ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

Αντικειμενοστραφής Προγραμματισμός

Ενότητα 4: Επί πλέον έννοιες αντικειμένων

Χ. ΑΓΓΕΛΗ

Τμήμα Ηλεκτρολόγων και Ηλεκτρονικών Μηχανικών

Μαθησιακά αποτελέσματα

- Κατανόηση των ενοτήτων και των πεδίων εφαρμογής
- Υπερφόρτωση μεθόδων
- Αποφυγή ασάφειας
- Δημιουργία και κλήση μεθόδων κατασκευής με παραμέτρους
- Χρήση της αναφοράς `this`

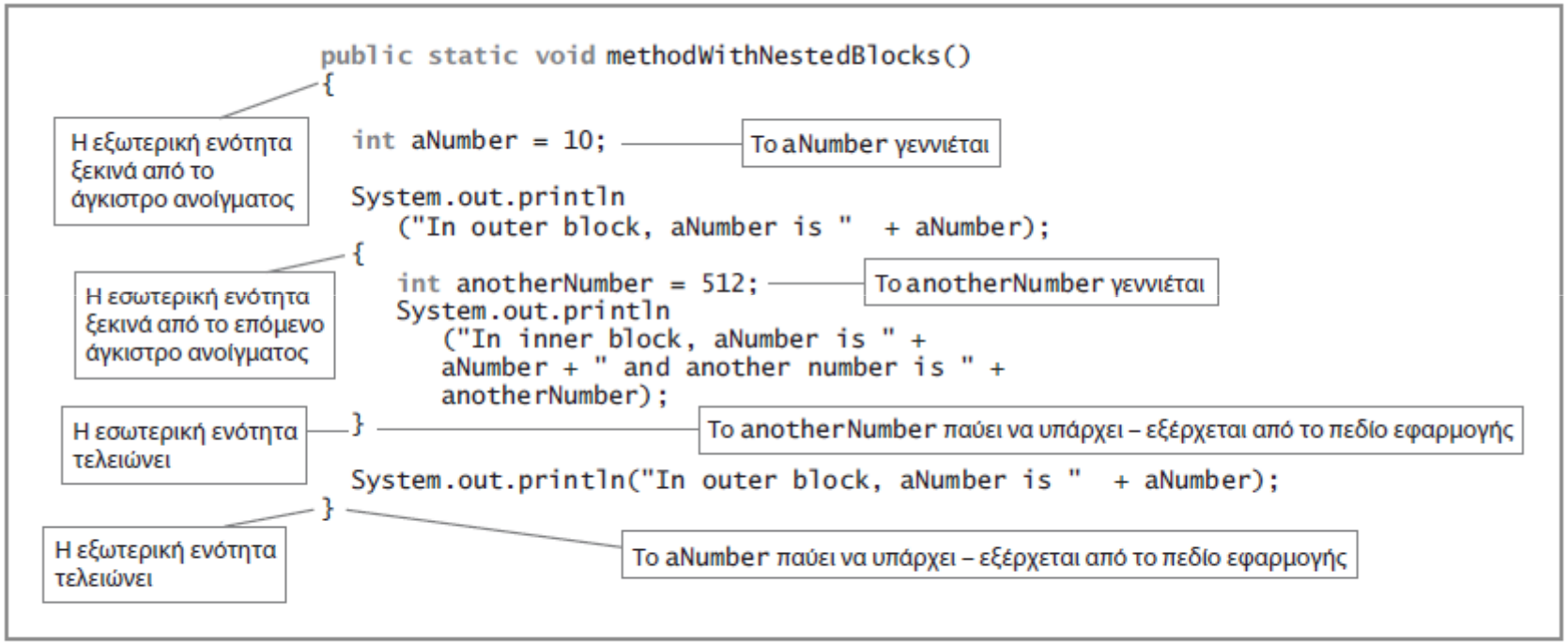
Μαθησιακά αποτελέσματα (συνέχεια)

