Αντικειμενοστρεφής Προγραμματισμός

(https://people.iee.ihu.gr/~adamidis/OOP.html)

Παναγιώτης Αδαμίδης
adamidis@ihu.gr

Διάλεξη 1η

Εισαγωγή στον Αντικειμενοστρεφή Προγραμματισμό.
Εισαγωγή στη Java.

Αρχές Αντικειμενοστραφούς Προγραμματισμού

- Ενθυλάκωση (Encapsulation)
- Αφαίρεση (Abstraction)
- Κληρονομικότητα (Inheritance)
- Πολυμορφισμός (Polymorphism)

Ενθυλάκωση

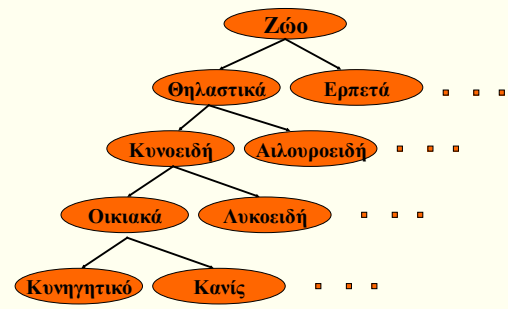
- Βασική έννοια: Κλάση (Class)
- Υλοποίηση: Αντικείμενο (Object) - στιγμιότυπο (instance) μίας κλάσης
- Χαρακτηριστικά: Μεταβλητές (instance variables)
- Συμπεριφορά/Λειτουργία: Μέθοδοι (methods)
- Πρόσβαση: default, private, public, protected

Ενθυλάκωση - Παράδειγμα

Αφαίρεση

- Φυσική απόρροια της ενθυλάκωσης
- Η αφαίρεση επιτρέπει να δηλώσουμε ποιες θα είναι οι λειτουργίες αδιαφορώντας για την υλοποίησή τους.
- Η εφαρμογή της αφαίρεσης σημαίνει ότι κάθε αντικείμενο θα πρέπει να εμφανίζει **μόνο** ένα μηχανισμό χρήσης υψηλού επιπέδου.
- Αυτός ο μηχανισμός θα πρέπει να κρύβει τις λεπτομέρειες υλοποίησης και να αποκαλύπτει μόνο λειτουργίες σχετικές με άλλα αντικείμενα.
- Θα πρέπει να είναι εύκολος στη χρήση και να αλλάζει σπάνια.

Κληρονομικότητα



Κληρονομικότητα

- Η κληρονομικότητα εκφράζει σχέσεις "is-a" μεταξύ δύο κλάσεων (→ αντικειμένων).
- Με χρήση κληρονομικότητας στις παραγόμενες κλάσεις (υποκλάσεις) επαναχρησιμοποιείται κώδικας των γονικών (υπερκλάσεις).
- Στιγμιότυπα των υποκλάσεων μπορούν να χρησιμοποιηθούν όπου απαιτούνται στιγμιότυπα των υπερκλάσεων.
- Στην Java, η έννοια "is-a" βασίζεται στην κληρονομικότητα κλάσεων (using extends) or στην υλοποίηση διασυνδέσεων (using implements).
- Οι υποκλάσεις εξειδικεύουν τη συμπεριφορά των υπερκλάσεών τους.

Πολυμορφισμός

- Συναρτησιακός / διαδικαστικός προγραμματισμός: διαφορετικές συναρτήσεις με διαφορετικό όνομα για να κάνεις διαφορετικά πράγματα
- Πολυμορφισμός: ένα αντικείμενο, πολλές μορφές. Μία μέθοδος μπορεί να έχει πολλές υλοποιήσεις, ανάλογα με τον τύπο του αντικειμένου που την καλεί.
- Στατικός (overloading) - Δυναμικός (overriding)

Ανάπτυξη εφαρμογής

- Εισαγωγή κώδικα Java με κάποιο συντάκτη κειμένου
- Μεταγλώττιση και δημιουργία του αρχείου κώδικα byte (byte-code)
- Κλήση του διερμηνέα (interpreter) για την εκτέλεση του αρχείου κώδικα byte
- Διόρθωση λαθών, αν χρειάζεται

Ανάπτυξη εφαρμογής : παράδειγμα

- Εισαγωγή κώδικα:

```
class HelloWorld {
    public static void main (String args[]) {
        System.out.println ("Hello World!");
    }
}
```

