροποποίηση μιας μόνο συμβολοσειράς:

```
StringBuffer name = new StringBuffer().append(title).append(" ").
        append(firstName).append(" ").append(lastName);
```

Κλάσεις που δημιουργούνται μόνο από δεδομένα μόνο πρωταρχικών τύπων

```
public class Human {
   private String name;
   private int height;
   public Human(){name="";height=150;}
   public Human(String name, int height){
      this.name=name;
      this.height=height;
   }
   @Override
   public String toString(){
      String result;
      result = "name: "+name+" height: "+height;
      return result;
   }
}
```

Driver

```
public class TestHuman {
    public static void main(String[ ] args){
        Human[ ] h = new Human[3];

        h[0]=new Human("lokijg",170);
        h[1]=new Human("lo",171);
        h[2]=new Human("kijg",172);

       for(int i=0; i<h.length; i++){
            System.out.println(h[i]);
       }
    }
}
```

Ένα παράδειγμα

Έστω μια κλάση που αναπαριστά μια ευθεία. Όπως γνωρίζουμε από τη γεωμετρία, μια ευθεία ορίζεται από δύο σημεία. Για το λόγο αυτό, προκειμένου να υλοποιηθεί μια κλάση StraightLine που αναπαριστά μια ευθεία, θα χρησιμοποιήσουμε ως μεταβλητές υπόστασης δύο μεταβλητές τύπου Point, όπου Point μια κλάση που αναπαριστά ένα σημείο στο επίπεδο.

ΚΛΑΣΗ Point

```
public class Point {
   private int x;
   private int y;

   public Point(int x, int y){
      this.x=x;
      this.y=y;
   }
   public String toString(){
      return "("+x+","+y+")";
   }
}
```

Κλάση TestPoint

```
public class TestPoint {
    public static void main(String[] args){
        Point[] p = new Point[10];

        for (int i=0; i<p.length; i++){
            p[i] = new Point(i, 2*i);
        }
        for (int i=0; i<p.length; i++){
            System.out.println(p[i]);
        }
    }
}
```

Σύνθεση αντικειμένων

- Ένα σχολείο/τμήμα πανεπιστημίου αποτελείται από πολλούς μαθητές/φοιτητές.
- Κάθε μαθητής/φοιτητής μπορεί να δηλώσει πολλά μαθήματα.

ΕΡΩΤΗΣΗ : Πως θα μπορούσαμε να προσομοιώσουμε το παραπάνω υπαρκτό πρόβλημα???

Η κλάση Course

```
public class Course {
    int courseID;
    int grade;

    public Course(int id, int bathmos){
        courseID=id;
        grade=bathmos;
    }
    public String toString(){
        return courseID+" "+grade;
    }
}
```

Η κλάση Student

```
import java.util.ArrayList;
public class Student {
    int am;
    String name;
    ArrayList<Course> courses;

    public Student(int am, String name){
        this.am=am;
        this.name=name;
        courses = new ArrayList<>();
    }
    public Student(){
        am=1;name="";
        courses = new ArrayList<>();
    }
    public void addCourse(Course c){
        courses.add(c);
    }
    @Override
    public String toString(){
        return am+" "+name;
    }
}
```

Η κλάση TestStudent

```
import static java.lang.System.out;
import java.util.Scanner;
public class TestStudent {
   public static void main(String[] args){
      Scanner in = new Scanner(System.in);
      Student[] s = new Student[3];
      int count=0;
      int choice,am;
      String name;
      while(true){
         out.println("1.insert student");
         out.println("2.print students");
         out.println("3.search by am");
         out.println("0. exit");
         out.print("choice: ");
         choice = in.nextInt();
         switch(choice){
            case 1://insert student
               if (count==s.length){
                  out.println("array full!!!");
                  break;
               }
               out.println("am: ");
               am=in.nextInt();in.nextLine();
               out.println("name: ");
               name= in.nextLine();
               s[count] = new Student(am, name);
               //insert grades
               /*
               for(int i=0; i<s[count].grades.length; i++){
                  out.println("grade: ");
                  int bathmos=in.nextInt();

//s[count].grades[i]=bathmos;
               }
               * */
               count++;
               break;
```

