ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS
ΚΕΝΤΡΟ ΕΠΙΜΟΡΦΩΣΗΣ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗΣ

Public API / Javadoc
Σχεδιασμός με UML

Αθανάσιος Ανδρούτσος

Απλές Data Classes

- Μέχρι στιγμής έχουμε δει μία κλάση που περιέχει ουσιαστικά μόνο δεδομένα
- Επειδή τα δεδομένα είναι private για λόγους ενθυλάκωσης χρησιμοποιούμε getters / setters για να έχουμε πρόσβαση σε αυτά
- Επίσης μας παρέχεται από το JVM ένας default constructor για να μπορούμε να δημιουργούμε instances της κλάσης
- Μπορούμε να ορίσουμε και υπερφορτωμένους constructors (μόνο που σε αυτή την περίπτωση αν θέλουμε και default constructor θα πρέπει να τον δηλώσουμε ρητά)

Functionality

- Έστω ότι θέλουμε να προσθέσουμε σε μια κλάση και **λειτουργικότητα (functionality)**, δηλαδή μεθόδους που επιτελούν επιχειρηματικές λειτουργίες (business services)
- **Σκεφτόμαστε τις μεθόδους σε όρους business services**, τι υπηρεσίες δηλαδή θέλουμε η κλάση μας να παρέχει προς τον client
- Αφού οι μέθοδοι αυτές επέχουν θέση υπηρεσιών που παρέχονται προς τρίτους, **πρέπει να είναι προσβάσιμες από όλους και άρα πρέπει να είναι public**

Παράδειγμα Κλάσης Account

- Έστω για παράδειγμα ότι θέλουμε να ορίσουμε μία κλάση **`Account`** που αναπαριστά ένα λογαριασμό τραπέζης
- Η **`Account`** θα περιέχει **ιδιωτικά πεδία** όπως το ονοματεπώνυμο του κατόχου του λογαριασμού, το IBAN, τον αριθμό ταυτότητας, το υπόλοιπο του λογαριασμού, και άλλα
- Επίσης, θα περιέχει constructors, ***setters*** και ***getters***
- Θα περιέχει όμως και **δημόσιες μεθόδους** όπως **ανάληψη**, **κατάθεση**, **ερώτηση υπολοίπου** και άλλες

JavaBean + Public API

- Η κλάση Account θα είναι ένα **JavaBean και επιπλέον θα έχει ένα Public API** (Application Programming Interface)
- Το **Public API είναι οι δημόσιες μέθοδοι**. Είναι η δημόσια διεπαφή της κλάσης μας. Σαν ένας ταμίας μία τράπεζας που παρέχει υπηρεσίες στους πελάτες
- Το **Public API είναι πολύ σημαντικό** γιατί ουσιαστικά ορίζει τις υπηρεσίες που παρέχει η κλάση μας προς τον client

Κλάση Account - Constructors


```java
package testbed.ch11;

public class Account {
    private int id;
    private String iban;
    private String firstname;
    private String lastname;
    private String ssn;
    private double balance;

    public Account() {}

    public Account(int id, String iban, String firstname,
                   String lastname, String ssn, double balance) {
        this.id = id;
        this.iban = iban;
        this.firstname = firstname;
        this.lastname = lastname;
        this.ssn = ssn;
        this.balance = balance;
    }
```

- Η κλάση Account περιέχει πεδία για το id, iban, lastname, firstname, social security number (ssn) και balance
- Τα πεδία της κλάσης **είναι όλα private λόγω ενθυλάκωσης**
- Το id λειτουργεί ως surrogate key (unique identifier)
- Παρέχονται **δύο constructors,** ένας default και ένας υπερφορτωμένος

Setters / Getters


```
23      public int getId() { return id; }
26      public void setId(int id) { this.id = id; }
29      public String getIban() { return iban; }
32      public void setIban(String iban) { this.iban = iban; }
35      public String getFirstname() { return firstname; }
38      public void setFirstname(String firstname) { this.firstname = firstname; }
41      public String getLastname() { return lastname; }
44      public void setLastname(String lastname) { this.lastname = lastname; }
47      public String getSsn() { return ssn; }
50      public void setSsn(String ssn) { this.ssn = ssn; }
53      public double getBalance() { return balance; }
56      public void setBalance(double balance) { this.balance = balance; }
```

- Για κάθε πεδίο ορίζουμε setters και getters

Μέθοδος deposit


```
60      // Public API
61
62      /**
63       * Deposits a certain amount of money.
64       *
65       * @param amount
66       *          the amount of money to be deposited.
67       * @throws Exception
68       *          if the amount is negative.
69       */
70      public void deposit(double amount) throws Exception {
71          try {
72              if (amount < 0) {
73                  throw new Exception("Negative amount exception");
74              }
75              balance += amount;
76          } catch (Exception e) {
77              e.printStackTrace();
78              throw e;
79          }
80      }
```

- Η **deposit()** λαμβάνει ως είσοδο ένα **ποσό κατάθεσης** και αφού ελέγξει αν το ποσό είναι αρνητικό, κάνει throw ένα exception.
- Αν το ποσό είναι θετικό ή μηδέν, το προσθέτει στο υπόλοιπο του λογαριασμού
- Παρατηρήστε ότι η λογική της εφαρμογής ορίζει πότε θα γίνει throw ένα exception, ενώ το exception γίνεται catch, log και rethrow

