Ενότητα 2: **Διαχείριση Μνήμης** και Σχεδιασμός Κλάσεων

Μάθημα 12: 1/3/2024

Στατικές μεταβλητές και Μέθοδοι
Ανασκόπηση Αρχικοποιήσεων
Περιβάλλουσες Κλάσεις
Ισότητα Αντικειμένων
Αναλλοίωτοι Περιορισμοί

University of Cyprus
Department of Computer Science

Περίγραμμα

University of Cyprus
Department of Computer Science

Ενότητα 2: Διαχείριση Μνήμης και **Σχεδιασμός Κλάσεων**

Στατικές μεταβλητές και Μέθοδοι

Κεφ. 5, Savitch

University of Cyprus
Department of Computer Science

Η χρήση του static

- Για την πρόσβαση σε δεδομένα και μεθόδους προϋποτίθεται η δημιουργία σχετικού αντικειμένου.
- Υπάρχουν δύο περιπτώσεις όπου αυτό **δεν** είναι επιθυμητό:
  - Αν θέλουμε να χρησιμοποιείται **μόνο μια θέση αποθήκευσης** για κάποιο συγκεκριμένο δεδομένο, ανεξάρτητα του πόσα αντικείμενα δημιουργούνται.
  - Αν χρειαζόμαστε μεθόδους που **δεν** αντιστοιχούν σε κάποιο συγκεκριμένο αντικείμενο.
  - Δηλ., αν θέλουμε να καλούμε μια μέθοδο χωρίς να έχουμε δημιουργήσει αντικείμενα.
- Οι δυνατότητες αυτές μπορούν να υλοποιηθούν με την χρήση της λέξης κλειδί **static.**
- Δεδομένα και μέθοδοι που ορίζονται σαν **static** αποκαλούνται και **class-data, class-methods**.

Στατικές Μέθοδοι

- Στατικές μέθοδοι είναι αυτές που μπορεί να κληθούν **χωρίς αντικείμενο κλήσης**.
- Μια στατική μέθοδος εξακολουθεί να ανήκει σε μια κλάση και ο ορισμός της δίνεται μέσα στο σώμα της κλάσης
- Όταν δηλώνουμε μια στατική μέθοδο, η λέξη-κλειδί `static` τοποθετείται στην επικεφαλίδα της μεθόδου

```
public static returnedType myMethod(parameters){
  . . . }
```

- Οι στατικές μέθοδοι καλούνται χρησιμοποιώντας το όνομα της κλάσης στη θέση καλούντος αντικειμένου:

```
returnedValue = MyClass.myMethod(arguments);
```

 Copyright © 2017 Pearson Ltd. All rights reserved.

Στο σώμα στατικής μεθόδου:

- Δεν επιτρέπεται να:
  - χρησιμοποιηθούν μεταβλητές στιγμιότυπου της κλάσης στην οποία ανήκει η μέθοδος
  - γίνει κλήση μη στατικών μεθόδων της κλάσης
- *Γιατί;*
- Μια στατική μέθοδος δεν έχει `this`, επομένως δεν μπορεί να χρησιμοποιήσει στοιχεία η πρόσβαση προς τα οποία γίνεται από ρητό ή υπόρρητο `this`
- Ωστόσο, μια στατική μέθοδος μπορεί να κάνει κλήση άλλης στατικής μεθόδου.

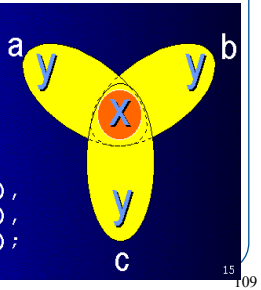
 Copyright © 2017 Pearson Ltd. All rights reserved.