Η κλάση TestStudent (1/2)

```
import static java.lang.System.out;
import java.util.Scanner;
public class TestStudent {
  public static void main(String[] args){
     Scanner in = new Scanner(System.in);
     Student[] s = new Student[3];
     int count=0;
     int choice,am;
     String name;
     while(true){
        out.println("1.insert student");
        out.println("2.print students");
        out.println("3.search by am");
        out.println("0. exit");
        out.print("choice: ");
        choice = in.nextInt();
        switch(choice){
           case 1://insert student
              if (count==s.length){
                 out.println("array full!!!");
                 break;
              }
              out.println("am: ");
              am=in.nextInt();in.nextLine();
              out.println("name: ");
              name= in.nextLine();
              s[count] = new Student(am, name);
              //insert grades
              /*
              for(int i=0; i<s[count].grades.length; i++){
                 out.println("grade: ");
                 int bathmos=in.nextInt();
                 //s[count].grades[i]=bathmos;
              }
              * */
              count++;
              break;
```

Η κλάση TestStudent (2/2)

```
           case  2:
              for(int i=0; i<count; i++){
                 out.println(s[i]);
                 for(int j=0; j<2; j++){
                    //out.println(s[i].grades[j]);
                 }
              }
              break;
           case 3:
              out.println("am: ");
              am=in.nextInt();
              int pos=-1;
              for(int i=0; i<count; i++){
                 if(s[i].am == am){
                    pos=i;
                    break;
                 }
              }
              if (pos!=-1){
                 out.println(s[pos]);
              }
              else out.println(am+" not found!!!");
              break;
           case 0:System.exit(1);
           default:out.println("wrong input!!");
        }
     }
  }
}
```

Η κλάση TestStudentArrayList (1/2)

```
import static java.lang.System.out;
import java.util.ArrayList;
import java.util.Scanner;
public class TestStudentArrayList {
  public static void main(String[] args){
    Scanner in = new Scanner(System.in);
    ArrayList<Student> s = new ArrayList<>();
    int choice,am;
    String name;
    while(true){
      out.println("1.insert student");
      out.println("2.print students");
      out.println("3.search by am");
      out.println("0. exit");
      out.print("choice: ");
      choice = in.nextInt();
      switch(choice){
        case 1://insert student
          out.println("am: ");
          am=in.nextInt();in.nextLine();
          out.println("name: ");
          name= in.nextLine();
          Student temp = new Student(am, name);
          s.add(temp);
          //insert courses
          while(true){
            out.print("courseID(0 to end): ");
            int id=in.nextInt();
            if (id==0) break;
            out.print("grade: ");
            int bathmos=in.nextInt();
            Course c = new Course(id, bathmos);

            //s.get(s.size()-1).courses.add(c);
            s.get(s.size()-1).addCourse(c);
          }
          break;
```

Η κλάση TestStudentArrayList (2/2)

```
case  2:
    for(int i=0; i<s.size(); i++){
      out.println(s.get(i));
      for(int j=0; j<s.get(i).courses.size(); j++){
          out.println(s.get (i).courses.get(j));
      }
    }
    break;
case 3:
    out.println("am: ");
    am=in.nextInt();
    int pos=-1;
    for(int i=0; i<s.size(); i++){
      if(s.get(i).am == am){
          pos=i;
          break;
      }
    }
    if (pos!=-1){
      out.println(s.get(pos));
    }
    else out.println(am+" not found!!!");
    break;
          case 0:System.exit(1);
          default:out.println("wrong input!!");
        }
      }
    }
}
```