Μέθοδος withdraw


```java
/**
 * Withdraws an amount of money based on
 * a valid ssn.
 *
 * @param amount
 *          the amount to be withdrawn.
 * @param ssn
 *          the given ssn.
 * @throws Exception
 *          if the ssn is not valid or the balance
 *          is not sufficient.
 */
public void withdraw(double amount, String ssn) throws Exception {
    try {
        if (!isSsnValid(ssn)) {
            throw new Exception("Ssn not valid exception");
        }
        if (amount > balance) {
            throw new Exception("Insufficient balance exception");
        }
        balance -= amount;
    } catch (Exception e) {
        e.printStackTrace();
        throw e;
    }
}
```

- Η ***withdraw()*** λαμβάνει ως είσοδο ένα ***ποσό ανάληψης*** και αν το ssn δεν είναι valid κάνει throw ένα Exception. Επίσης αν το ποσό ανάληψης (amount) είναι μεγαλύτερο από το υπόλοιπο, τότε επίσης γίνεται throw ένα Exception.
- Αλλιώς αφαιρεί το ποσό από το υπόλοιπο του λογαριασμού

Μέθοδος getAccountBalance


```
109     /**
110      * Returns the balance of the account.
111      *
112      * @return
113      *      the account's balance.
114      */
115     public double getAccountBalance() {
116         return getBalance();
117     }
```

- Χρησιμοποιεί τον getter του balance και επιστρέφει το υπόλοιπο του λογαριασμού
- Η getAccountBalance ανήκει στο public API

Μέθοδος getAccountState


```
119     /**
120      * Returns the account's state in string format.
121      *
122      * @return
123      *      the string-representation of the state of the account.
124      */
125     public String accountToString() {
126         return "(" + id + ", " + iban + ", " + firstname + ", " +
127                 lastname + ", " + ssn + ", " + balance + ")";
128     }
```

- Η **accountToString**() επιστρέφει ένα String που αναπαριστά το state του Account. Χρησιμοποιείται όταν ο client θέλει να εκτυπώσει ένα account object. Να υπογραμμίσουμε ότι 'εκτύπωση ενός object' σημαίνει την εκτύπωση του state και επομένως την μετατροπή των πεδίων του object σε String

Μέθοδος isSsnValid


```
130     private boolean isSsnValid(String ssn) {
131         return this.ssn.equals(ssn);
132     }
133 }
```

- Παρατηρούμε ότι η μέθοδος isSsnValid είναι private γιατί δεν είναι μέρος του Public API. Δεν είναι μέθοδος που παρέχεται ως υπηρεσία στους χρήστες, αλλά αφορά τις εσωτερικές λειτουργίες (internals) της κλάσης

API (Application Programming Interface)

- Οι δημόσιες μέθοδοι μιας κλάσης αναφέρονται και ως **Public API (Application Programming Interface)**
- Πρόκειται ουσιαστικά για τις **υπηρεσίες που παρέχει η κλάση προς τους clients**
- Το να ξέρουμε πως να καλέσουμε το API μιας κλάσης (που πιθανά δεν έχουμε αναπτύξει εμείς) είναι αρκετό για να χρησιμοποιήσουμε τη λειτουργικότητά της
- Τα **Javadoc που δίνουμε στις public μεθόδους μιας κλάσης λειτουργούν ως documentation** προς όσους θέλουν να χρησιμοποιήσουν το API μας

Private Methods

- Οι μέθοδοι που **δεν αποτελούν μέρος του public API πρέπει να ορίζονται ως private**
- Οι **private** μέθοδοι (private methods) δεν είναι ορατές στον «έξω κόσμο» δηλαδή δεν μπορούν να κληθούν από άλλες κλάσεις ή αντικείμενα μέσα και έξω από το package παρά μόνο από την ίδια την κλάση

Withdraw updated

- Έστω ότι θέλουμε όταν κάνουμε *withdraw* να ελέγχουμε και το ssn αυτού που προσπαθεί να κάνει ανάληψη και αν ισούται με το ssn του κατόχου του λογαριασμού, τότε να γίνεται η ανάληψη
- Τότε όπως είδαμε η μέθοδος isSsnvallid() μπορεί να είναι private

Μέθοδος isSsnValid


```
117     private boolean isSsnValid(String ssn) {
118         return this.ssn.equals(ssn);
119     }
```

- Με this.ssn εννοείται το ssn της κλάσης (για την ακρίβεια του instance)
- Αν το ssn που περνάμε ως παράμετρο είναι ίσο με το ssn του instance της κλάσης, τότε επιστρέφουμε true. Σε κάθε άλλη περίπτωση επιστρέφουμε false

Αντικείμενα κλάσεων

- Αφού ορίσουμε την κλάση **`Account`** στην συνέχεια μπορούμε να ορίζουμε στιγμιότυπα της Account μέσα σε μια Main class
- Οι όροι στιγμιότυπα (**instances**) και αντικείμενα (**objects**) είναι ισοδύναμοι και εναλλακτικοί

Test με Client (main)