- Χρήση στατικών πεδίων
- Χρήση αυτόματα εισαγόμενων, έτοιμων σταθερών και μεθόδων
- Χρήση σύνθεσης και ένθεσης κλάσεων

Κατανόηση των ενοτήτων και των πεδίων εφαρμογής

- **Ενότητες**
  - Χρησιμοποιούν άγκιστρα ανοίγματος και κλεισίματος
  - Μπορεί να υφίσταται εξ ολοκλήρου μέσα σε άλλη ενότητα ή εντελώς έξω και ξεχωριστά από άλλη ενότητα
  - Δεν επιτρέπεται η επικάλυψη
  - Τύποι:
    - **Εξωτερική ενότητα**
    - **Εσωτερική ενότητα**
    - **Ένθετη** (περιέχεται εξ ολοκλήρου μέσα στην εξωτερική ενότητα)

Κατανόηση των ενοτήτων και των πεδίων εφαρμογής (συνέχεια)



Εικόνα 4-1

Μέθοδος με ένθετες ενότητες

Κατανόηση των ενοτήτων και των πεδίων εφαρμογής (συνέχεια)

- Πεδίο εφαρμογής (εμβέλεια μεταβλητής)
  - Το τμήμα ενός προγράμματος μέσα στο οποίο είναι γνωστή μια μεταβλητή
  - **Εισέρχεται στο πεδίο εφαρμογής**
    - Ξεκινά η ύπαρξη της μεταβλητής
  - **Εξέρχεται από το πεδίο εφαρμογής**
    - Η μεταβλητή παύει να υπάρχει

Κατανόηση των ενοτήτων και των πεδίων εφαρμογής (συνέχεια)

- **Παράκαμψη**
  - Όταν χρησιμοποιείτε το όνομα μιας μεταβλητής μέσα στη μέθοδο στην οποία έχει δηλωθεί
    - Η μεταβλητή έχει προτεραιότητα σε σχέση με οποιαδήποτε άλλη μεταβλητή με το ίδιο όνομα σε άλλη μέθοδο
  - **Σκίαση**: οι τοπικά δηλωμένες μεταβλητές πάντα παρακάμπτουν ή κρύβουν άλλες μεταβλητές με το ίδιο όνομα που υπάρχουν αλλού στην κλάση

Κατανόηση των ενοτήτων και των πεδίων εφαρμογής (συνέχεια)

```
public class OverridingVariable
{
   public static void main(String[] args)
   {
     int aNumber = 10;
     System.out.println("In main(), aNumber is " + aNumber);
     firstMethod();
     System.out.println("Back in main(), aNumber is " + aNumber);
     secondMethod(aNumber);
     System.out.println("Back in main() again, aNumber is " + aNumber);
   }
   public static void firstMethod()
   {
     int aNumber = 77;
     System.out.println("In firstMethod(), aNumber is "
        + aNumber);
   }
   public static void secondMethod(int aNumber)
   {
     System.out.println("In secondMethod(), at first " +
        "aNumber is " + aNumber);
     aNumber = 862;
     System.out.println("In secondMethod(), after an assignment " +
        "aNumber is " + aNumber);
   }
}
```

Το aNumber δηλώνεται στη main().

Όποτε χρησιμοποιείται το aNumber στη main(), διατηρεί την τιμή 10.

Αυτό το aNumber βρίσκεται σε άλλη διεύθυνση μνήμης από εκείνο της main(). Δηλώνεται τοπικά σ' αυτή τη μέθοδο.

Αυτό το aNumber επίσης βρίσκεται σε διαφορετική διεύθυνση μνήμης από εκείνο της main(). Δηλώνεται τοπικά σ' αυτή τη μέθοδο.