Γεννήτρια τυχαίων αριθμών

- Ανατρέξτε στην τεκμηρίωση της Java για να εξοικειωθείτε με τη δόμηση σε πακέτα και κλάσεις.
- Η μέθοδος random επιστρέφει πραγματικούς αριθμούς διπλής ακρίβειας (**double**) τυχαίους αριθμούς στο διάστημα [0.0, 1.0).
- Εάν χρειαζόμαστε τυχαίους ακέραιους αριθμούς στο διάστημα [x, y], τότε απλώνουμε το εύρος πολλαπλασιάζοντας και μεταθέτουμε προσθέτοντας ως εξής:
  - x + (**int**) ((y-x+1) * Math.random()).
- Για παράδειγμα αν θέλουμε τυχαίους αριθμούς από το 1 ως το 6 (ζάρι), η παράσταση διαμορφώνεται
  - `1 + (int) ((6-1+1) * Math.random())` ή
  - `1 + (int) (6 * Math.random())`

Γεννήτρια τυχαίων αριθμών με τα util.random αντικείμενα

- Το κλάση java.util.Random χρησιμοποιείται για να δημιουργήσει ένα ρεύμα ψευδοτυχαίων αριθμών.
- Ακολουθούν μερικά βασικά στοιχεία για την κλάση random
  - Η τάξη χρησιμοποιεί ένα σπόρο 48 - bit , ο οποίος έχει τροποποιηθεί χρησιμοποιώντας μια γραμμική συμβατική φόρμουλα .
  - Οι αλγόριθμοι που εφαρμόζονται από την κλάση Random χρησιμοποιούν μία προστατευόμενη μέθοδο που σε κάθε κάλεσμά της μπορεί να παρέχει μέχρι και 32 ψευδοτυχαία παραγόμενα bits

Μέθοδοι των αντικειμένων του util.random

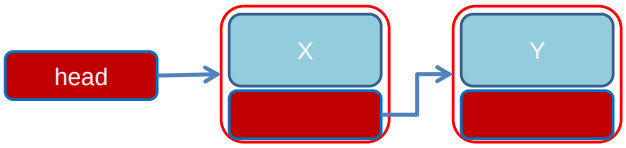
- Boolean nextBoolean
- double nextDouble
- float nextFloat
- double nextGaussian (κανονικά κατανεμημένη τιμή μεταξύ 0.0 και 1.0)
- int nextInt
- long nextLong

Σύνθεση αντικειμένων

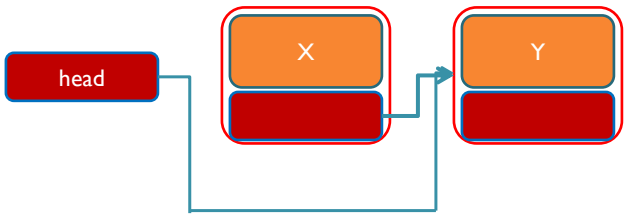
- Ορίζουμε κλάσεις για να ορίσουμε τύπους δεδομένων τους οποίους χρειαζόμαστε
  - Π.χ., ο τύπος δεδομένων Date για να μπορούμε να χειριζόμαστε μια ημερομηνία.
  - Π.χ., ο τύπος δεδομένων Examination κρατάει πληροφορία για μία εξέταση
- Τους τύπους δεδομένων που ορίζουμε τους χρησιμοποιούμε για να δημιουργήσουμε μεταβλητές (αντικείμενα).
- Τα αντικείμενα μπορεί να είναι πεδία άλλων κλάσεων
  - Π.χ., η κλάση Examination έχει ένα πεδίο τύπου Date
- Μία κλάση χρησιμοποιεί αντικείμενα άλλων κλάσεων και έτσι συνθέτουμε πιο περίπλοκους τύπους δεδομένων.