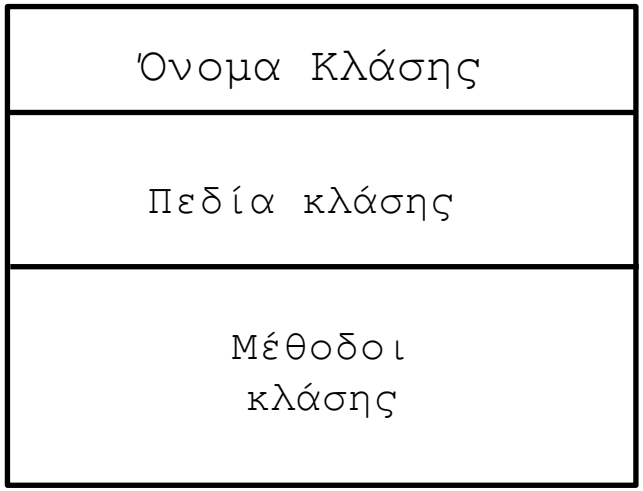

```java
package testbed.ch11;

public class AccountMain {

    public static void main(String[] args) {
        Account alice = new Account(1, "GR12345", "Alice", "W.", "R123", 100);

        try {
            alice.deposit(900);
            System.out.println("Successful deposit");

            alice.withdraw(500, "R123");
            System.out.println("Successful withdrawal");

            System.out.println("Alice account balance: " + alice.getAccountBalance());
            System.out.println(alice.getAccountState());

            alice.withdraw(100, "T456");
            System.out.println("Successful withdrawal");
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
}
```

Σχεδιασμός κλάσεων

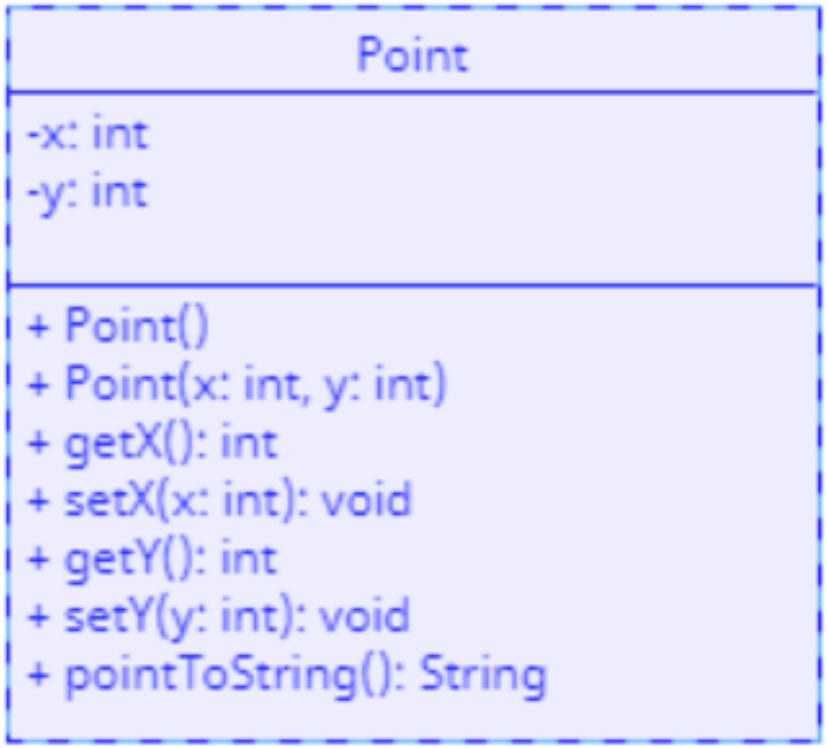
- Είναι χρήσιμο πριν υλοποιούμε να σχεδιάζουμε, ιδιαίτερα σε μεγάλες εφαρμογές
- **UML** – Unified Modeling Language
  - Απεικονίζει τις κλάσεις και τις σχέσεις μεταξύ τους διαγραμματικά
  - Χρησιμοποιείται για το σχεδιασμό κλάσεων και σχέσεων κλάσεων

Κανόνες UML

- Αναπαράσταση κλάσεων με παραλληλόγραμμα που χωρίζονται σε τρία μέρη: ***(1) Όνομα κλάσης, (2) Πεδία κλάσης, (3) Μέθοδοι κλάσης***