- Μεταγλώτιση: javac HelloWorld.java -->HelloWorld.class
- Εκτέλεση: java HelloWorld

Ανάπτυξη applet

- Εισαγωγή κώδικα Java applet με κάποιο συντάκτη κειμένου.
- Μεταγλώττιση και δημιουργία του αρχείου κώδικα byte (byte-code).
- Δημιουργία αρχείου HTML το οποίο περιέχει το applet.
- Εκτέλεση του αρχείου κώδικα byte μέσω κλήσης του αρχείου HTML από ένα πρόγραμμα πλοήγησης ή με το **appletviewer**.
- Διόρθωση λαθών, αν χρειάζεται

Ανάπτυξη applet : παράδειγμα

- Εισαγωγή κώδικα:

```
import java.applet.Applet;
import java.awt.Graphics;
class HelloWorld extends Applet{
    public void paint (Graphics g) {
        g.drawString ("Hello World!", 50, 40);
    }
}
```

- Μεταγλώτιση: javac HelloWorld.java -->HelloWorld.class

Ανάπτυξη applet : παράδειγμα (2)

- Εκτέλεση:
  - Δημιουργία αρχείου HTML: hiworld.html

```
<HTML>
<HEAD><TITLE> Hello World</TITLE></HEAD>
<BODY>
Αυτό είναι το πρώτο μου applet: <p>
<APPLET CODE="HelloWorld.class" WIDTH=300 height=200>
</APPLET>
</BODY>
</HTML>
```

- appletviewer hiworld.html

Στοιχεία της γλώσσας: Σύνταξη

- Όλες οι εντολές τελειώνουν με «;»
- Οι λευκοί χαρακτήρες (white characters: tabs, spaces, and newlines) αγνοούνται

```
x=a+b;
x = a + b;
x          =a
+     b    ;
```

- Μερικές φορές είναι σημαντικοί

```
OurWeight = MyWeight + YourWeight;
Our  Weight = My  Weight + Your  Weight;
```

Στοιχεία της γλώσσας

- Σχόλια
  - //, /* … */
- Αναγνωριστικά (Identifiers)
  - γράμματα, αριθμοί, _, $
- Τύποι Δεδομένων (Data types)
  - byte, int, short, long, float, double, char, boolean,
- Εκφράσεις (Expressions)
  - a = b + c;

Στοιχεία της γλώσσας: Αναγνωριστικά (Identifiers)

- Αποτελούν το όνομα μεταβλητών, συναρτήσεων, δεδομένων κλπ.
- Συνδυασμός αλφαριθμητικών χαρακτήρων και του underline (_). Ο πρώτος πρέπει να είναι αλφαβητικός ή underline.
- Δύο βασικοί κανόνες ονοματολογίας:
  - Οι κεφαλαίοι χαρακτήρες είναι διαφορετικοί από τούς πεζούς. Π.χ. Οι "identifiers" index και INDEX είναι διάφοροι μεταξύ τους και επίσης διάφοροι του InDeX. Και οι τρείς αναφέρονται σε διαφορετικές μεταβλητές
  - Σύμφωνα με το ANSI-C standard, το όνομα μπορεί να έχει μήκος μέχρι 31 χαρακτήρες. Οι υπόλοιποι αγνούνται.

Σύνθετες εντολές - Blocks

- Αρχίζουν με αριστερό άγκιστρο { και τελειώνουν με δεξί }
- Δεν τελειώνουν με semicolon (;)

```
{
  temp = a;
  a = b;
  b = temp;
}
```

Βασικοί τύποι δεδομένων

- boolean, 8 bits, true/false
- char, 16 bits, '\u0000' έως '\uFFFF' (ISO Unicode)
- byte, 8 bits, -128 έως 127
- short, 16 bits, -32768 έως 32767
- int, 32 bits, -2.147.483.648 έως 2.1147.483.647
- long, 64 bits, -9.223.372.036.854.775.808 έως 9.223.372.036.854.775.807
- float, 32 bits, -3.40292347E+38 έως 3.40292347E+38
- double, 64 bits, -1.79769313486231570E+308 έως 1.79769313486231570E+308