Παράδειγμα

- Υλοποιήστε το **Stack** που φτιάξαμε στα προηγούμενα μαθήματα ώστε να μην έχει περιορισμό στο μέγεθος (**capacity**).
- Βασική ιδέα:
  - Δημιουργούμε στοιχεία της στοίβας και τα συνδέουμε το ένα να δείχνει στο άλλο.
  - Χρειάζεται να ξέρουμε και την κορυφή της στοίβας

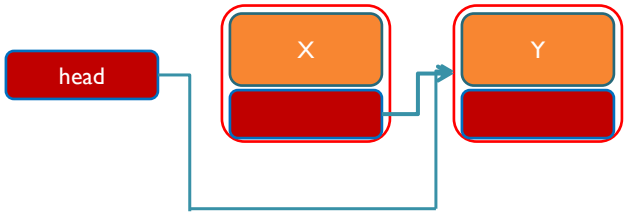


Στοίβα



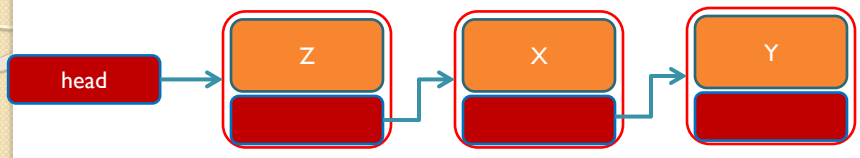
- **Pop()**: Αφαιρεί το στοιχείο στην κορυφή της στοίβας και επιστρέφει την τιμή του (X στο παράδειγμα μας)

Στοίβα



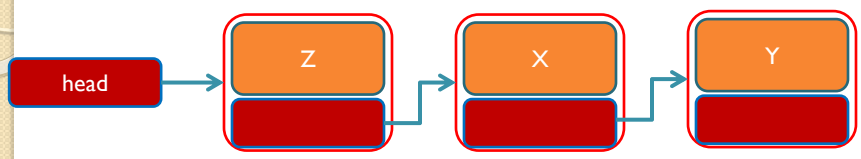
- **Pop()**: Αφαιρεί το στοιχείο στην κορυφή της στοίβας και επιστρέφει την τιμή του (X στο παράδειγμα μας)

Στοίβα



- **Push(Z)**: Προσθέτει την τιμή Z στην κορυφή της στοίβας

Στοίβα - Υλοποιηση



- Θα ορίσουμε StackElement μια κλάση που κρατάει το κάθε στοιχείο της στοίβας.

```
class StackElement
{
        private int value;
        private StackElement next = null;

        public StackElement(int value){
                this.value = value;
        }

        public int getValue(){
                return value;
        }

        public StackElement getNext(){
                return next;
        }

        public void setNext(StackElement element){
                next = element;
        }

}
```

```
class StackElement
{
        private int value;
        private StackElement next = null;

        public StackElement(int value){
                this.value = value;
        }

        public int getValue(){
                return value;
        }

        public StackElement getNext(){
                return next;
        }

        public void setNext(StackElement element){
                next = element;
        }

}
```

Το επόμενο στοιχείο

Επιστρέφει αντικείμενο

```
class Stack
{
        private StackElement head;
        private int size = 0;

        public int pop(){
                if (size == 0){ // head == null
                        System.out.println("Pop from empty stack");
                        System.exit(-1);
                }
                int value = head.getValue();
                head = head.getNext();
                size --;
                return value;
        }

        public void push(int value){
                StackElement element = new StackElement(value);
                element.setNext(head);
                head = element;
                size ++;
        }

}
```

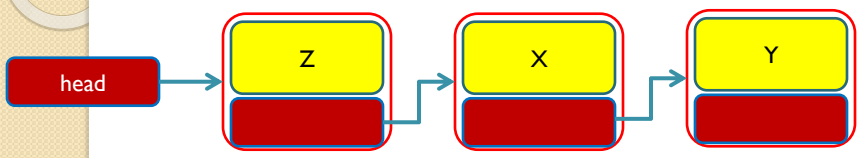
Το πρώτο στοιχείο της στοίβας μας φτάνει για τα βρούμε όλα

Σταματάει την εκτέλεση του προγράμματος

Τα αντικείμενα τύπου StackElement δημιουργούνται μέσα στην Stack.

```
class StackExample
{
        public static void main(String[] args){
                Stack s = new Stack();
                s.push(3);
                s.push(2);
                s.push(1);
                System.out.println(s.pop());
                System.out.println(s.pop());
                System.out.println(s.pop());
                System.out.println(s.pop());
        }
}
```