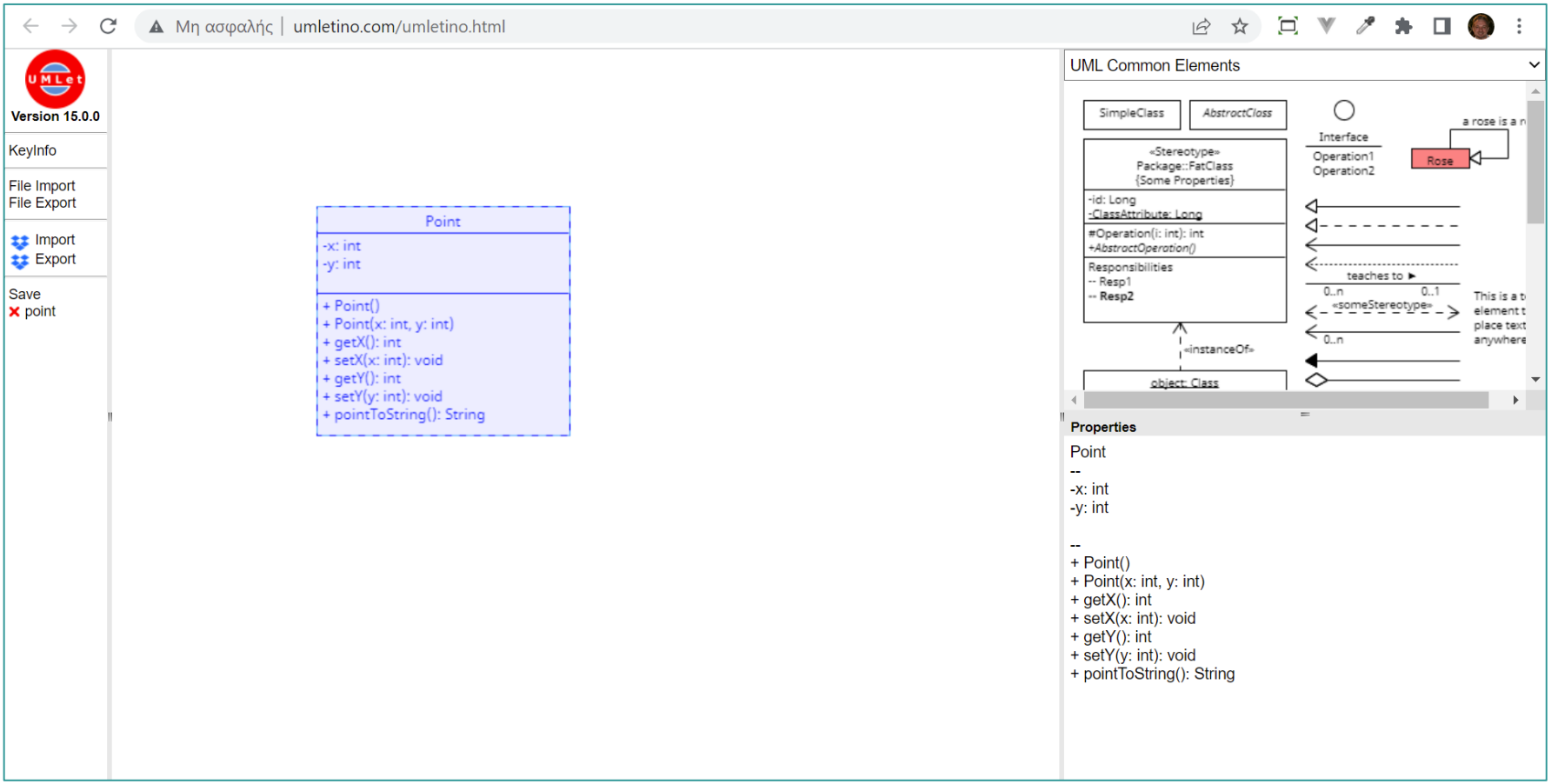


Παράδειγμα: Κλάση Point σε UML



- Έστω ότι θέλουμε να ορίσουμε σε UML μία κλάση για να απεικονίσουμε ένα σημείο στο επίπεδο που έχει συντεταγμένες (x, y) με x, y ακέραιους
- Για να ορίσουμε την ορατότητα (visibility) πεδίων και μεθόδων χρησιμοποιούμε:
  - **+** για public
  - **-** για private
  - **#** για protected
  - **~** για package private
- Οι στατικές μέθοδοι και πεδία είναι <u>υπογραμμισμένα</u>
- Οι σταθερές με κεφαλαία

Umletino.com



- Drag & Drop και ορίζουμε τα properties κάτω δεξιά. Κάνουμε save

Παράδειγμα κλάσης Point

- Στο πλαίσιο ορισμού των πεδίων της κλάσης Point για να απεικονίσουμε ένα σημείο στο επίπεδο με βάση δύο ακεραίους αριθμούς που ορίζουν τις συντεταγμένες του σημείου χρειαζόμαστε **δύο πεδία:** **`int x`** και **`int y`**
- Στην UML τα πεδία αναπαρίστανται με το όνομά τους μία άνω-κάτω τελεία (colon) ακολουθούμενη από τον τύπο του πεδίου. Τα μείον μπροστά από το όνομα κάθε πεδίου συμβολίζει στην UML ότι τα πεδία είναι private

Δημιουργία σημείου (1)

- Η δημιουργία ενός αντικειμένου τύπου Point με συντεταγμένες (0, 0) μπορεί να γίνεται καλώντας με new τον default constructor. Ο default constructor της κλάσης Point είναι η μέθοδος Point()
- Ο default constructor δεν παίρνει παραμέτρους και αρχικοποιεί σε default τιμές τα πεδία του αντικειμένου. Στο Point θα αρχικοποιήσει τα x και y στο 0 γιατί x και y είναι int και το JVM τους int που ανήκουν στον επίπεδο της κλάσης–αν δεν τους αρχικοποιήσει ο constructor- τους αρχικοποιεί αυτόματα στο 0

Δημιουργία σημείου (2)

- Αν όμως θέλαμε να μπορούμε να δημιουργήσουμε ένα σημείο, όχι μόνο το (0,0) αλλά γενικά ένα σημείο (x, y) με τιμές που θέλουμε εμείς, έστω (2, 4) ή (5, 12) κλπ. πως θα το κάναμε;
- **Θα ήταν βολικό να ορίσουμε ένα υπερφορτωμένο constructor** που να παίρνει ως τυπικές παραμέτρους τις τιμές x, y. Η δήλωση θα είναι

  public Point(int x, int y)