Εικόνα 4-6
Η κλάση OverridingVariable

Κατανόηση των ενοτήτων και των πεδίων εφαρμογής (συνέχεια)

Εικόνα 4-7

Έξοδος της εφαρμογής OverridingVariable

Υπερφόρτωση μεθόδων

- **Υπερφόρτωση**
  - Χρήση ενός όρου με διάφορα νοήματα
  - Σύνταξη πολλαπλών μεθόδων με το ίδιο όνομα αλλά διαφορετικές παραμέτρους
  - Ο μεταγλωττιστής «καταλαβαίνει το νόημα» με βάση τις παραμέτρους που χρησιμοποιούνται με την κλήση μεθόδου
  - Οι προγραμματιστές μπορούν να χρησιμοποιούν ένα μόνο κατανοητό όνομα
    - Για εργασίες που είναι λειτουργικά ίδιες

Υπερφόρτωση μεθόδων (συνέχεια)

```
public static void calculateInterest(double bal, double rate)
{
    double interest;
    interest = bal * rate;
    System.out.println("Simple interest on $" + bal +
        " at " + rate + "% rate is " + interest);
}
```

Εικόνα 4-12

Η μέθοδος calculateInterest() με δύο παραμέτρους double

Αυτόματος προβιβασμός τύπων σε κλήσεις μεθόδων

- Αν εφαρμογή περιέχει μόνο μία εκδοχή μεθόδου:

  Καλείτε τη μέθοδο χρησιμοποιώντας μια παράμετρο του σωστού τύπου δεδομένων ή μία που μπορεί να προβιβαστεί στον σωστό τύπο δεδομένων

  - Σειρά προβιβασμού:

    `double`, `float`, `long`, `int`

Αποφυγή ασάφειας

- **Απροσδιόριστη** κατάσταση
  - Όταν ο μεταγλωττιστής δεν μπορεί να αποφασίσει ποια μέθοδο να χρησιμοποιήσει
- Υπερφόρτωση μεθόδων
  - Ορθή παροχή διαφορετικών λιστών παραμέτρων για μεθόδους με το ίδιο όνομα
- Μη έγκυρες μέθοδοι
  - Μέθοδοι με παρόμοια ονόματα που έχουν παρόμοιες λίστες παραμέτρων αλλά διαφορετικούς τύπους επιστροφής

Δημιουργία και κλήση μεθόδων κατασκευής με παραμέτρους

- Η Java παρέχει αυτόματα μεθόδους κατασκευής όταν οι προκαθορισμένες μέθοδοι κατασκευής που δημιουργεί η κλάση δεν απαιτούν παραμέτρους
- Δημιουργία της δικής σας μεθόδου κατασκευής:
  - Εξασφαλίζει ότι τα πεδία μέσα στις κλάσεις αρχικοποιούνται με τις κατάλληλες προεπιλεγμένες τιμές
  - Οι μέθοδοι κατασκευής μπορούν να δέχονται παραμέτρους
    - Χρησιμοποιούνται για σκοπούς αρχικοποίησης κυρίως

Δημιουργία και κλήση μεθόδων κατασκευής με παραμέτρους (συνέχεια)

- Όταν γράφετε μια μέθοδο κατασκευής για μια κλάση, δεν έχετε πλέον στη διάθεσή σας την αυτόματα παρεχόμενη, προκαθορισμένη, μέθοδο κατασκευής
- Αν η μοναδική μέθοδος κατασκευής κάποιας κλάσης απαιτεί μια παράμετρο, πρέπει να παρέχετε την παράμετρο αυτή για κάθε αντικείμενο της κλάσης.

Υπερφόρτωση μεθόδων κατασκευής

- Χρησιμοποιείτε τις παραμέτρους των μεθόδων κατασκευής για να αρχικοποιήσετε τις τιμές των πεδίων και για οποιαδήποτε άλλη χρήση
- Αν οι λίστες παραμέτρων της μεθόδου κατασκευής διαφέρουν, δεν υπάρχει ασάφεια σχετικά με τη μέθοδο κατασκευής που πρέπει να κληθεί

Υπερφόρτωση Constructors (συνέχεια)

```
public class Employee
{
    private int empNum;
    Employee(int num)
    {
        empNum = num;
    }
    Employee()
    {
        empNum = 999;
    }
}
```

