Αντικειμενοστρεφής Προγραμματισμός
Διάλεξη – 7 : ΣΥΝΘΕΤΕΣ ΚΛΑΣΕΙΣ ΚΑΙ ΜΕΘΟΔΟΙ

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

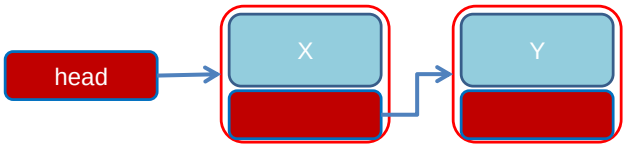
- Boolean nextBoolean
- double nextDouble
- float nextFloat
- double nextGaussian (κανονικά κατανεμημένη τιμή μεταξύ 0.0 και 1.0)
- int nextInt
- long nextLong

Σύνθεση αντικειμένων

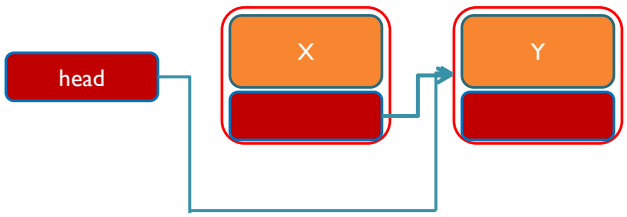
- Ορίζουμε κλάσεις για να ορίσουμε τύπους δεδομένων τους οποίους χρειαζόμαστε
  - Π.χ., ο τύπος δεδομένων Date για να μπορούμε να χειριζόμαστε μια ημερομηνία.
  - Π.χ., ο τύπος δεδομένων Examination κρατάει πληροφορία για μία εξέταση
- Τους τύπους δεδομένων που ορίζουμε τους χρησιμοποιούμε για να δημιουργήσουμε 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

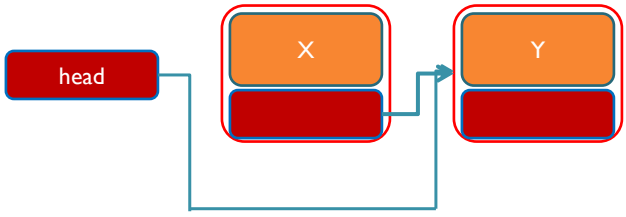


Στοίβα



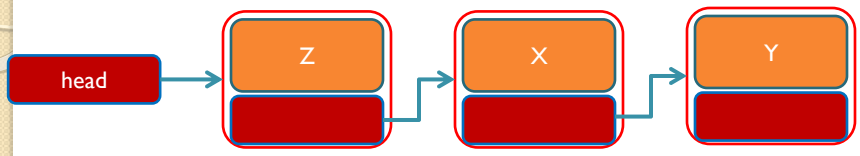
- **Pop()**: Αφαιρεί το στοιχείο στην κορυφή της στοίβας και επιστρέφει την τιμή του (X στο παράδειγμα μας)

Στοίβα



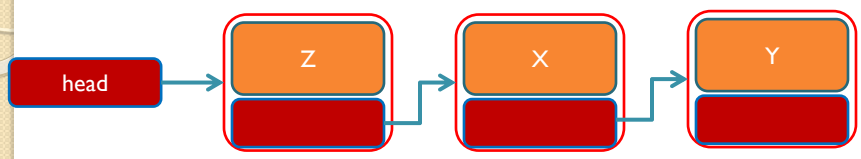
- **Pop()**: Αφαιρεί το στοιχείο στην κορυφή της στοίβας και επιστρέφει την τιμή του (X στο παράδειγμα μας)

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 ++;
        }

}
```

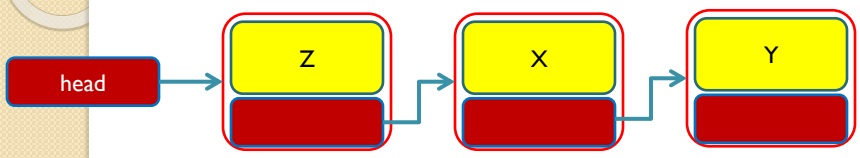
Το πρώτο στοιχείο της στοίβας μας φτάνει για τα βρούμε όλα

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

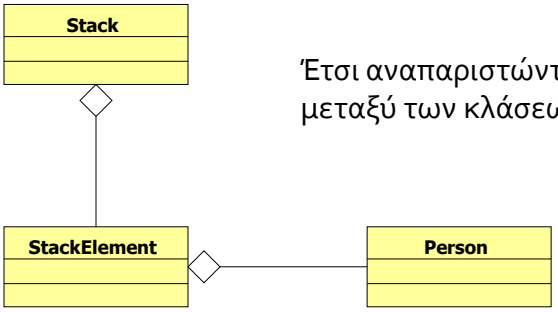
Το αντικείμενο το χειριζόμαστε σαν μια οποιαδήποτε μεταβλητή

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

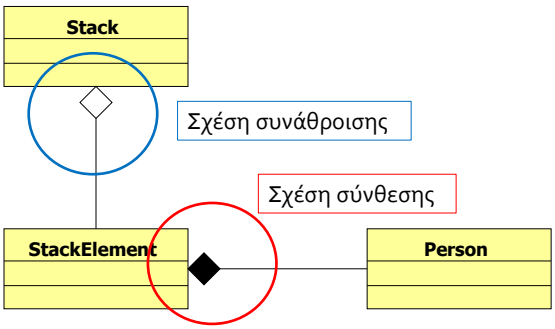
- Η κλάση X έχει σχέση συνάθροισης με την κλάση Y, αν αντικείμενο/α της κλάσης Y ανήκουν στο αντικείμενο της κλάσης X.
  - Τα αντικείμενα της κλάσης Y έχουν υπόσταση και εκτός της κλάσης X.
    - Όταν καταστρέφεται ένα αντικείμενο της κλάσης X δεν καταστρέφονται απαραίτητα και τα αντικείμενα της κλάσης 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.