Static Variables (στατικές μεταβλητές)

- Μια στατική μεταβλητή (static variable) ανήκει-αποθηκεύεται στην κλάση στην οποία έχει δηλωθεί:
  - Για κάθε στατική μεταβλητή υπάρχει ένα μοναδικό αντίγραφο στην κλάση της.
  - Αντίθετα, για μια μεταβλητή στιγμιοτύπου, κάθε αντικείμενο κρατάει το δικό του αντίγραφο.
- Όλα τα αντικείμενα μιας
  και να τροποποιήσουν τι
  κλάσης.
- Οι στατικές μεταβλητές ε
  από τις στατικές μεθόδου

```
class WithStaticData {
  static int x;
  int y;
}

WithStaticData
  a = new WithStaticData(),
  b = new WithStaticData(),
  c = new WithStaticData();
```



Στατικές Μεταβλητές

- Οι στατικές μεταβλητές μπορούν να δηλωθούν και να αρχικοποιηθούν ταυτόχρονα:

```
private static int myStaticVariable = 0;
```

- Εάν δεν αρχικοποιηθεί ρητά, μια στατική μεταβλητή θα αρχικοποιηθεί αυτόματα σε μια προκαθορισμένη τιμή (default value)
  - `boolean` -> `false`
  - Other primitive types -> `zero` of their type
  - Class type static variables -> `null`
- *Είναι προτιμότερο να αρχικοποιείτε ρητά τις στατικές μεταβλητές αντί να βασίζεστε στην προκαθορισμένη αρχικοποίηση τους.*

 Copyright © 2017 Pearson Ltd. All rights reserved.

Σταθερές στη Java

- Μια στατική μεταβλητή πρέπει να ορίζεται πάντοτε ως `private`, εκτός εάν είναι σταθερά.
- Οι **στατικές σταθερές** ορίζονται με χρήση του προσδιοριστή (modifier) `final` που υποδηλώνει ότι η τιμή της δεν αλλάζει:

```
public static final int BIRTH_YEAR = 1954;
```

- Η τιμή μιας στατικής σταθεράς δεν αλλάζει:
  - Επομένως, είναι ασφαλές να δηλωθεί ως `public`
- Όταν αναφέρεστε σε μια στατική σταθερά εκτός του σώματος της κλάσης της, χρησιμοποιήστε το όνομα της κλάσης της στη θέση καλούντος αντικειμένου

```
int year = MyClass.BIRTH_YEAR;
```


Copyright © 2017 Pearson Ltd. All rights reserved.

Static (παραδείγματα)

```
class StaticTest {
        private static int i = 47;
}
StaticTest st1 = new StaticTest();
StaticTest st2 = new StaticTest();
```

- st1.i και st2.i δείχνουν στην ίδια τιμή.
- Αναφορά σε στατικές μεταβλητές:
  - είτε μέσω αντικειμένου
  - είτε μέσω κλάσης: StaticTest.i ++;

```
class StaticFun {
        static void incr() {StaticTest.i ++; }
}
StaticFun sf = new StaticFun();
sf.incr();
StaticFun.incr();
```


Ενότητα 2: Διαχείριση Μνήμης και Σχεδιασμός Κλάσεων

Ανασκόπηση Αρχικοποιήσεων

University of Cyprus
Department of Computer Science

Σειρά Αρχικοποιήσεων

- *Μέσα σε μια κλάση*, η σειρά των αρχικοποιήσεων καθορίζεται από τη σειρά δήλωσης των πεδίων δεδομένων της κλάσης.
- Ακόμη κι αν οι αρχικοποιήσεις είναι διεσπαρμένες ανάμεσα σε δηλώσεις μεθόδων, τα πεδία θα αρχικοποιηθούν πριν την κλήση οποιασδήποτε μεθόδου, **ακόμη και του constructor.**
- Πότε γίνεται η αρχικοποίηση **στατικών μεταβλητών;**
  - Μόνο όταν αυτό καταστεί αναγκαίο, είτε λόγω δημιουργίας του πρώτου σχετικού αντικειμένου, είτε λόγω κλήσης κάποιας στατικής μεθόδου της αντίστοιχης κλάσης.

