Μεθοδολογίες και Γλώσσες Προγραμματισμού ΙΙ - Java
2ο φυλλάδιο εργαστηρίου - 21/10/2024

Κλάσεις - Αντικείμενα

Όπως γνωρίζετε μια κλάση (**class**) αποτελεί την προδιαγραφή *αντικειμένων* (**object)**, τα οποία δημιουργούνται κατά την εκτέλεση του προγράμματος. Ένα αντικείμενο ενσωματώνει χαρακτηριστικά/δεδομένα αλλά και συμπεριφορά. Τα αντικείμενα της ίδιας κλάσης έχουν κοινές ιδιότητες δηλαδή τύπους δεδομένων και κοινή συμπεριφορά, δηλαδή κοινές μεθόδους.

Ακολουθεί αναλυτικό παράδειγμα έτσι ώστε να γίνει κατανοητός ο τρόπος ορισμού κλάσεων, αντικειμένων, μεταβλητών/ιδιοτήτων και μεθόδων.

Όπως μπορούμε να παρατηρήσουμε στο ακόλουθο παράδειγμα, ο ορισμός μιας κλάσης (π.χ. **Box**) περιέχει δηλώσεις των πεδίων (ιδιοτήτων) και των μεθόδων της κλάσης. Οι ιδιότητες είναι τα δεδομένα των αντικειμένων της κλάσης. Η συγκεκριμένη κλάση έχει τρείς ιδιότητες, τις πραγματικές τιμές *width*, *height* και *depth*. Οι τύποι των ιδιοτήτων μιας κλάσης, όπως και κάθε μεταβλητής που δηλώνεται στη γλώσσα, είναι δύο ειδών: **Βασικοί** (primitive) **τύποι**, π.χ. ***int***, ***char***, ***long***, ***double*** και **τύποι αναφοράς** (references) όπως ***πίνακες***, ***συμβολοσειρές***, κλπ. δηλαδή αναφορές σε άλλα αντικείμενα. Οι μέθοδοι υλοποιούν τις λειτουργίες των αντικειμένων της κλάσης και ορίζουν την συμπεριφορά τους. Η συγκεκριμένη κλάση έχει μία μέθοδο, με όνομα *volume* η οποία αναλαμβάνει να εμφανίσει τον όγκο ενός αντικειμένου χωρίς να επιστρέφει κάποια τιμή (void συνάρτηση).

```
public class Box {
double width;
double height;
double depth;

// μέθοδος για την εμφάνιση του όγκου του κουτιού.
void volume() {
   System.out.print("The volume is ");
   System.out.println(width * height * depth);
   }
}

public class BoxDemo {

public static void main(String args[]) {
//δημιουργία 2 αντικειμένων τύπου Box()
Box mybox1 = new Box();
Box mybox2; // δήλωση αναφοράς αντικειμένου
mybox2 = new Box(); // αρχικοποίηση

// εισαγωγή τιμών στο αντικείμενο mybox1
mybox1.width = 10;
mybox1.height = 20;
mybox1.depth = 15;

// εισαγωγή τιμών στο αντικείμενο mybox2
mybox2.width = 3;
mybox2.height = 6;
mybox2.depth = 9;
```

```
// εμφάνιση του όγκου των δύο αντικειμένων
mybox1.volume();
mybox2.volume();
   }
}
```

Κανόνες σωστής πρακτικής:

- Θα πρέπει να ορίζετε στις κλάσεις σας όλες τις ιδιότητες ως *private* και μόνο τις μεθόδους που απαιτείται η πρόσβαση από άλλες κλάσεις, ως *public* (υπάρχουν ελάχιστες εξαιρέσεις του κανόνα αυτού).
- Δίνουμε στις μεθόδους ονόματα που περιγράφουν τη λειτουργία τους. Ο πρώτος χαρακτήρας του ονόματος είναι πεζός ενώ το πρώτο γράμμα κάθε νέας λέξης γράφεται με κεφαλαίο χαρακτήρα, π.χ. displayMessage().