Στοίβα - Υλοποίηση



- Τα X,Y,Z μπορεί να είναι δεδομένα οποιουδήποτε τύπου ή κλάσης. Π.χ. αντί για ακέραιους θα μπορούσαμε να έχουμε αντικείμενα τύπου Person.

```
class Person
{
        private String name;
        private int number;

        public Person(String name, int num){
                this.name = name;
                this.number = num;
        }

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

```
class PersonStackElement
{
        private Person value;
        private PersonStackElement next;

        public PersonStackElement(Person val){
                value = val;
        }

        public void setNext(PersonStackElement element){
                next = element;
        }

        public PersonStackElement getNext(){
                return next;
        }

        public Person getValue(){
                return value;
        }
}
```

Ο constructor παίρνει σαν όρισμα το αντικείμενο που έχει ήδη δημιουργηθεί

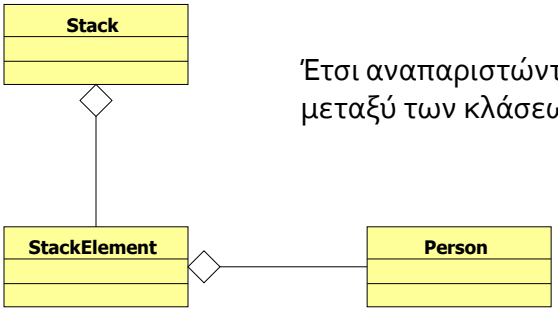
Το αντικείμενο το χειριζόμαστε σαν μια οποιαδήποτε μεταβλητή

Σχέσεις μεταξύ κλάσεων

- Στο παράδειγμα με τη στοίβα έχουμε τρείς διαφορετικές κλάσεις (Person, StackElement, Stack) τις οποίες συσχετίζονται μεταξύ τους με διαφορετικούς τρόπους.
- Μπορεί να υπάρχουν πολλές διαφορετικές σχέσεις μεταξύ κλάσεων.
  - Στην περίπτωση μας, η μία κλάση ορίζεται χρησιμοποιώντας αντικείμενα της άλλης
- Αυτού του είδους τη σχέση την λέμε σχέση σύνθεσης
  - Μερικές φορές την ξεχωρίζουμε σε σχέση σύνθεσης (composition) και συνάθροισης (aggregation).

Η UML γλώσσα

- Η UML (Unified Modeling Language) είναι μια γλώσσα για να περιγράφουμε και να καταλαβαίνουμε τον κώδικα μας.
- Τα UML διαγράμματα παρέχουν μια οπτικοποίηση των σχέσεων μεταξύ των κλάσεων.



Έτσι αναπαριστώνται οι σχέσεις μεταξύ των κλάσεων

Σχέσεις κλάσεων

- Όταν έχουμε κλάσεις που έχουν αντικείμενα άλλων κλάσεων ένα θέμα που προκύπτει είναι πότε και πού θα γίνεται η δημιουργία των αντικειμένων και πότε η καταστροφή τους
  - Πιο σημαντικό σε γλώσσες που δεν έχουν garbage collector.
- Π.χ., τα αντικείμενα τύπου StackElement στο προηγούμενο παράδειγμα δημιουργούνται μέσα στην κλάση Stack, και καταστρέφονται μέσα στην Stack, ή αν η Stack καταστραφεί.
- Τα αντικείμενα τύπου Person που χρησιμοποιούνται στην StackElement δημιουργούνται εκτός της κλάσης και μπορεί να υπάρχουν αφού καταστραφεί η κλάση.