Ανασκόπηση Δημιουργίας Αντικειμένου

- Έστω ότι έχουμε ορίσει μια κλάση Dog.
- Την πρώτη φορά που δημιουργείται ένα αντικείμενο Dog, ή την πρώτη φορά που καλείται μια **στατική** μέθοδος της Dog ή γίνεται πρόσβαση σε ένα **στατικό** πεδίο της κλάσης Dog, ο διερμηνέας της Java πρέπει να βρεί την κλάση **Dog.class**, την οποία αναζητεί με την βοήθεια του **`classpath`.**
- Καθώς η **Dog.class** φορτώνεται (οπότε και *δημιουργείται ένα αντικείμενο* ***Class***), εκτελούνται όλοι οι **στατικοί αρχικοποιητές** (static initializers). Επομένως η στατική αρχικοποίηση συμβαίνει μόνο μια φορά, όταν το **class object** φορτώνεται για πρώτη φορά.

Ανασκόπηση Δημιουργίας Αντικειμένου

- Όταν δημιουργηθεί αντικείμενο Dog με την **`new Dog()`**, η διαδικασία κατασκευής δεσμεύει πρώτα αρκετό χώρο στον σωρό.
- Ο χώρος που κρατήθηκε αρχικοποιείται με μηδενικά.
- Εκτελούνται οι οποιεσδήποτε αρχικοποιήσεις έχουν δηλωθεί.
- Εκτελούνται οι constructors τού αντικειμένου.
  - Στο σημείο αυτό μπορούμε να έχουμε αρκετή δραστηριότητα, ιδιαίτερα όταν η κλάση μας κληρονομεί χαρακτηριστικά άλλων κλάσεων.

Ρητή στατική αρχικοποίηση

- Μπορούμε να συγκεντρώσουμε τις στατικές αρχικοποιήσεις (static initializations) μέσα σε ένα ειδικό «στατικό πλαίσιο» (static block), στο εσωτερικό μιας κλάσης.

```
public class Spoon {
      static int i;
      static {
            i = 47;
      }
}
```

```
class Cups {
      static Cup cup1;
      static Cup cup2;
      static {
            cup1 = new Cup(1);
            cup2 = new Cup(2);
      }
}
```

- Οι ρητές στατικές αρχικοποιήσεις εκτελούνται μαζί με τους στατικούς αρχικοποιητές.

Μη στατικοί αρχικοποιητές

- Non-static instance initialization: παρόμοια σύνταξη με τους ρητούς στατικούς αρχικοποιητές:
- Π.χ.:

```
public class Mugs {
  Mug mug1;
  Mug mug2;
  {
   mug1 = new Mug(1);
   mug2 = new Mug(2);
   System.out.println("Instance Initialization");
  }
...
}
```

Ενότητα 2: Διαχείριση Μνήμης και Σχεδιασμός Κλάσεων

Παραδείγματα χρήσης static

University of Cyprus
Department of Computer Science

ΠΩΣ ΜΠΟΡΟΥΜΕ ΝΑ ΚΡΑΤΑΜΕ ΕΝΑ ΜΕΤΡΗΤΗ ΤΩΝ ΑΝΤΙΚΕΙΜΕΝΩΝ ΤΥΠΟΥ ΦΟΡΤΙΟ ΠΟΥ ΔΗΜΙΟΥΡΓΟΥΝΤΑΙ ΚΑΤΑ ΤΗΝ ΕΚΤΕΛΕΣΗ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ ΜΑΣ;

University of Cyprus
Department of Computer Science

M. D. Dikaiakos

Υλοποίηση Κλάσης Charge

```
public class Charge {
    private double rx, ry;   // position
    private double q;        // charge
    private static int counter=0;    // charge counter

    public Charge(double x0, double y0, double q0) {
        rx = x0;
        ry = y0;
        q  = q0;
        counter += 1;
    }
    public double potentialAt(double x, double y) {
      ...
    }

    public String toString() {
        return q + “number “ + counter +
              " at " + "(" + rx + ", " + ry + ")";
    } }
```