Σταθερές

- Ακέραιες π.χ. 100
  - `0144` base8, `0x64` base16
  - Δεν μπορεί να περιέχει κενό, κόμμα, τελεία
  - Μπορεί να προηγείται πρόσημο
- Κινητής υποδιαστολής (floating point) π.χ. 3.14, 10., .01, 123e4
  - Δεν μπορεί να περιέχει κενό, κόμμα.
  - Πρέπει να περιέχει τελεία
  - Μπορεί να προηγείται πρόσημο
- Χαρακτήρων π.χ. 'A', '.', '%'

  Escape sequences: \a (bell), \b (backspace), \f (formfeed), \n (newline), \r (carriage return), \t (horiz. tab), \v (vert. tab), \', \", \\, \?
- Strings - σύνολο χαρακτήρων
  π.χ. "apple", "Τρίτη"

Αντικειμενοστρεφής Προγραμματισμός

Τελεστές - Μεταβλητές στη Java

Τελεστές (Operators)

- Αριθμητικοί
  - +, +=, -, -=, *, *=, /, /=, ++, --, %, %=
- Σχεσιακοί
  - ==, >, >=, <, <=, !=,
- Λογικοί
  - |, |=, ^, ^=, &, &=, !, &&, ||,
- Εκχώρησης: =

Δήλωση μεταβλητών

- **type identifier [= value], [,identifier [=value]...];**

  ↓

  byte, short, int, long, char, float, double, boolean ή όνομα τάξης ή διασύνδεσης(interface)
- Παραδείγματα:
  - int a, b, c;
  - int d=3, e ,f=5;
  - byte z=22;
  - double pi=3.14159;
  - char x='x';
  - int float_var=3;
  - float int_var=2.3;

Προκαθορισμένες τιμές βασικών τύπων-μελών

Βασικός τύπος	Τιμή
`boolean`	`false`
`char`	`'\u0000'` (null)
`byte`	`(byte)0`
`short`	`(short)0`
`int`	`0`
`long`	`0L`
`float`	`0.0f`
`double`	`0.0d`

Μετατροπές τύπων (type casting)

- Αυτόματα σε εκφράσεις και αναθέσεις:
  - char/short ---> int
  - float --> double
- Σε πράξεις προάγονται στον μεγαλύτερο τύπο:
  - 7.0 / 2 --> 7.0 / 2.0 --> 3.5
  - 2L + 3.4L --> 2.0L +3.4L --> 5.4L
- Σε ανάθεση τιμής σε μεταβλητή μικρότερης χωρητικότητας, χάνεται μέρος των δεδομένων
- Μετατροπές μπορούν να γίνουν άμεσα:

```
float a = 5 / 2;          /* a = 2.0 */
float a = 5.0 / 2;        /* a = 2.5 */
float a = (float)5 / 2;   /* a = 2.5 */
float a = (float)(5 / 2); /* a = 2.0 */
```

Τελεστές : Παράδειγμα

```
public class Increment Decrement {
    public static void main (String[] args){
        char c = 'R';  byte j = 127;
        short k = 32767;
        System.out.println("c= " + c);      → R
        ++c;
        System.out.println("c= " + c);      → S
        System.out.println("j= " + j);      → 127
        --j;
        System.out.println("j= " + j);      → 126
        ++j;                                j → 127
        ++j;                                j → -128
        System.out.println("k= " + k);      → 32767
        k -= 4;                             k → 32763
        k += 5;                             k → -32768
    }
}
```

Απλό παράδειγμα

```
public class Exercise1{
   static float ComputePay(float hours, float rate){
       float pay;
       pay=hours * rate;
       return pay;
   }
   public static void main (String argv[]){
       float hours;
       float payrate;
       hours=40f;
       payrate=12.50f;
       System.out.println("The pay is: " +
                          ComputePay(hours, payrate));
   }
}
```

Αντικειμενοστρεφής Προγραμματισμός

Κλάσεις,
Αντικείμενα,
Μέθοδοι

Δημιουργία αντικειμένου

- Δεδομένης μίας κλάσης με όνομα **`EnasTypos`**:

```
class EnasTypos {/* σώμα κλάσης */}
```

- Η δημιουργία ενός αντικειμένου γίνεται με την χρήση του **new**:

```
EnasTypos a = new EnasTypos();
```

Declaration | Instantiation | Initialization

Πρόσβαση σε μέλη

```
class Box {
    float width;
    float length;
    float height;
}
```

- Δημιουργία αντικειμένου:

```
Box myBox = new Box();
```

- Πρόσβαση:
  - `myBox.length = 3;`
  - `myBox.width = 4.0f;`
  - `myBox.height = 2.5f;`