1η εργαστηριακή άσκηση

Με βάση το προηγούμενο παράδειγμα δημιουργήστε μια κλάση αντικειμένων **Student** η οποία να περιγράφει τα στοιχεία ενός φοιτητή. Ένας φοιτητής έχει το επώνυμο του, το όνομα του και τον αριθμό μητρώου του.
Φροντίστε τα δεδομένα της κλάσης να προστατεύονται από ανεξέλεγκτη πρόσβαση από άλλα σημεία του κώδικα (αρχή της ενθυλάκωσης). Συνεπώς η κλάση που θα δημιουργήσετε θα πρέπει να περιλαμβάνει και μεθόδους εισαγωγής των στοιχείων του φοιτητή (set μεθόδους για κάθε ιδιότητα ξεχωριστά). Τέλος δημιουργήστε και κατάλληλη μέθοδο που θα επιστρέφει όλες τις τιμές των ιδιοτήτων της κλάσης, ως συμβολοσειρά (*toString()*).
Στη συνέχεια υλοποιήστε ένα κυρίως πρόγραμμα (Main κλάση) στο οποίο να δημιουργήσετε δύο αντικείμενα της κλάσης Student. Εισάγετε κατάλληλα δεδομένα σε αυτά και εκτυπώστε τα.

Δημιουργία αντικειμένου μιας κλάσης

Για να δημιουργήσουμε ένα αντικείμενο (instance) μιας κλάσης πρέπει να καλέσουμε τον δημιουργό (constructor) της κλάσης. Αν και ο δημιουργός μοιάζει με μέθοδο, δεν είναι. Αυτό φαίνεται και από το γεγονός ότι ένας δημιουργός δεν έχει επιστρεφόμενη τιμή. Αν δεν γράψουμε εμείς έναν δημιουργό για την κλάσης μας, δημιουργείται αυτόματα ένας (default constructor) χωρίς παραμέτρους. Σκοπός του είναι να δώσει την τιμή 0 στις πρωταρχικές μεταβλητές και την τιμή null στις αναφορές σε αντικείμενα.
Μια κλάση μπορεί να έχει περισσότερους από έναν δημιουργό (overloaded constructor). Αυτοί οι δημιουργοί πρέπει όμως να διαφέρουν στην υπογραφή τους, δηλαδή στο πλήθος, στον τύπο και/ή την σειρά των παραμέτρων που δέχονται. Ο ρόλος ενός δημιουργού είναι να αρχικοποιήσει την κατάσταση του νέου αντικειμένου.

Χαρακτηριστικό είναι το παράδειγμα που ακολουθεί:

```
// Κλάση για την διαχείριση ασφαλιστηρίων
public class CarInsurancePolicy {
    private int policyNumber;    // Αριθμός ασφαλιστήριου συμβολαίου
    private int paymentsNumber;  // Αριθμός δόσεων ανά έτος
    private String residentCity; // Πόλη κατοικίας ασφαλισμένου

    // Constructor που αρχικοποιεί και τις τρεις παραμέτρους
    public CarInsurancePolicy(int number, int payments, String city) {
        policyNumber = number;
```

```
        paymentsNumber = payments;
        residentCity = city;
    }

// Έστω ότι το πρακτορείο πουλά ασφάλειες κυρίως στην πόλη που εδρεύει δηλαδή
// στην Αθήνα. Ορίζουμε Constructor που θέτει την Αθήνα ως τιμή στο σχετικό πεδίο
    public CarInsurancePolicy(int number, int payments) {
        policyNumber = number;
        paymentsNumber = payments;
        residentCity = "Αθήνα";
        }
}
```

Ένας constructor καλείται μόνο κατά τη δημιουργία ενός αντικειμένου. Ο προγραμματιστής δε μπορεί να καλέσει άμεσα έναν constructor όπως μπορεί οποιαδήποτε άλλη απλή μέθοδο, είναι όμως αυτός που εκκινεί τη διαδικασία κλήσης κάποιου constructor γράφοντας μία εντολή στην οποία δημιουργείται κάποιο αντικείμενο. Όταν λοιπόν ο διερμηνέας συναντήσει μια τέτοια έκφραση, καλείται αυτόματα ο αντίστοιχος constructor, του οποίου όπως είπαμε η βασική λειτουργία είναι να δεσμεύσει την απαραίτητη μνήμη και να δημιουργήσει το αντικείμενο.

Εναλλακτικά οι δύο constructors της κλάσης CarInsurancePolicy θα μπορούσαν να υλοποιηθούν και ως εξής:

```
    public CarInsurancePolicy(int number, int payments, String city) {
        policyNumber = number;
        paymentsNumber = payments;
        residentCity = city;
    }

    public CarInsurancePolicy(int number, int payments) {
        this(number, payments, "Αθήνα");
    }
```

Σημείωση : Με τη χρήση του ***this()*** έχουμε τη δυνατότητα να καλέσουμε τον constructor της τρέχουσας κλάσης ενώ με το ***super()***, τον constructor της υπερκλάσης (κληρονομικότητα).