- Λέγεται ***υπερφορτωμένος*** γιατί –όπως έχουμε πει - έχει το ίδιο όνομα με τον default constructor αλλά διαφορετική υπογραφή

Δημιουργία σημείου (3)

- Έτσι θα έχουμε δύο constructors, έναν default (non-argument) και έναν υπερφορτωμένο, ως εξής:
  - ***public Point()***
  - ***public Point(int x, int y)***
- Όπως παρατηρούμε οι constructors δεν έχουν επιστρεφόμενη τιμή (ούτε void)

Πρόσβαση στις συντεταγμένες

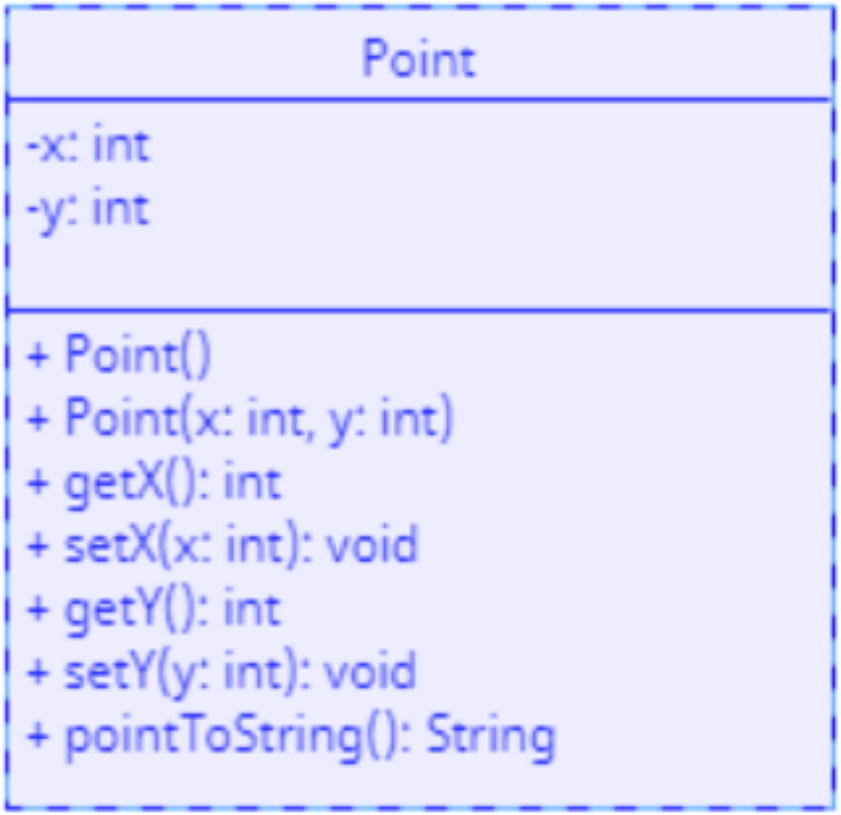
- Επειδή τα x, y της κλάσης θα είναι private (σύμφωνα με την βασική αρχή της ενθυλάκωσης) και **δεν μπορούμε άμεσα** να ανακτήσουμε (get) και να αλλάξουμε (set) τις τιμές τους, θα πρέπει να δημιουργήσουμε public setters και getters

Μετατροπή του State της κλάσης σε String

- Η μέθοδος *pointToString* είναι μία utility method (δεν είναι business logic) που 'μετατρέπει' το state ενός αντικειμένου Point σε String, ώστε η main να μπορεί να κάνει *println* points

Σχεδιασμός Point σε UML

- Έτσι τελικά έχουμε αυτό που είδαμε και στην αρχή δηλαδή το UML Class Diagram της κλάσης Point



Όνομα κλάσης

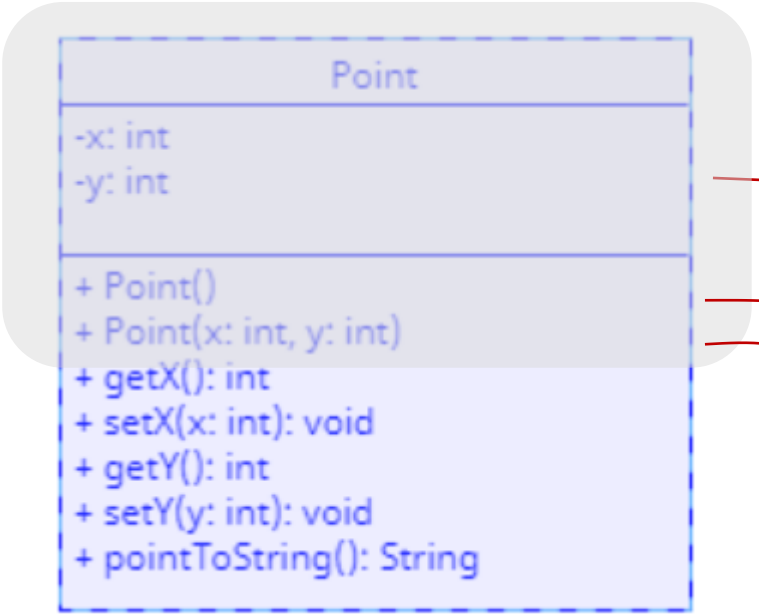
Πεδία κλάσης (private)