Μέθοδοι

- Ορίζουν την συμπεριφορά/λειτουργία των αντικειμένων που θα δημιουργηθούν.
- Είναι ένα ανεξάρτητο τμήμα κώδικα (με συγκεκριμένο όνομα) το οποίο εκτελεί μία συγκεκριμένη εργασία και επιστρέφει, προαιρετικά, μία τιμή.
- Η εκτέλεση μιας εφαρμογής Java αρχίζει πάντα με την (αυτόματη) κλήση της μεθόδου **`main`**.
- Στην **`main`** μπορεί να καλούνται άλλες μέθοδοι οι οποίες μπορούν να καλούν άλλες κ.ο.κ.

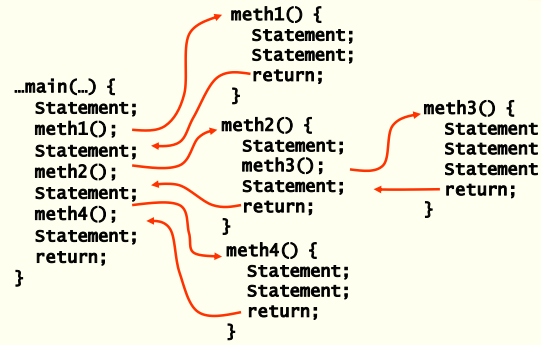
Ορισμός μεθόδου

- Βασικά μέρη: όνομα, τύπος επιστροφής, λίστα παραμέτρων και σώμα μεθόδου:

```
<τύπος> <όνομα>(<παράμετροι>) {
  <σώμα_μεθόδου>
}
```

- Επιπλέον προαιρετικά μέρη (ισχύουν και για μεταβλητές): Προσδιοριστές πρόσβασης (**private, public, protected**, πακέτο/default), **final, static**

Κλήση Μεθόδων



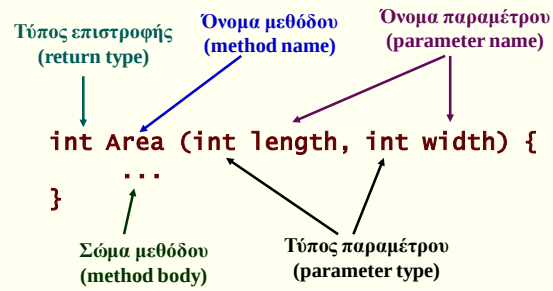
Ορισμός Μεθόδου

- Ο ορισμός της μεθόδου λέει στον μεταγλωττιστή τι ακριβώς κάνει η μέθοδος.
- Ο ορισμός μιας μεθόδου αποτελείται από την δήλωση και το σώμα της μεθόδου.
- Η δήλωση παρέχει στον μεταγλωττιστή: τύπος επιστροφής, όνομα, λίστα ορισμάτων.

```
<τύπος_επιστρ.> <όνομα>(<λίστα_ορισμάτων>){
                <σώμα_μεθόδου>
}
<λίστα_ορισμάτων>::= <τύπος_ορ> [<αναγνωριστ>]
      [, <τύπος_ορ> [<αναγνωριστ>] [, ...]]
```

- Η ύπαρξη των ονομάτων των μεταβλητών που αντιστοιχούν στα ορίσματα είναι υποχρεωτική (δεν είναι υποχρεωτικό να είναι τα ίδια όταν κληθεί η μέθοδος).

Στοιχεία δήλωσης/ορισμού



Ανάθεση τιμών αντικειμένων - Λ

```
class Number { int i; }
public class Assignment {
   public static void main(String[] args) {
      Number n1 = new Number();
      Number n2 = new Number();
      n1.i = 9;
      n2.i = 47;
      System.out.println("1: n1.i: " + n1.i +
                         ", n2.i: " + n2.i);
      n1 = n2;
      System.out.println("2: n1.i: " + n1.i +
                          ", n2.i: " + n2.i);
      n1.i = 27;
      System.out.println("3: n1.i: " + n1.i +
                          ", n2.i: " + n2.i);
   }
}
```

```
Output - Assignment_wrong (run)
run:
1: n1.i: 9, n2.i: 47
2: n1.i: 47, n2.i: 47
3: n1.i: 27, n2.i: 27
```