Εικόνα 4-22

Η κλάση Employee που περιέχει δύο μεθόδους κατασκευής

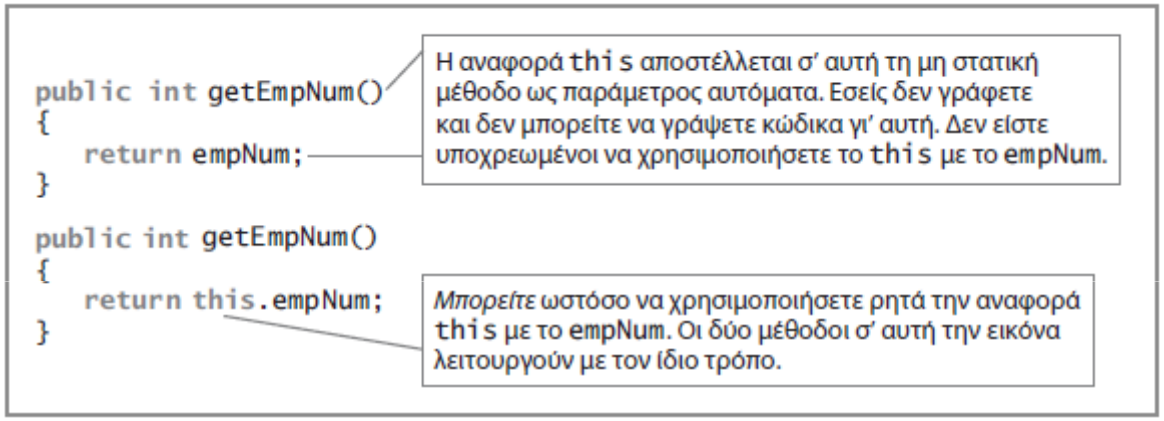
Υπερφόρτωση Constructors (συνέχεια)

- `Π.χ. Employee aWorker = new Employee();`
  - Καλείται η η εκδοχή της μεθόδου που δεν έχει παραμέτρους
- `Π.χ. Employee anotherWorker = new Employee(7677);`
  - Καλείται η εκδοχή της μεθόδου που απαιτεί ακέραια παράμετρο

Χρήση της αναφοράς `this`

- **Αναφορά `this`**
  - Η αναφορά σε ένα αντικείμενο
  - Περνά στη μη στατική μέθοδο κλάσης οποιουδήποτε αντικειμένου
  - Δεσμευμένη λέξη στη Java
- Δεν χρειάζεται να χρησιμοποιήσετε την αναφορά `this` σε μεθόδους που γράφετε στις περισσότερες περιπτώσεις

Χρήση της αναφοράς `this` (συνέχεια)



Εικόνα 4-24

Δύο εκδοχές της μεθόδου getEmpNum(), με και χωρίς ρητή αναφορά this

Χρήση της αναφοράς `this` (συνέχεια)

- **Αναφορά `this`** (συνέχεια)
  - Χρησιμοποιείται ώστε οι κλάσεις να λειτουργούν σωστά
  - Όταν χρησιμοποιείται μαζί με όνομα πεδίου σε μια μέθοδο κλάσης, η αναφορά γίνεται στο πεδίο της κλάσης και όχι στην τοπική μεταβλητή που δηλώνεται μέσα στη μέθοδο (ίδια ονόματα παραμέτρων και πεδίων αντικειμένων)
  - Π.χ. `stuNum = stuNum`

    `this.stuNum = stuNum`

Χρήση της αναφοράς `this` έτσι ώστε οι υπερφορτωμένες μέθοδοι κατασκευής να γίνουν πιο αποδοτικές