Η Κλάση `Math`

- Η κλάση **`Math`** παρέχει έναν αριθμό γνωστών μαθηματικών μεθόδων
- Βρίσκεται στο πακέτο **`java.lang`**, επομένως δεν απαιτεί δήλωση εισαγωγής **`import`**
- Όλες οι μέθοδοι και τα δεδομένα του είναι στατικά, επομένως καλούνται με το όνομα της κλάσης **`Math`**
- Η **`Math`** έχει δύο προκαθορισμένες σταθερές, την **`E`** (e, η βάση του φυσικού λογαρίθμου συστήματος) και την **`PI`** (π, 3.1415 …)

```
area = Math.PI * radius * radius;
```


Copyright © 2017 Pearson Ltd. All rights reserved.

Μέθοδοι της Math

Display 5.6 **Some Methods in the Class Math**

The Math class is in the java.lang package, so it requires no import statement.

```
public static double pow(double base, double exponent)
```

Returns base to the power exponent.

EXAMPLE

Math.pow(2.0,3.0) returns 8.0.

(continued)

Copyright © 2017 Pearson Ltd. All rights reserved.

Μέθοδοι της Math

Display 5.6 **Some Methods in the Class Math**

```
public static double abs(double argument)
public static float abs(float argument)
public static long abs(long argument)
public static int abs(int argument)
```

Returns the absolute value of the argument. (The method name abs is overloaded to produce four similar methods.)

EXAMPLE

Math.abs(−6) and Math.abs(6) both return 6. Math.abs(−5.5) and Math.abs(5.5) both return 5.5.

```
public static double min(double n1, double n2)
public static float min(float n1, float n2)
public static long min(long n1, long n2)
public static int min(int n1, int n2)
```

Returns the minimum of the arguments n1 and n2. (The method name min is overloaded to produce four similar methods.)

EXAMPLE

Math.min(3, 2) returns 2.

Μέθοδοι της Math

Display 5.6 **Some Methods in the Class Math**

```
public static double max(double n1, double n2)
public static float max(float n1, float n2)
public static long max(long n1, long n2)
public static int max(int n1, int n2)
```

Returns the maximum of the arguments n1 and n2. (The method name max is overloaded to produce four similar methods.)

EXAMPLE

Math.max(3, 2) returns 3.

```
public static long round(double argument)
public static int round(float argument)
```

Rounds its argument.

EXAMPLE

Math.round(3.2) returns 3; Math.round(3.6) returns 4.

(continued)

 Copyright © 2017 Pearson Ltd. All rights reserved.

Μέθοδοι της Math

Display 5.6 **Some Methods in the Class Math**

```
public static double ceil(double argument)
```

Returns the smallest whole number greater than or equal to the argument.

EXAMPLE

Math.ceil(3.2) and Math.ceil(3.9) both return 4.0.

(continued)

Copyright © 2017 Pearson Ltd. All rights reserved.

Μέθοδοι της Math

Display 5.6 **Some Methods in the Class Math**

`public static double floor(double argument)`

Returns the largest whole number less than or equal to the argument.

EXAMPLE

`Math.floor(3.2)` and `Math.floor(3.9)` both return `3.0`.

`public static double sqrt(double argument)`

Returns the square root of its argument.

EXAMPLE

`Math.sqrt(4)` returns `2.0`.

Copyright © 2017 Pearson Ltd. All rights reserved.

Τυχαίοι Αριθμοί

- Η κλάση **`Math`** παρέχει επίσης τη δυνατότητα δημιουργίας ψευδοτυχαίων αριθμών

```
public static double random()
```

- Ένας ψευδοτυχαίος αριθμός εμφανίζεται σαν τυχαίος αλλά στην πραγματικότητα δημιουργείται από μια ντετερμινιστική συνάρτηση:

```
double num = Math.random();
```

- Επιστρέφει έναν ψευδοτυχαίο αριθμό μεγαλύτερο ή ίσο με 0,0 και μικρότερο από 1,0
- Περισσότερες δυνατότητες για τη δημιουργία τυχαίων αριθμών μας παρέχει η κλάση **`Random`**.