Ανάθεση τιμών αντικειμένων – Σ(1)

```
class Number {
    private int i;
    public int getI(){
        return i;
    }
    public void setI(int n){ i=n; }
    Number (){}
    Number (int n){
        i = n;
    }
    public String toString(){
        String s = new String();
        s += i;
        return s;
    }
}
```

Ανάθεση τιμών αντικειμένων – Σ(2)

```
public class Assignment {
   public static void main(String[] args) {
      Number n1 = new Number();
      Number n2 = new Number();
      n1.setI(9);
      n2.setI(47);
      System.out.println("1: n1: " + n1 + ", n2: " + n2);
      n1 = n2;
      System.out.println("2: n1: " + n1 + ", n2: " + n2);
      n1.setI(27);
      System.out.println("1: n1: " + n1 + ", n2: " + n2);
      System.out.println("1: n1: " + n1 + ", n2: " + n2);
   }
}
```

```
run:
1: n1: Value of number is 9, n2: Value of number is 47
2: n1: Value of number is 47, n2: Value of number is 47
1: n1: Value of number is 27, n2: Value of number is 27
1: n1: Value of number is 27, n2: Value of number is 27
BUILD SUCCESSFUL (total time: 0 seconds)
```

Παράδειγμα: Παράμετροι βασικοι τύποι

```
class Letter { char c; }
public class PassObject {
   static void f(char y) {
       y = 'z';
}
   public static void main(String[] args) {
       Letter x = new Letter();
       x.c = 'a';
       System.out.println("1: x.c: "+x.c);
       f(x.c);
       System.out.println(”2: x.c: "+x.c);
  }
}
```

Παράδειγμα: Αντικείμενα παράμετροι

```
class Letter { char c; }
public class PassObject {
   static void f(Letter y) {
       y.c = 'z';
   }
   public static void main(String[] args) {
       Letter x = new Letter();
       x.c = 'a';
       System.out.println("1: x.c: "+x.c);
       f(x);
       System.out.println("2: x.c: "+x.c);
  }
}
```

Παράδειγμα: CD Collection

```
class CDCollection {
    private int numCDs;
    private double valueCDs;
    public CDCollection(int initial_num, double initial_val){
            numCDs=initial_num;
            valueCDs=initial_val;
    }
    public void addCDs(int numCDs, double value){
             this.numCDs += numCDs;
             valueCDs=valueCDs+value;
    }
```

Παράδειγμα: CD Collection (2)

```
    public void print() {
        System.out.println("* * * * * * * * * " );
        System.out.println("Πλήθος CD:" + numCDs );
        System.out.println("Αξία Συλλογής : €" + valueCDs);
        System.out.println("Μέση τιμή ανά CD:" +
                                        averageCost());
    }
    private double averageCost(){
        return  valueCDs / numCDs;
    }
} //class CDCollection
```

Παράδειγμα: CD Collection (3)

```
class Tunes{
    public static void main (String[] args){
        CDCollection myCDs = new CDCollection(5, 59.65);
        // Initial Collection
        myCDs.print();
        // Increase Collection Day 1
        myCDs.addCDs(1,17.99);
        myCDs.addCDs(3,39.72);
        myCDs.print();
        // Increase Collection Day 2
        myCDs.addCDs(2,20.82);
        myCDs.addCDs(4,54.57);
        myCDs.print();   } // main
} // Tunes
```

Αντικειμενοστρεφής Προγραμματισμός

Υπερφόρτωση μεθόδων

Υπερφόρτωση Μεθόδου

- **Υπερφόρτωση** είναι η ύπαρξη στην ίδια κλάση περισσοτέρων της μίας, μεθόδων με το ίδιο όνομα.
- Η διαφοροποίηση μεταξύ τους γίνεται από τις διαφορές τους στα ορίσματά τους (πλήθος, τύπος και σειρά).
- Οι μέθοδοι μπορούν να έχουν διαφορετικό τύπο επιστροφής.