Δημιουργοί (Default και υπερφορτωμένος)

Setters
Getters

pointToString

Σχεδιασμός και υλοποίηση (1)




```java
package gr.aueb.cf.testbed;

/**
 * Ορίζει ένα σημείο στο επίπεδο με
 * συντεταγμένες x, y.
 *
 * @author A. Androutsos
 */
public class Point {

    private int x;
    private int y;

    public Point() {}

    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }
```

- Το **this** που χρησιμοποιείται στην **Point(int x, int y)** είναι προκαθορισμένη αναφορική μεταβλητή (δείκτης) της Java που δείχνει στο ***ίδιο το αντικείμενο*** (τρέχον, όποιο κι αν είναι αυτό) ώστε να διαχωρίζουμε τα **x, y** που είναι τυπικές παράμετροι από τα **x, y** της κλάσης/αντικειμένου που είναι **this.x** και **this.y**

Default Constructor (1)

- Στη γραμμή 14 της προηγούμενης διαφάνειας ο default constructor δεν έχει σώμα. Εναλλακτικά θα μπορούσε να περιέχει το παρακάτω αλλά είναι περιττό:

```
14  @       public Point() {
15              x = 0;
16              y = 0;
17          }
```

- Είναι το ίδιο πράγμα μιας και όπως αναφέραμε το JVM αρχικοποιεί αυτόματα τους int μιας κλάσης στο 0

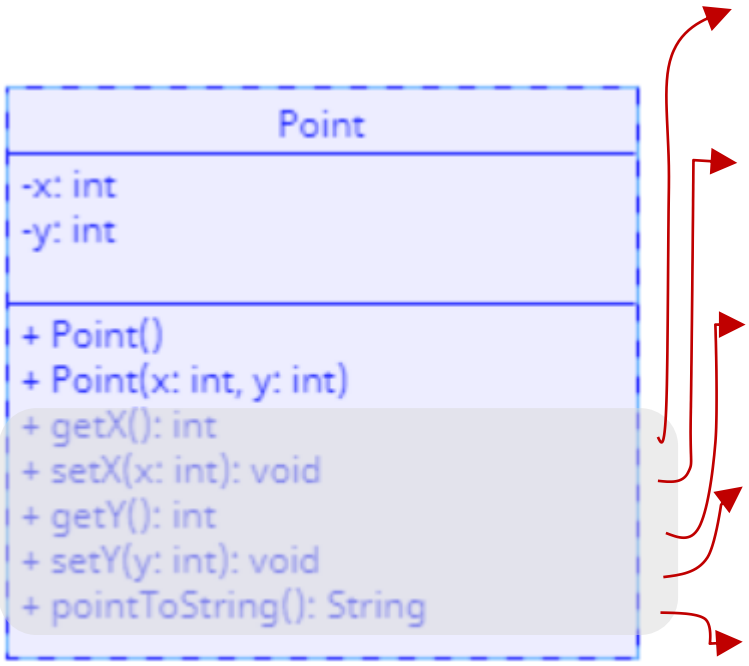
Default Constructor (2)

- Επίσης, θα ήταν το ίδιο αν γράφαμε τις αναφορές στα πεδία της κλάσης με την χρήση του δείκτη ***this***

```
14  @       public Point() {
15              this.x = 0;
16              this.y = 0;
17          }
```

- Ωστόσο εδώ δεν είναι απαραίτητη η χρήση του ***this*** όπως στον υπερφορτωμένο constructor, γιατί εδώ δεν έχουμε τυπικές παραμέτρους x, y ώστε να υπάρχει conflict (σύγκρουση) ονομάτων και γιαυτό σε αυτή την περίπτωση η χρήση του this είναι περιττή

Σχεδιασμός και υλοποίηση (2)




```java
public int getX() {
    return x;
}

public void setX(int x) {
    this.x = x;
}
public int getY() {
    return y;
}

public void setY(int y) {
    this.y = y;
}

public String pointToString() {
    return "(" + x + "," + y + ")";
}
```

- Συνεχίζουμε και υλοποιούμε ***setters και getters*** καθώς και την ***pointToString***

Η Point class με Javadoc


```java
package gr.aueb.cf.testbed.pointapp;

/**
 * The Point class represents <i>points</i> in a
 * two-dimensional space, where <b>x</b> and <b>y</b>
 * coordinates are considered integers. All points,
 * suc as (0, 0) or (5, 74) could be implemented as
 * instances of this class.
 */
public class Point {
    int x;
    int y;

    /**
     * Initializes a newly created point,
     * so that it represents a (0,0) point.
     */
    public Point() {}
```

- Στα Javadoc, τα οποία περικλείονται μέσα σε /** */ (τα υπόλοιπα αστεράκια * είναι προαιρετικά) μπορούμε να δίνουμε και HTML tags
- Javadoc μπορούμε να δίνουμε στο επίπεδο της **κλάσης**, στο επίπεδο των **μεθόδων** και στο επίπεδο των **packages** (για τα packages χρησιμοποιούμε αρχεία package-info.java μέσα σε κάθε package)

Point overloaded constructor


```
20      /**
21       * Constructs a new Point with specific
22       * (x,y) coordinates.
23       *
24       * @param x     the x coordinate
25       * @param y     the y coordinate
26       */
27      public Point(int x, int y) {
28          this.x = x;
29          this.y = y;
30      }
```