Copyright © 2017 Pearson Ltd. All rights reserved.

Βρείτε τυχαίο ακέραιο μεταξύ 0 - N

```
public class RandomInteger {
    public static int printRandInt(int max) {

        // a pseudo-random real between 0.0 and 1.0
        double r = Math.random();

        // a pseudo-random integer between 0 and max-1
        int n = (int) (r * max);

        return n;
    }
}
```


Περίγραμμα

University of Cyprus
Department of Computer Science

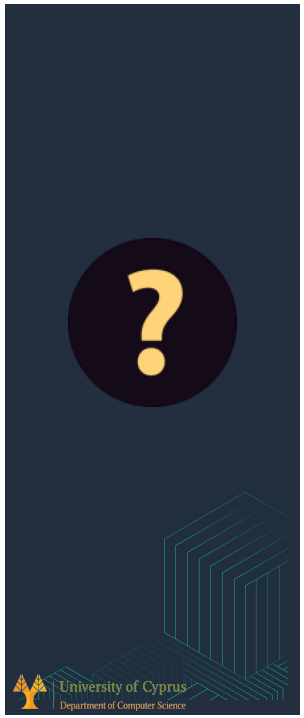
Ενότητα 2: Διαχείριση Μνήμης και Σχεδιασμός Κλάσεων

Περιβάλλουσες Κλάσεις

University of Cyprus
Department of Computer Science

Περιβάλλουσες κλάσεις (Wrapper Classes)

- Οι περιβάλλουσες κλάσεις (wrapper classes) παρέχουν τύπους κλάσης (class types) που αντιστοιχούν σε καθέναν από τους αρχέγονους τύπους.
  - Αυτό καθιστά δυνατή την ύπαρξη τύπων κλάσεων που συμπεριφέρονται σαν αρχέγονοι τύποι:
    - `byte`, `short`, `long`, `float`, `double`, και `char`:
    - `Byte`, `Short`, `Long`, `Float`, `Double`, κ. `Character`
- Οι περιβάλλουσες κλάσεις περιέχουν επίσης έναν αριθμό από χρήσιμες προκαθορισμένες σταθερές (predefined constants) και στατικές μεθόδους (static methods).



ΠΩΣ ΜΕΤΑΦΡΑΖΟΥΜΕ ΜΙΑ ΤΙΜΗ ΑΡΧΕΓΟΝΟΥ ΤΥΠΟΥ ΣΕ ΤΙΜΗ ΠΕΡΙΒΑΛΛΟΥΣΑΣ ΚΛΑΣΗΣ ΚΑΙ ΠΙΣΩ;

Εγκιβωτισμός

- Boxing (συσκευασία - εγκιβωτισμός): η διαδικασία μετάβασης από μια τιμή αρχέγονου τύπου σε ένα αντικείμενο της περιβάλλουσας κλάσης του
- Για να μετατρέψετε μια αρχέγονη τιμή σε μια ισοδύναμη τιμή τύπου περιβάλλουσας κλάση, δημιουργήστε ένα αντικείμενο της κλάσης με την αρχέγονη τιμή ως όρισμα:
  - `Integer integerObject = new Integer(42);`
  - Το νέο αντικείμενο θα περιέχει μια μεταβλητή στιγμιότυπου που αποθηκεύει ένα αντίγραφο της αρχέγονης τιμής.
  - Σε αντίθεση με τις περισσότερες κλάσεις, *μια περιβάλλουσα κλάση δεν έχει κατασκευαστή χωρίς όρισμα.*

Αποκιβωτισμός

- Unboxing (αποσυσκευασία - αποκιβωτισμός): η διαδικασία μετάβασης από ένα αντικείμενο περιβάλλουσας κλάσης στην αντίστοιχη τιμή ενός πρωτόγονου τύπου

Αρχέγονος τύπος	Μέθοδος αποκιβωτισμού
`Byte`	`byteValue()`
`Short`	`shortValue()`
`Integer`	`intValue()`
`Long`	`longValue()`
`Float`	`floatValue()`
`Double`	`doubleValue()`
`Character`	`charValue()`