- Αποφυγή επαναλαμβανόμενου κώδικα σε μεθόδους κατασκευής
- Η μέθοδος κατασκευής καλεί άλλη μέθοδο κατασκευής
  - `this()`
  - Πιο αποδοτικός τρόπος και λιγότερο ευάλωτος σε λάθη

```
public class Student
{
    private int stuNum;
    private double gpa;
    Student(int num, double avg)
    {
       stuNum = num;
       gpa = avg;
    }
    Student(double avg)
    {
       stuNum = 999;
       gpa = avg;
    }
    Student(int num)
    {
       stuNum = num;
       gpa = 0.0;
    }
    Student()
    {
       stuNum = 999;
       gpa = 0.0;
    }
}
```

Κάθε μέθοδος κατασκευής περιέχει παρόμοιες εντολές.

Εικόνα 4-30
Η κλάση Student με τέσσερις μεθόδους κατασκευής

```
public class Student
{
   private int stuNum;
   private double gpa;
   Student(int num, double avg)
   {
      stuNum = num;
      gpa = avg;
   }
   Student(double avg)
   {
      this(999, avg);
   }
   Student(int num)
   {
      this(num, 0.0);
   }
   Student()
   {
      this(999, 0.0);
   }
}
```

Καθεμία απ' αυτές τις κλήσεις στο `this()` καλεί την εκδοχή δύο παραμέτρων της μεθόδου κατασκευής.

Εικόνα 4-31
Η κλάση Student όταν χρησιμοποιεί το this σε τρεις από τις τέσσερις μεθόδους κατασκευής

Χρήση σταθερών πεδίων

- Δημιουργείτε επώνυμες σταθερές χρησιμοποιώντας τη δεσμευμένη λέξη `final`
  - Η τιμή τους πρέπει να είναι σταθερή μετά την κατασκευή
- Μπορεί να οριστεί στη μέθοδο κατασκευής κλάσης
  - Μετά την κατασκευή, δεν μπορείτε να αλλάξετε την τιμή του πεδίου `final`

Χρήση σταθερών πεδίων (συνέχεια)

```
public class Student
{
    private static final int SCHOOL_ID = 12345;
    private int stuNum;
    private double gpa;
    public Student(int stuNum, double gpa)
    {
        this.stuNum = stuNum;
        this.gpa = gpa;
    }
    public void showStudent()
    {
        System.out.println("Student #" + stuNum +
            " gpa is " + gpa);
    }
}
```

Εικόνα 4-35

Η κλάση Student που περιέχει μια συμβολική σταθερά

Χρήση αυτόματα εισαγόμενων, έτοιμων σταθερών και μεθόδων

- Πολλές κλάσεις χρησιμοποιούνται από κοινού από πολλούς προγραμματιστές
- **Πακέτο** ή **βιβλιοθήκη κλάσεων**
  - Ένας φάκελος που προσφέρεται για πρακτική ομαδοποίηση των κλάσεων αυτών
  - Πολλά περιέχουν κλάσεις που είναι διαθέσιμες μόνο αν κατονομάζονται ρητά μέσα σε ένα πρόγραμμα
  - Κάποιες κλάσεις διατίθενται αυτόματα

Χρήση αυτόματα εισαγόμενων, έτοιμων σταθερών και μεθόδων (συνέχεια)

- Πακέτο **`java.lang`**
  - Εισάγεται αυτόματα σε κάθε πρόγραμμα
  - Το μόνο αυτόματα εισαγόμενο, επώνυμο πακέτο
  - Οι κλάσεις που περιέχει είναι **θεμελιώδεις κλάσεις** (ή βασικές κλάσεις)
- **Προαιρετικές κλάσεις (optional classes)**
  - Πρέπει να κατονομάζονται ρητά
  - π.χ. Πακέτο Javax.swing για την κλάση JOptionPane

Χρήση αυτόματα εισαγόμενων, έτοιμων σταθερών και μεθόδων (συνέχεια)