Σε αντίθεση με τη C++ όπου υπάρχει η έννοια του destructor, στη Java η καταστροφή των αντικειμένων γίνεται εξ' ολοκλήρου με ευθύνη της ίδιας της γλώσσας, η οποία διαθέτει τους κατάλληλους μηχανισμούς που φέρουν σε πέρας τη διαδικασία αυτή. Πιο συγκεκριμένα, όταν ένα αντικείμενο έχει ολοκληρώσει τη διάρκεια ζωής του, ή όταν έχει χάσει όλες τις αναφορές σε αυτό, τότε «χαρακτηρίζεται» ως υποψήφιο για διαγραφή. Το αντικείμενο παραμένει στη μνήμη του υπολογιστή μέχρις ότου αναλάβει δράση ο *συλλέκτης απορριμμάτων (garbage collector)*.
Ο garbage collector είναι ένα πρόγραμμα που εκτελείται ανά τακτά χρονικά διαστήματα και έχει ως σκοπό όπως είναι ξεκάθαρο από το όνομα του, να περισυλλέξει τα απορρίμματα που βρίσκονται στη μνήμη, τα αντικείμενα δηλαδή που έχουν χαρακτηρισθεί ως υποψήφια για διαγραφή. Ο garbage collector λοιπόν, κάθε φορά που θα τρέξει θα διαγράψει όσα αντικείμενα έχουν ολοκληρώσει τη διάρκεια ζωής τους και θα απελευθερώσει τη δεσμευμένη από αυτά μνήμη.

2η εργαστηριακή άσκηση

Επεκτείνετε το παραπάνω παράδειγμα δημιουργώντας μία κλάση που θα δημιουργεί αντικείμενα **CarInsurancePolicy** καλώντας τους αντίστοιχους δημιουργούς. Δημιουργήστε έναν ακόμα δημιουργό που απαιτεί μόνο την παράμετρο για τον αριθμό του ασφαλιστήριου συμβολαίου. Θα πρέπει να χρησιμοποιεί την προεπιλεγμένη τιμή των δύο δόσεων κατά έτος και την «Αθήνα» ως πόλη κατοικίας. Προσθέστε μια μέθοδο display() η οποία εμφανίζει όλα τα δεδομένα του ασφαλιστήριου συμβολαίου.

3η εργαστηριακή άσκηση

Εντοπίστε ποιο είναι το λάθος στον ακόλουθο κώδικα:

```
public class Student {
private String onoma;
private String eponimo;
private int am;

public Student(String onoma, String eponimo, int am) {
       onoma = onoma;
       eponimo = eponimo;
        am = am;
}

public String toString() {
       return onoma + " " + eponimo + " " + am;
         }
}
```

Μέθοδοι μιας κλάσης – Υπερφόρτωση μεθόδων

Η δήλωση της μεθόδου μιας κλάσης αποτελείται από :

- προσδιοριστικό ορατότητας (public, private, protected, (default))
- επιστρεφόμενο τύπο (int, double, void, …)
- όνομα συνάρτησης
- λίστα παραμέτρων
- το σώμα της μεθόδου

Επίσης και οι μέθοδοι μιας κλάσης μπορούν να υπερφορτωθούν. Η υπερφόρτωση μεθόδων μας επιτρέπει να έχουμε μεθόδους με το ίδιο ακριβώς όνομα αλλά διαφορετική υπογραφή (signature). Ως υπογραφή μιας μεθόδου ορίζεται το τμήμα της που αποτελείται από το όνομά της και από το όρισμά της, όπως φαίνεται στη γραμμή που ακολουθεί.

```
int sumInt(int x, int y)
```

Method signature

Έτσι λοιπόν μια κλάση μπορεί να έχει υπερφορτωμένες μεθόδους που έχουν το ίδιο όνομα αλλά διαφέρουν στον αριθμό και/ή τύπο των παραμέτρων που δέχονται. Οι υπερφορτωμένες μέθοδοι εκτελούν την ίδια λειτουργία με διαφορετικό τρόπο.

Υπάρχουν συγκεκριμένοι κανόνες που θα πρέπει να προσέχουμε κατά την υπερφόρτωση :

- Οι μέθοδοι θα πρέπει να αλλάζουν την λίστα παραμέτρων (την υπογραφή). Αν η υπογραφή δύο η περισσοτέρων μεθόδων είναι η ίδια, θα προκύψει σφάλμα.