- Καμία από αυτές τις μεθόδους δεν δέχεται όρισμα

```
int i = integerObject.intValue();
```


Αυτόματος Εγκιβωτισμός / Αποκιβωτισμός

- Ξεκινώντας με την έκδοση 5.0, η Java μπορεί να κάνει αυτόματα **εγκιβωτισμό-boxing** και **αποκιβωτισμό-unboxing**.
- Αντί να δημιουργήσετε ένα αντικείμενο περιβάλλουσας κλάσης χρησιμοποιώντας τον τελεστή new, μπορεί να γίνει αυτόματος μετασχηματισμός τύπου:
  - `Integer integerObject = 42;`
- Αντί να καλέσετε κατάλληλη μέθοδο (`intValue`, `doubleValue`, `charValue`, etc.) προκειμένου να μετατραπεί ένα αντικείμενο περιβάλλουσας κλάσης σε μια τιμή του συσχετιζόμενου αρχέγονου τύπου, η τιμή μπορεί να ανακτηθεί αυτόματα μέσα από το αντικείμενο:
  - `int i = integerObject;`

Σταθερές σε Wrapper Classes

- Οι περιβάλλουσες κλάσεις περιλαμβάνουν χρήσιμες σταθερές που παρέχουν τις μεγαλύτερες και τις μικρότερες τιμές για οποιονδήποτε από τους αρχέγονους τύπους αριθμών:
  - `Integer.MAX_VALUE, Integer.MIN_VALUE, Double.MAX_VALUE, Double.MIN_VALUE`, κλπ.
  - `Boolean.TRUE` and `Boolean.FALSE`

Στατικές Μέθοδοι σε Wrapper Classes

- Οι περιβάλλουσες κλάσεις διαθέτουν στατικές μεθόδους που μετατρέπουν μια **σωστά διαμορφωμένη αναπαράσταση συμβολοσειράς** (string) ενός αριθμού στον αριθμό του κατάλληλου τύπου
- Οι μέθοδοι `Integer.parseInt, Long.parseLong, Float.parseFloat,` και `Double.parseDouble` επιστρέφουν (κατά σειρά): `int, long, float,` και `double` Πχ:
  - `Double.parseDouble("123.99");` επιστρέφει: `123.99`
  - `Double.parseDouble(" 123.99 ".trim());`
- Έχουν επίσης στατικές μεθόδους που μετατρέπουν μια αριθμητική τιμή σε μια συμβολοσειρά αναπαράστασης της τιμής. Π.χ.
  - `Double.toString(123.99);` επιστρέφει `"123.99"`

Some Methods in the Class `Character` (Part 1 of 3)

- The Character class contains a number of static methods that are useful for string processing

Display 5.8 Some Methods in the Class Character

The class `Character` is in the `java.lang` package, so it requires no `import` statement.

`public static char toUpperCase(char argument)`

Returns the uppercase version of its `argument`. If the `argument` is not a letter, it is returned unchanged.

EXAMPLE

`Character.toUpperCase('a')` and `Character.toUpperCase('A')` both return `'A'`.

`public static char toLowerCase(char argument)`

Returns the lowercase version of its `argument`. If the `argument` is not a letter, it is returned unchanged.

EXAMPLE

`Character.toLowerCase('a')` and `Character.toLowerCase('A')` both return `'a'`.

`public static boolean isUpperCase(char argument)`

Returns `true` if its `argument` is an uppercase letter; otherwise returns `false`.

EXAMPLE

`Character.isUpperCase('A')` returns `true`. `Character.isUpperCase('a')` and `Character.isUpperCase('%')` both return `false`.

Some Methods in the Class `Character` (Part 2 of 3)

Display 5.8 Some Methods in the Class Character

`public static boolean isLowerCase(char argument)`

Returns `true` if its `argument` is a lowercase letter; otherwise returns `false`.