- Javadoc σε επίπεδο μεθόδου

x coordinate - getter and setter


```
32      /**
33       * Gets x-coordinate.
34       *
35       * @return
36       *      the value of x-coordinate.
37       */
38      public int getX() {
39          return x;
40      }
41
42      /**
43       * Sets x-coordinate.
44       *
45       * @param x
46       *      the x-coordinate.
47       */
48      public void setX(int x) {
49          this.x = x;
50      }
```

- Επίσης τεκμηριώνουμε setters και getters εφόσον είναι public και άρα μέρος του API

y coordinate - getter and setter


```
52      /**
53       * Gets y-coordinate.
54       *
55       * @return
56       *      the value of y-coordinate.
57       */
58      public int getY() {
59          return y;
60      }
61
62      /**
63       * Sets y-coordinate.
64       *
65       * @param y
66       *      the y-coordinate.
67       */
68      public void setY(int y) {
69          this.y = y;
70      }
```

- Το ίδιο και εδώ με τον getter και setter για το πεδίο y

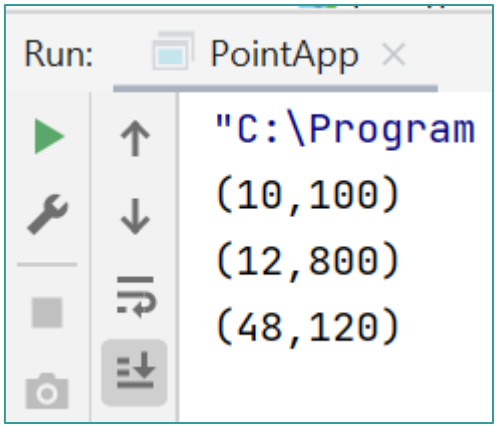
pointToString


```
72      /**
73       * Returns the state of this point.
74       *
75       * @return
76       *      the x and y coordinates transformed into String (x,y).
77       */
78      public String pointToString() {
79          return "(" + x + "," + y + ")";
80      }
81  }
```

- Το ποια είναι η ακριβής συντακτική μορφή του String που επιστρέφει η convertToString είναι είτε αυθαίρετο ή σύμφωνα με τις απαιτήσεις, αν υπάρχουν

Συνάρτηση main και έλεγχος της κλάσης Point


```
1  package gr.aueb.cf.testbed.pointapp;
2
3  public class PointApp {
4
5      public static void main(String[] args) {
6          Point p1 = new Point();
7          Point p2 = new Point(12, 800);
8          Point p3 = new Point(48, 120);
9
10         p1.setX(10);
11         p1.setY(100);
12
13         System.out.println(p1.pointToString());
14         System.out.println(p2.pointToString());
15         System.out.println(p3.pointToString());
16     }
17 }
```



Constructors και new

- Στις γραμμές 6, 7, 8 η ***new*** καλεί τους constructors

```
6        Point p1 = new Point();
7        Point p2 = new Point(12, 800);
8        Point p3 = new Point(48, 120);
```

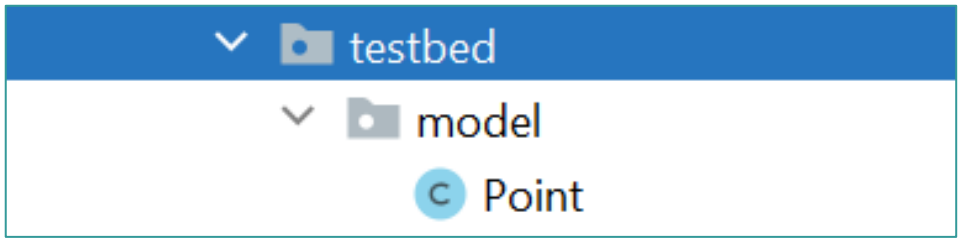
- Αυτό που γίνεται είναι:
  - Η ***new*** δεσμεύει χώρο στη μνήμη (heap) δυναμικά σε χρόνο εκτέλεσης και ***δημιουργεί τα instances*** τύπου Point ενώ οι constructors αρχικοποιούν τα fields των instances. Στα p1, p2, p3 επιστρέφονται οι διευθύνσεις μνήμης των νέο-δημιουργηθέντων αντικειμένων
- ***Οι constructors επομένως είναι βασικά αρχικοποιητές (initializers)*** γιατί δεν δεσμεύουν αυτοί τον χώρο, αλλά μόνο αρχικοποιούν τα πεδία των αντικειμένων –η δέσμευση χώρου γίνεται από τη new

Φυσικός και Λογικός Διαχωρισμός

- Μέχρι τώρα γράφαμε όλο τον κώδικά μας και τις κλάσεις μας μέσα σε ένα package
- Ωστόσο, μεθοδολογικά θα ήταν καλύτερα αν οργανώναμε λογικά το project μας σε διαφορετικά packages ανάλογα με το τι περιέχει το κάθε package

Λογική οργάνωση κλάσεων

- Στο root package, δημιουργούμε ένα sub-package με όνομα **model**
- Μέσα στο **model** θα δημιουργήσουμε την κλάση Point