...καλύτερη CD Collection (1)

```
class CDCollection {
    private int numCDs;
    private double valueCDs;

    public CDCollection(){}
    public CDCollection(int initial_num){
        numCDs=initial_num;
    }
    public CDCollection(double initial_val){
        valueCDs=initial_val;
    }
    public CDCollection(int initial_num, double initial_val){
            numCDs=initial_num;
            valueCDs=initial_val;
    }
```

...καλύτερη CD Collection (2)

```
public int getNumCDs(){
        return numCDs;
}
public void setNumCDs(int n){
        this.numCDs = n;
}
public int getNumCDs(){
        return numCDs;
}
public void setNumCDs(int n){
        this.numCDs = n;
}
```

...καλύτερη CD Collection (3)

```
    public void addCDs(int numCDs, double value){
        this.numCDs += numCDs;
        valueCDs=valueCDs+value;
    }
    public String toString() {
        String s = new String ("* * * * * * * * * *" );
        s += "\nNumber of CDs:" + numCDs ;
        s += "\nCollection Value : €" + valueCDs);
        s += "\nAverage CD price:" + averageCost());
        return s;
    }
    private double averageCost(){
        return  valueCDs / numCDs;
    }
} //class CDCollection
```

Υπερφόρτωση δομητών και μεθόδων (1)

```
import java.util.*;
class Tree {
    int height;
    Tree() {
        prt("Planting a seedling");
        height = 0;
    }
    Tree(int i) {
        prt("Creating new Tree that is "+i+
                                "meters tall");
        height = i;
    }
    static void prt(String s) {
        System.out.println(s);
    }
```


Υπερφόρτωση δομητών και μεθόδων (2)

```
    void info() {
       prt("Tree is " + height + " meters tall");
    }
    void info(String s) {
       prt(s + ": Tree is " + height + " meters tall");
    }
}
public class Overloading {
  public static void main(String[] args) {
    for(int i = 0; i < 5; i++) {
      Tree t = new Tree(i);
      t.info();
      t.info("overloaded method");
    }
    // Overloaded constructor:
    new Tree();
  }
}
```

Υπερφόρτωση: σειρά ορισμάτων

```
public class OverloadingOrder {
  static void print(String s, int i) {
    System.out.println("String: " + s +
      ", int: " + i);
  }
  static void print(int i, String s) {
    System.out.println("int: " + i +
      ", String: " + s);
  }
  public static void main(String[] args) {
    print("String first", 11);
    print(99, "Int first");
  }
}
```

Κλήση δομητών από δομητές (1)

```
public class Flower {
   int petalCount = 0;
  String s = new String("null");
  Flower(int petals) {
    petalCount = petals;
    System.out.println("Constructor w/ int arg
           only, petalCount= " + petalCount);
  }
  Flower(String ss) {
     System.out.println(
      "Constructor w/ String arg only, s=" + ss);
     s = ss;
  }
```

Κλήση δομητών από δομητές (2)

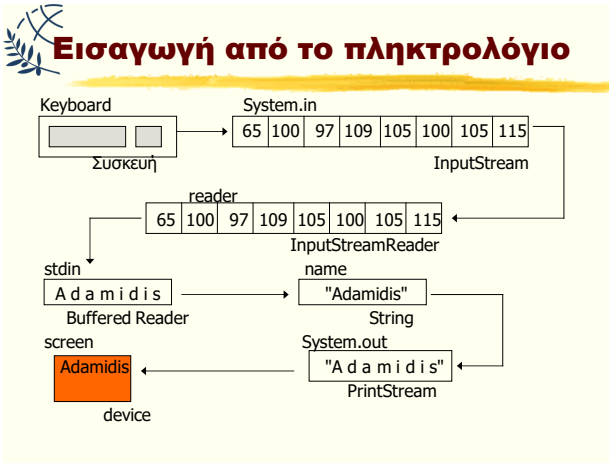
```
  Flower(String s, int petals) {
    this(petals);
    //!  this(s); // Can't call two!
    this.s = s; // Another use of "this"
    System.out.println("String & int args");
  }

  Flower() {
    this("hi", 47);
    System.out.println(
      "default constructor (no args)");
  }
```

Κλήση δομητών από δομητές (3)

```
void print() {
  //! this(11); // Μόνο μέσα σε δομητές!
    System.out.println(
      "petalCount = " + petalCount +
      " s = "+ s);
  }
  public static void main(String[] args) {
      Flower x = new Flower();
      x.print();
  }
}
```

Αντικειμενοστρεφής Προγραμματισμός

Είσοδος από πληκτρολόγιο


Εισαγωγή από το πληκτρολόγιο
Keyboard
Συσκευή
System.in
65 100 97 109 105 100 105 115
InputStream
reader
65 100 97 109 105 100 105 115
InputStreamReader
stdin
A d a m i d i s
Buffered Reader
name
"Adamidis"
String
screen
Adamidis
device
System.out
"A d a m i d i s"
PrintStream