- Κλάση `java.lang.Math`
  - Περιέχει σταθερές και μεθόδους τις οποίες μπορείτε να χρησιμοποιείτε για να υπολογίζετε κοινές μαθηματικές συναρτήσεις
  - Δεν χρειάζεται να δημιουργείτε αντικείμενα
  - Εισάγονται αυτόματα ή
  - π.χ. σταθερά `PI` (π=3.14159265358979323846)

```
areaOfCircle = java.lang.Math.PI *
radius *radius
```

ή

```
areaOfCircle = Math.PI * radius
*radius
```

Μέθοδος	Τιμή που επιστρέφει η μέθοδος
`abs(x)`	Απόλυτη τιμή του *x*
`acos(x)`	Τόξο συνημιτόνου *x*
`asin(x)`	Τόξο ημιτόνου *x*
`atan(x)`	Τόξο εφαπτομένης *x*
`atan2(x, y)`	Η συνιστώσα *θ* της πολικής συντεταγμένης coordinate (*r*, *θ*) που αντιστοιχεί στην Καρτεσιανή συντεταγμένη *x*, *y*
`ceil(x)`	Η μικρότερη ακέραια τιμή που δεν είναι μικρότερη από το *x* (ο επόμενος μεγαλύτερος ακέραιος αριθμός)
`cos(x)`	Συνημίτονο του *x*
`exp(x)`	Εκθέτης, όπου το *x* είναι η βάση των φυσικών λογαρίθμων
`floor(x)`	Η μεγαλύτερη ακέραια τιμή που δεν είναι μεγαλύτερη από το *x* (το ακέραιο μέρος του *x*)
`log(x)`	Φυσικός λογάριθμος του *x*
`max(x, y)`	Το μεγαλύτερο από τα *x* και *y*
`min(x, y)`	Το μικρότερο από τα *x* και *y*
`pow(x, y)`	Το *x* υψωμένο στη δύναμη *y*
`random()`	Τυχαίος double αριθμός μεταξύ 0,0 και 1,0
`rint(x)`	Ο εγγύτερος ακέραιος στο *x* (το *x* είναι double και η τιμή επιστροφής εκφράζεται ως double)
`round(x)`	Ο εγγύτερος ακέραιος στο *x* (όπου το *x* είναι float ή double και η τιμή επιστροφής είναι int ή long)
`sin(x)`	Ημίτονο του *x*
`sqrt(x)`	Τετραγωνική ρίζα του *x*
`tan(x)`	Εφαπτομένη του *x*

Πίνακας 4-1 Κοινές μέθοδοι της κλάσης Math

Εισαγωγή κλάσεων που δεν εισάγονται αυτόματα

- Χρήση έτοιμων κλάσεων
  - Χρήση ολόκληρου του μονοπατιού με το όνομα της κλάσης
  - Εισαγωγή της κλάσης
  - Εισαγωγή του πακέτου που περιέχει την κλάση την οποία χρησιμοποιείτε

Εισαγωγή κλάσεων που δεν εισάγονται αυτόματα (συνέχεια)

- **Σύμβολο μπαλαντέρ (*)**
  - Εναλλακτική επιλογή για την εισαγωγή της κλάσης
    - Εισαγωγή ολόκληρου του πακέτου κλάσεων
  - Μπαλαντέρ αστερίσκος
    - Μπορεί να αντικαταστήσει οποιοδήποτε σύνολο χαρακτήρων
    - Αναπαριστά όλες τις κλάσεις σε ένα πακέτο
  - Π.χ. `Import java.time.LocalDate;`
  - `Import java.time.*;` (εισάγει όλες τις κλάσεις ενός πακέτου αλλά ΟΧΙ όλα τα πακέτα)