Κλάση Point


```java
package gr.aueb.cf.testbed.model;

/**
 *  Point Java Bean.
 */
public class Point {
    private int x;
    private int y;

    public Point() {
    }

    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public int getX() {
        return x;
    }

    public void setX(int x) {
        this.x = x;
    }
```

```java
    public int getY() {
        return y;
    }

    public void setY(int y) {
        this.y = y;
    }

    public String convertToString() {
        return "Point{" +
                "x=" + x +
                ", y=" + y +
                '}';
    }
}
```

- Παρατηρήστε το package στην 1η γραμμή για να δείτε σε ποιο package ανήκει η κλάση

Driver Class – import (1)


```
1   package gr.aueb.cf.testbed;
2
3   import gr.aueb.cf.testbed.model.Point;
4   import gr.aueb.cf.testbed.model.*;
5
6   public class ProjectApp {
7
8       public static void main(String[] args) {
9           Point x = new Point();
10
11          gr.aueb.cf.testbed.model.Point y = new gr.aueb.cf.testbed.model.Point();
12      }
13  }
```

- Όταν θέλουμε να χρησιμοποιήσουμε το API κλάσεων που βρίσκονται σε άλλα packages και όχι στο τρέχον, θα πρέπει να κάνουμε import και να χρησιμοποιούμε τα απλά ονόματα των κλάσεων (βλ. γραμμή 9)
- **Χωρίς import** μπορούμε να τις χρησιμοποιήσουμε αλλά τα ονόματα πρέπει να είναι να FQN (Fully Qualified Names), δηλαδή τα πλήρη ονόματα από το classpath (βλ. γραμμή 11)

Driver Class – import (2)


```
1   package gr.aueb.cf.testbed;
2
3   import gr.aueb.cf.testbed.model.Point;
4   import gr.aueb.cf.testbed.model.*;
5
6   public class ProjectApp {
7
8       public static void main(String[] args) {
9           Point x = new Point();
10
11          gr.aueb.cf.testbed.model.Point y = new gr.aueb.cf.testbed.model.Point();
12      }
13  }
```

- Import μπορούμε να κάνουμε με το ακριβές όνομα της κλάσης (βλ. γραμμή 3) ή με αστεράκι, δηλαδή όλες οι κλάσεις που είναι μέσα στο package (model - βλ. γραμμή 4)
- Η 2η περίπτωση είναι πλεονασμός και δεν παρέχει readability στο ποιες κλάσεις χρησιμοποιούμε

Import με *

- Σημειωτέον αν κάνουμε :

```
import gr.aueb.cf.testbed.model.*;
```

Εννοούμε όλες τις κλάσεις μέσα στο model. Αν υπάρχει υποpackage μέσα στο model π.χ. model.users τότε πρέπει να γίνει και δεύτερο import:

```
import gr.aueb.cf.testbed.model.users.*;
```

Static import

- Μπορούμε επίσης να κάνουμε import στατικές μεθόδους, όπως για παράδειγμα από την κλάση Math

```
3    import static java.lang.Math.abs;
4    import static java.lang.Math.*;
```

```
15        int abs = abs(-10);
```

Εργασία 1

- Ορίστε σε UML και υλοποιήστε σε Java μία κλάση PointXYZ που ορίζει ένα σημείο στον χώρο με συντεταγμένες (x, y, z) κατά τον ίδιο τρόπο που ορίσαμε την κλάση Point και το σημείο (x, y) στο επίπεδο. Ορίστε μία μέθοδο **public String convertToString()** για να μετατρέπετε τα PointXYZ instances σε Strings, ώστε να εκτυπώνετε points με println
- Καθώς και 'business' μεθόδους **getXYDistance()**, **getYZDistance()** και **getXZDistance()** και **getXYZDistance()** που επιστρέφουν τις αποστάσεις μεταξύ των αντίστοιχων σημείων (ως η τετραγωνική ρίζα των αθροισμάτων των τετραγώνων της απόστασης κάθε σημείου από την αρχή των αξόνων)
- Υλοποιήστε μία άλλη κλάση **Main** που μέσα στην **main()**:
    1. Δημιουργεί ένα σημείο PointXYZ με τον υπερφορτωμένο Constructor
    2. Καλεί τις παραπάνω μεθόδους και επιβεβαιώνει τα αναμενόμενα αποτελέσματα

Εργασία 2

- Ορίστε σε **UML** και **υλοποιήστε**:
- Μία class **OverfraftAccount** που δίνει τη δυνατότητα ανάληψης ποσών μεγαλύτερων από το υπόλοιπο του λογαριασμού, καθώς και
- Μία κλάση **JointAccount** που περιλαμβάνει δύο κατόχους ενός λογαριασμού
- Δημιουργήστε ένα package **bankapp** και μέσα σε αυτό ένα package model μέσα στο οποίο να έχετε τις δύο κλάσεις
- Δημιουργήστε μία **Main class** μέσα στο bankapp package, κάντε import τις δύο παραπάνω κλάσεις και ελέγξτε τη λειτουργία τους μέσα στη **main()**
- Γράψτε doc comments για τις κλάσεις OverdraftAccount και JointAccount και τις μεθόδους του Public API

GitHub

- Δημιουργήστε στον υπολογιστή σας (local) και στο GitHub ένα repository για να κρατάτε back-up όλων των project σας στο O-O Programming
- Θα μπορούσε το repository στο GitHub να ονομαστεί java-cf-oo-projects ή κάτι αντίστοιχο με πεζά γράμματα και παύλες για να είναι πιο ευανάγνωστο