- Συχνά οι σχέσεις του δεύτερου τύπου λέγονται σχέσεις συνάθροισης, ενώ του πρώτου σχέσεις σύνθεσης.

Σχέση συνάθροισης – Aggregation

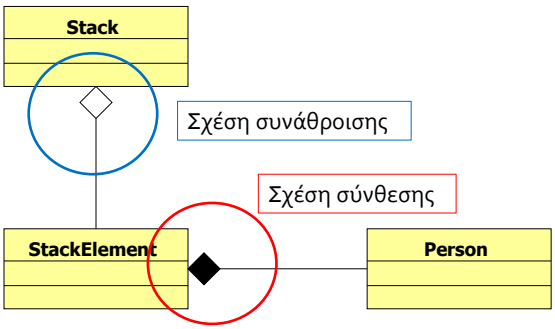
- Η κλάση X έχει σχέση συνάθροισης με την κλάση Y, αν αντικείμενο/α της κλάσης Y ανήκουν στο αντικείμενο της κλάσης X.
  - Τα αντικείμενα της κλάσης Y έχουν υπόσταση και εκτός της κλάσης X.
    - Όταν καταστρέφεται ένα αντικείμενο της κλάσης X δεν καταστρέφονται απαραίτητα και τα αντικείμενα της κλάσης Y.
- Παραδείγματα:
  - Σε έναν άνθρωπο μπορεί να ανήκει ένα αυτοκίνητο, ρούχα, κλπ.
  - Ένα κτήριο μπορεί να έχει μέσα ανθρώπους, έπιπλα, κλπ.
- Στην περίπτωση μας η κλάση StackElement έχει σχέση συνάθροισης με την κλάση Person.

Σχέση σύνθεσης – Composition

- Η κλάση X έχει σχέση σύνθεσης με την κλάση Y, αν το αντικείμενο της κλάσης X αποτελείται από αντικείμενα της κλάσης Y.
  - Τα αντικείμενα της κλάσης Y δεν υπάρχουν εκτός της κλάσης X.
    - Η κλάση X δημιουργεί τα αντικείμενα της κλάσης Y, και καταστρέφονται όταν καταστρέφεται το αντικείμενο της κλάσης X.
- Παραδείγματα:
  - Ένας άνθρωπος αποτελείται από μέρη του σώματος: κεφάλι, πόδια, χέρια κλπ.
  - Ένα κτήριο αποτελείται από τοίχους, δωμάτια, πόρτες, κλπ.
- Στην περίπτωση μας η κλάση Stack έχει σχέση σύνθεσης με την κλάση StackElement.

UML διαγράμματα

- Για να ξεχωρίζουν μεταξύ τους (κάποιες φορές) αναπαριστώνται διαφορετικά στα UML διαγράμματα.



Aggregation and Composition

- Το αν θα είναι μια σχέση, σχέση συνάθροισης ή σύνθεσης εξαρτάται κατά πολύ και από την υλοποίηση μας και τον σχεδιασμό.
  - Π.χ., σε ένα διαφορετικό πρόγραμμα μπορεί να επαναχρησιμοποιούμε το StackElement.
  - Π.χ., σε μία διαφορετική εφαρμογή, τα ανθρώπινα όργανα υπάρχουν και χωρίς τον άνθρωπο.

Προσοχή!

- Ο διαχωρισμός σε σχέσεις συνάθροισης και σύνθεσης είναι ως ένα βαθμό ένας φορμαλισμός.
  - Μην «κολλήσετε» προσπαθώντας να ορίσετε την σχέση.
  - Το σημαντικό είναι όταν δημιουργείτε το πρόγραμμα σας να σκεφτείτε ποιες κλάσεις χρειάζονται τα αντικείμενα που δημιουργούνται και πότε πρέπει να δημιουργηθούν μέσα στον κώδικα.
  - Δεν υπάρχει χρυσός κανόνας. Γενικά το πώς θα σχεδιαστεί το πρόγραμμα είναι κάτι που μπορεί να γίνει με πολλούς τρόπους συνήθως. Διαλέξτε αυτόν που θα κάνει το πρόγραμμα πιο απλό, ευανάγνωστο, εύκολο να επεκταθεί, να ξαναχρησιμοποιηθεί και να διατηρηθεί.