- Οι μέθοδοι μπορούν να αλλάξουν τον τύπο επιστροφής. Ο τύπος επιστροφής δεν παίζει κανέναν ρόλο στην υπερφόρτωση μιας και δεν είναι μέρος της υπογραφής.
- Οι μέθοδοι μπορούν να αλλάξουν τον προσδιοριστή πρόσβασης χωρίς όμως το επίπεδο πρόσβασης να είναι πιο περιοριστικό από το αρχικό.
- Οι μέθοδοι μπορούν να υπερφορτωθούν στην ίδια κλάση ή σε κάποια υποκλάση της.

```
public class Clock {
private int hours, minutes, seconds;

public Clock() {
    hours = 0;
    minutes = 0;
    seconds = 0;
}

public Clock (int h, int min, int sec) {
    if ((sec >= 0) && (sec < 60) &&
        (min >= 0) && (min < 60) &&
        (h >= 0) && (h < 24)) {
        hours = h;
        minutes = min;
        seconds = sec;
    }
    else {
        hours = 0;
        minutes = 0;
        seconds = 0;
    }
}

public void setTime (int h, int min, int sec){
    if ((sec >= 0) && (sec < 60))
        seconds = sec;
    else
        seconds=0;

    if ((min >= 0) && (min < 60))
        minutes = min;
    else
        minutes=0;

    if ((h >= 0) && (h < 24))
        hours = h;
    else
        hours =0;
}

public void setTime (int h, int min){
    seconds = 0;
    if ((min >= 0) && (min < 60))
        minutes = min;
    else
        minutes=0;

    if ((h >= 0) && (h < 24))
        hours = h;
    else
        hours =0;
```

```
}

public void display_Clock() {
    System.out.println ("Time is: " + hours + ":" + minutes + ":" + seconds);
    }
}


public class ClockDemo {
public static void main(String[] args) {
Clock rolex = new Clock();
Clock omega = new Clock(17, 49, 23);

omega.display_Clock();
rolex.setTime(13, 59);
rolex.display_Clock();
omega.setTime(12, 45, 50);
omega.display_Clock();
    }
}
```

4η εργαστηριακή άσκηση

Μελετήστε το παρακάτω java πρόγραμμα. Ποια μέθοδος της κλάσης Overload θα κληθεί σε κάθε περίπτωση.

```
public class Overload {
void test() {
    System.out.println("No parameters");
}

void test(int a, int b) {
    System.out.println("a and b: " + a + " " + b);
}

void test(double a) {
    System.out.println("Inside test(double) a: " + a);
    }
}

public class OverloadDemo {
public static void main(String args[]) {
    Overload ob = new Overload ();
    int i = 30;
    ob.test();
    ob.test(20, 40);
    ob.test(i);
    ob.test(13.5);
    }
}
```

5η εργαστηριακή άσκηση

Μελετήστε το παρακάτω τμήμα προγράμματος που χρησιμοποιεί υπερφόρτωση μεθόδων. Θεωρείτε ότι υπάρχει περίπτωση να δημιουργηθεί ασάφεια στην περίπτωση κλήσης των μεθόδων με συγκεκριμένο τύπο παραμέτρων; Δικαιολογήστε την απάντηση σας.

```
public static void calculate(int amount, double rate);
public static void calculate(double amount, int rate);
```

Δημιουργία και Επεξεργασία Τυχαίων Αριθμών

Η δημιουργία τυχαίων αριθμών μπορεί να επιτευχθεί με δύο τρόπους. Είτε χρησιμοποιώντας:

- ένα αντικείμενο της κλάσης Random που βρίσκεται στο πακέτο java.util (https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/Random.html)

```
Random randNums = new Random();
int randValue = randNums.nextInt(6);
```

Τα αντικείμενα της κλάσης Random παράγουν τυχαίες τιμές των τύπων boolean, byte, float, double, int, long και Gaussian. Στο παράδειγμα αυτό η επιστρεφόμενη τιμή θα είναι μια ακέραια τιμή από το [0 – 6).

- είτε τη static μέθοδο random() της κλάσης Math, η οποία επιστρέφει μια τυχαία τιμή του τύπου double από 0.0 έως 1.0, μη συμπεριλαμβανόμενης της τιμής 1.0.

```
int randomValue = (int) (Math.random()*10);
```

6η Εργαστηριακή άσκηση

Υλοποιήστε μια main κλάση η οποία θα παράγει 10 τυχαίους αριθμούς στο διάστημα [20..50], θα εμφανίζει τους αριθμούς αυτούς, θα βρίσκει το μέγιστο και θα τον τυπώνει. Για τη δημιουργία των τυχαίων αριθμών χρησιμοποιήστε έναν από τους παραπάνω τρόπους.