EXAMPLE

`Character.isLowerCase('a')` returns `true`. `Character.isLowerCase('A')` and `Character.isLowerCase('%')` both return `false`.

`public static boolean isWhitespace(char argument)`

Returns `true` if its `argument` is a whitespace character; otherwise returns `false`. Whitespace characters are those that print as white space, such as the space character (blank character), the tab character (`'\t'`), and the line break character (`'\n'`).

EXAMPLE

`Character.isWhitespace(' ')` returns `true`. `Character.isWhitespace('A')` returns `false`.

(continued)

Copyright © 2017 Pearson Ltd. All rights reserved.

Some Methods in the Class `Character` (Part 3 of 3)

Display 5.8 Some Methods in the Class Character

`public static boolean isLetter(char argument)`

Returns `true` if its `argument` is a letter; otherwise returns `false`.

EXAMPLE

`Character.isLetter('A')` returns `true`. `Character.isLetter('%')` and `Character.isLetter('5')` both return `false`.

`public static boolean isDigit(char argument)`

Returns `true` if its `argument` is a digit; otherwise returns `false`.

EXAMPLE

`Character.isDigit('5')` returns `true`. `Character.isDigit('A')` and `Character.isDigit('%')` both return `false`.

`public static boolean isLetterOrDigit(char argument)`

Returns `true` if its `argument` is a letter or a digit; otherwise returns `false`.

EXAMPLE

`Character.isLetterOrDigit('A')` and `Character.isLetterOrDigit('5')` both return `true`. `Character.isLetterOrDigit('&')` returns `false`.

Περίγραμμα

University of Cyprus
Department of Computer Science

- Οργάνωση και Διαχείριση Μνήμης
- Διεργασίες και Εικονική Μνήμη
- Διαχείριση Μνήμης Java κ. JVM
- Αρχικοποιήσεις και Κατασκευή Αντικειμένων
- Σκύβαλα και Αποκομιδή
- Στατικές Μεταβλητές και Μέθοδοι
- Περιβάλλουσες Κλάσεις
- **Έλεγχος ισότητας αντικειμένων**
- Αναλλοίωτοι Περιορισμοί
- Παραβίαση ιδιωτικότητας και κατασκευαστές αντιγράφου

Ενότητα 2: Διαχείριση Μνήμης και **Σχεδιασμός Κλάσεων**

Έλεγχος ισότητας αντικειμένων

University of Cyprus
Department of Computer Science

Pitfall: Use of = and == with Variables of a Class Type

- Όταν χρησιμοποιείται με μεταβλητές τύπου κλάσης, ο *τελεστής ανάθεσης* (=) παράγει δύο μεταβλητές που ονομάζουν το ίδιο αντικείμενο: a=b;
  - Aliasing (ψευδωνυμία)
  - Διαφορετικό από το πώς συμπεριφέρεται ο τελετής με μεταβλητές πρωτόγονου τύπου
- Ο *τελεστής ελέγχου ισότητας* (==) επίσης συμπεριφέρεται διαφορετικά με τις μεταβλητές κλάσης (class type variables)
  - Ο τελεστής == ελέγχει αν δύο χειριστήρια είναι ίσα, δηλαδή δείχνουν στην ίδια θέση μνήμης.
  - Σε αντίθεση με την μέθοδο **`equals`**, ο τελεστής == δεν μπορεί να ελέγξει ότι οι μεταβλητές παρουσίας τους έχουν τις ίδιες τιμές
  - Δύο αντικείμενα σε δύο διαφορετικές τοποθεσίες των οποίων οι μεταβλητές στιγμιότυπου έχουν ακριβώς τις ίδιες τιμές θα εξακολουθούσαν να εξετάζονται ως "μη ίσα"

 Copyright © 2017 Pearson Ltd. All rights reserved.