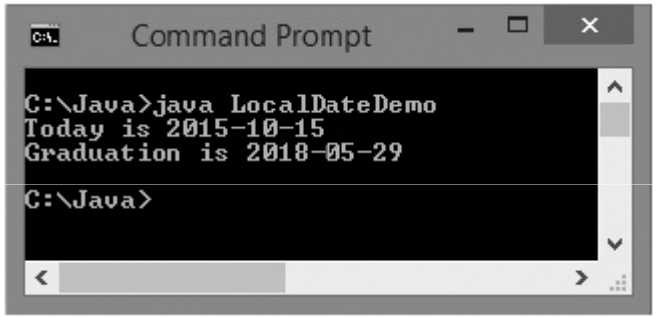
Χρήση της κλάσης `LocalDate`

- Κλάση `LocalDate`
  - Σ’ αυτήν την κλάση δεν περιλαμβάνονται πληροφορίες ζώνης ώρας (μόνο τοπική ημερομηνία)
  - Στατικές μέθοδοι `now()` και `of()`
  - Οι μέθοδοι κατασκευής κλάσεων δεν είναι public
  - Παρέχονται διάφορες μέθοδοι για την εκτέλεση πράξεων με ημερομηνίες
- Απαρίθμηση
  - Τύπος δεδομένων που αποτελείται από μια λίστα τιμών, όπως
    - JANUARY, FEBRUARY, MARCH κ.λπ..

Χρήση της κλάσης `LocalDate` (συνέχεια)

```
import java.time.*;
public class LocalDateDemo
{
    public static void main(String[] args)
    {
        LocalDate today = LocalDate.now();
        LocalDate graduationDate = LocalDate.of(2018, 5, 29);
        System.out.println("Today is " + today);
        System.out.println("Graduation is " + graduationDate);
    }
}
```

Εικόνα 4-37
Η εφαρμογή LocalDateDemo



Εικόνα 4-38
Εκτέλεση της εφαρμογής LocalDateDemo

Χρήση της κλάσης `LocalDate` (συνέχεια)

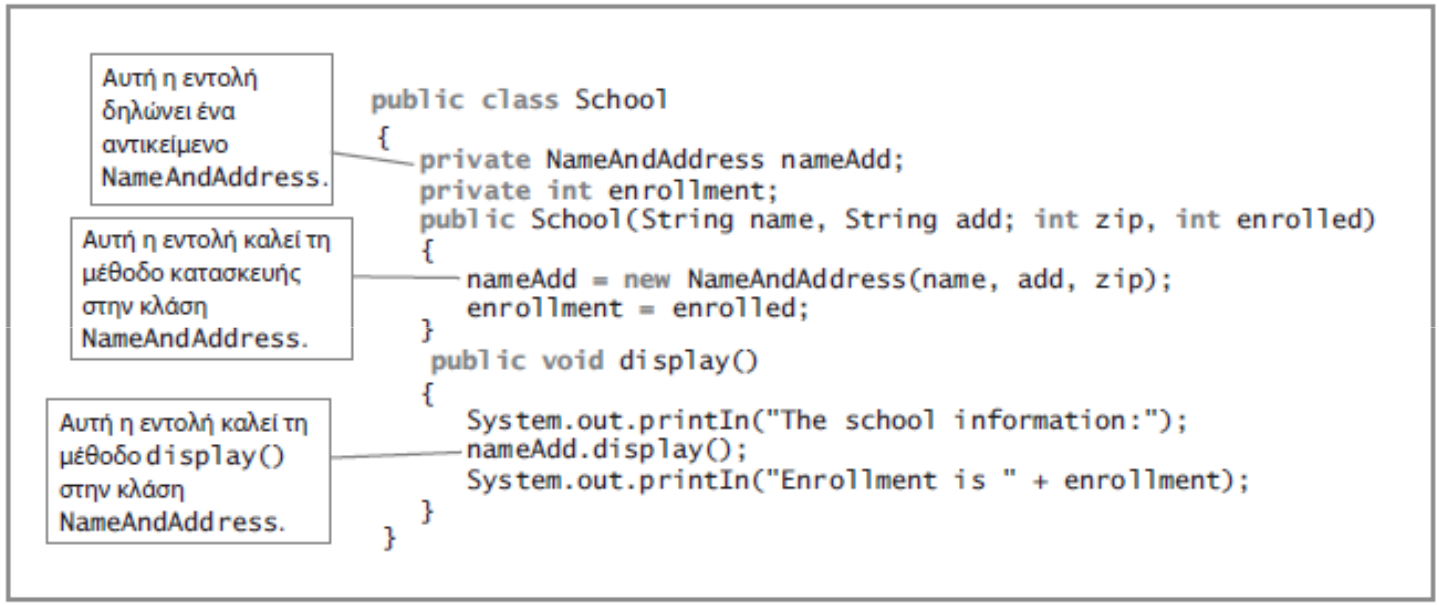
```
import java.time.*;
import java.util.Scanner;
public class DeliveryDate
{
    public static void main(String[] args)
    {
        Scanner input = new Scanner(System.in);
        LocalDate orderDate;
        int mo;
        int day;
        int year;
        final int WEEKS_FOR_DELIVERY = 2;
        System.out.print("Enter order month ");
        mo = input.nextInt();
        System.out.print("Enter order day ");
        day = input.nextInt();
        System.out.print("Enter order year ");
        year = input.nextInt();
        orderDate = LocalDate.of(year, mo, day);
        System.out.println("Order date is " + orderDate);
        System.out.println("Delivery date is " +
            orderDate.plusWeeks(WEEKS_FOR_DELIVERY));
    }
}
```