Έλεγχος «ισότητας» αντικειμένων

- Ισότητα/Ισοδυναμία αντικειμένων.
  - Πώς ορίζεται;
  - Πώς ελέγχεται;
- Για τον έλεγχο των περιεχομένων ενός αντικειμένου για ισοδυναμία με άλλο αντικείμενο, μπορεί να καθοριστεί κατάλληλα η μέθοδος **`equals`**, η οποία υπάρχει σε όλες τις κλάσεις / αντικείμενα.

Έλεγχος ισότητας αντικειμένων

```
public class EqualsMethod {
   public static void main(String[] args) {
        Integer n1 = new Integer(47);
        Integer n2 = new Integer(47);
        System.out.println(n1.equals(n2));
   } }

class Value { int i; }
public class EqualsMethod2 {
   public static void main(String[] args) {
        Value v1 = new Value();
        Value v2 = new Value();
        System.out.println(v1.equals(v2));
   }}
```

Επιστρέφει true

Επιστρέφει false

Έλεγχος ισότητας αντικειμένων

Η σταθερά `null`

University of Cyprus
Department of Computer Science

The Constant `null`

- `null` είναι μια ειδική σταθερά που μπορεί να εκχωρηθεί σε μια μεταβλητή οποιουδήποτε τύπου κλάσης

```
YourClass yourObject = null;
```

- Χρησιμοποιείται για να δείξει ότι η μεταβλητή δεν έχει "πραγματική τιμή"
  - Συχνά χρησιμοποιείται σε κατασκευαστές για την προετοιμασία μεταβλητών στιγμιότυπου τύπου κλάσης όταν δεν υπάρχει προφανές αντικείμενο προς χρήση.
- `null` **δεν είναι αντικείμενο:** είναι κωδικός «αναπλήρωσης» (placeholder) για μια αναφορά που δεν παραπέμπει σε κάποια θέση μνήμης
  - Επειδή είναι σαν διεύθυνση μνήμης, χρησιμοποιείστε τους τελεστές `==` or `!=` (αντί της `equals`) για να δείτε αν μια μεταβλητή κλάσης είναι null

```
if (yourObject == null)  . . .
```


Pitfall: Null Pointer Exception

- Ακόμα κι αν μια μεταβλητή κλάσης μπορεί να αρχικοποιηθεί σε **null**, αυτό δεν σημαίνει ότι το null είναι αντικείμενο
  - Το null είναι μόνο ένας κωδικός αναπλήρωσης για ένα αντικείμενο
- Μια μέθοδος δεν μπορεί να κληθεί χρησιμοποιώντας μια μεταβλητή που έχει αρχικοποιηθεί σε **null**
  - Το αντικείμενο μέσω του οποίου θα πρέπει να κληθεί η μέθοδος δεν υπάρχει!
- Οποιαδήποτε προσπάθεια να γίνει αυτό θα έχει ως αποτέλεσμα μήνυμα σφάλματος: `"Null Pointer Exception"`
  - For example, if the class variable has not been initialized at all (and is not assigned to **null**), the results will be the same

 Copyright © 2017 Pearson Ltd. All rights reserved.

Ανώνυμα αντικείμενα (anonymous objects)

- Ο τελεστής `new` καλεί έναν κατασκευαστή που αρχικοποιεί ένα αντικείμενο και επιστρέφει μια αναφορά στη θέση μνήμης του αντικειμένου που δημιουργήθηκε
  - Αυτή η αναφορά μπορεί να ανατεθεί σε μεταβλητή του τύπου κλάσης του αντικειμένου.
- Μερικές φορές το αντικείμενο που δημιουργείται χρησιμοποιείται ως όρισμα σε μια μέθοδο και δεν χρησιμοποιείται ποτέ ξανά
  - Σε αυτήν την περίπτωση, το αντικείμενο δεν χρειάζεται να εκχωρηθεί σε μια μεταβλητή, δηλαδή να δοθεί ένα όνομα
- Ένα αντικείμενο του οποίου η αναφορά δεν έχει εκχωρηθεί σε μια μεταβλητή ονομάζεται **ανώνυμο αντικείμενο** (anonymous object).

 Copyright © 2017 Pearson Ltd. All rights reserved.