Εικόνα 4-39
Η εφαρμογή DeliveryDate

Χρήση σύνθεσης και ένθεσης κλάσεων

- **Σύνθεση**
  - Περιγράφει τη σχέση ανάμεσα στις κλάσεις όταν το αντικείμενο μίας κλάσης είναι πεδίο δεδομένων σε άλλη κλάση
  - Ονομάζεται **σχέση has-a (έχει ένα)**
    - Επειδή μία κλάση "έχει ένα" στιγμιότυπο άλλης
- Μην ξεχνάτε να παρέχετε τιμές για το περιεχόμενο αντικείμενο, αν αυτό δεν έχει προκαθορισμένη μέθοδο κατασκευής

Χρήση σύνθεσης και ένθεσης κλάσεων (συνέχεια)



Εικόνα 4-43
Η κλάση School

Ένθετες κλάσεις

- **Ένθετες κλάσεις**
  - Μια κλάση μέσα σε άλλη κλάση
  - Αποθηκεύονται μαζί σε ένα αρχείο
- Ένθετοι τύποι κλάσεων
  - **`static` κλάσεις-μέλη**
  - **Μη στατικές κλάσεις-μέλη**
  - **Τοπικές κλάσεις**
  - **Ανώνυμες κλάσεις**

Περίληψη

- Πεδίο εφαρμογής μεταβλητής
  - Το τμήμα ενός προγράμματος μέσα στο οποίο μπορεί να γίνει αναφορά σε μια μεταβλητή
- Ενότητα
  - Κώδικας μεταξύ δύο αγκίστρων
- Υπερφόρτωση
  - Δημιουργία πολλαπλών μεθόδων με το ίδιο όνομα αλλά διαφορετικές λίστες παραμέτρων
- Αποθήκευση διαφορετικών αντίγραφο πεδίων δεδομένων για κάθε αντικείμενο
  - Αλλά μόνο ένα αντίγραφο κάθε μεθόδου

Περίληψη (συνέχεια)

- `static` μεταβλητές κλάσεων
  - Χρησιμοποιούνται από κοινού από κάθε στιγμιότυπο μιας κλάσης
- Έτοιμες κλάσεις
  - Αποθηκεύονται σε πακέτα
- Εντολή `import`
  - Ειδοποιεί το πρόγραμμα Java ότι τα ονόματα κλάσεων αναφέρονται σε εκείνα μέσα στην εισηγμένη κλάση
- Μια κλάση μπορεί να περιέχει άλλα αντικείμενα ως μέλη δεδομένων
- Μπορείτε να δημιουργείτε ένθετες κλάσεις που αποθηκεύονται στο ίδιο αρχείο