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

Αριθμοί κινητής υποδιαστολής (Float/Double) και Δομημένος Προγραμματισμός

Αθ. Ανδρούτσος

Αναπαράσταση δεκαδικών αριθμών

- Οι δεκαδικοί αριθμοί Αναπαρίστανται σύμφωνα με το πρότυπο **IEEE 754** – Αριθμητική Κινητής Υποδιαστολής
  - Πρόσημο (**Sign**) -- Πρώτο αριστερό bit 0 για θετικούς /1 για αρνητικούς
  - Κλασματικό μέρος (**Mantissa**)
  - Εκθέτης (**Exponent**) -- Προσημασμένος ακέραιος με βάση το 2
- Αναπαράσταση αριθμών κινητής υποδιαστολής
  - Πρόσημο * κλασματικό μέρος * $2^{\text{εκθέτης}}$
  - $-1^{S} * M * 2^{E}$

Τύποι Δεδομένων Float / Double (1)

- **Float**
  - Μονής ακρίβειας 32-bit -> 1 / 23 / 8
- **Double**
  - Διπλής ακρίβειας 64-bit -> 1 / 52 / 11
- Τα **δεκαδικά literals** θεωρούνται double
- Για παράδειγμα το 3.5 δεσμεύει 64-bits στη μνήμη και η Java το αναπαριστά ως double
- Η Java χρησιμοποιεί **τελεία (.)** για τα δεκαδικά

Float – Double Data Types


```java
package gr.aueb.cf.ch5;

/**
 * Floating point numbers (float, double) type range.
 */
public class FloatingTypeApp {

    public static void main(String[] args) {
        System.out.printf("Type: %s, Size: %d, Min: %e, Max: %e\n", Float.TYPE, Float.SIZE, Float.MIN_VALUE, Float.MAX_VALUE);
        System.out.printf("Type: %s, Size: %d, Min: %e, Max: %e", Double.TYPE, Double.SIZE, Double.MIN_VALUE, Double.MAX_VALUE);
    }
}
```

```
Run: FloatingTypeApp
"C:\Program Files\Amazon Corretto\jdk11.0.10_9\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Editi
Type: float, Size: 32, Min: 1,401298e-45, Max: 3,402823e+38
Type: double, Size: 64, Min: 4,900000e-324, Max: 1,797693e+308
Process finished with exit code 0
```

- Το **%e** στην *printf* αναπαριστά το scientific notation των δεκαδικών: π.χ. 1.3E5 που σημαίνει $1.3 * 10^5$
- Οποιασδήποτε δεκαδικός αριθμός, μικρότερος ή μικρότερος ή μεγαλύτερος από το εύρος δημιουργεί underflow ή overflow
- Ο τύπος double έχει μεγαλύτερο εύρος τιμών από τους float

Δηλώσεις Float / Double (1)

- **Μεταβλητές δηλώνουμε ως:**
  - float f;
  - float height=3.5F (Το F είναι type suffix και μετατρέπει το 3.5 από double που είναι κανονικά σε float ώστε να εκχωρηθεί στο height που είναι float. Αλλιώς υπάρχει error)
  - double radius = 7.9768;
  - double price = 15D; (Το D είναι type suffix και μετατρέπει το 15 από int που είναι κανονικά σε double ώστε να εκχωρηθεί στο height που είναι float. Αλλιώς δεν υπάρχει error, γίνεται αυτόματα typecast από int σε double, αλλά με type suffix βοηθάμε στην γρηγορότερη μετατροπή)

Δηλώσεις Float / Double (2)


```java
package gr.aueb.cf.ch5;

/**
 * Δηλώσεις και αρχικοποιήσεις Float και Double
 */
public class DeclarationsApp {

    public static void main(String[] args) {
        float f = 31.4F;        // Typecast από double σε float
        float age = 23F;        // Typecast από int σε float

        double d = 9.17;        // Δε χρειάζεται typecast, double σε double
        double price = 17D;     // Typecast από int σε double
    }
}
```

- Όπως έχουμε αναφέρει όταν έχουμε δεξιά και αριστερά μιας εκχώρησης διαφορετικούς τύπους πρέπει να γίνει typecast είτε έμμεσα ή άμεσα

Τύποι Δεδομένων Float / Double (1)

- Σταθερές
  - final float LIGHT_SPEED = 3.0E5F;
  - final double PI = 3.141592653532385;
- Οι wrapper κλάσεις για τους float, double είναι οι κλάσεις Float, Double

Παραστάσεις (Expressions)


```java
package gr.aueb.cf.ch5;

/**
 * Παραστάσεις floating point.
 */
public class FloatingExpressionsApp {

    public static void main(String[] args) {
        int intNum = 10;
        float floatNum = 0.1F;
        double doubleNum1 = 12.5D;
        double doubleNum2 = 2D;

        float floatResult = 0.0F;
        double doubleResult = 0.0;

        // Αν υπάρχει ένας float οι μικρότερου μεγέθους τύποι
        // μετατρέπονται σε float.
        floatResult = floatNum + intNum;

        // Αν υπάρχει έστω και ένας double,
        // τα intNum και floatNum μετατρέπονται σε double
        doubleResult = doubleNum1 + floatNum * intNum;

        System.out.printf("flotResult = %f", floatResult);
        System.out.printf("doubleResult = %f", doubleResult);
    }
}
```

- Ισχύουν όλοι οι αριθμητικοί τελεστές, όπως και στους ακέραιους:
- **+, -, *, /, %**
- Στην printf οι floating points float και double αναπαρίστανται με **%f**

Διαίρεση (1)


```java
package gr.aueb.cf.ch5;

/**
 * Διαίρεση και υπόλοιπο δεκαδικών
 */
public class DivisionApp {

    public static void main(String[] args) {
        double num1 = 12.5;
        double num2 = 2.0;
        double result = 0.0;
        double remaining = 0.0;

        result = num1 / num2;
        remaining = num1 % num2;

        System.out.printf("%.2f / %.2f = %05.2f\n", num1, num2, result);
        System.out.printf("%.2f / %.2f = %5.2f", num1, num2, remaining);
    }
}
```

```
Run: DivisionApp
"C:\Program Files\Amazon Corretto\jdk11.0.10_9\bin\java.exe" "-javaagent
12,50 / 2,00 = 06,25
12,50 / 2,00 =  0,50
```

- Οι τελεστές / και % έχουν την παρακάτω ερμηνεία:
- Το πηλίκο είναι το **κανονικό πηλίκο**, το **mod είναι όπως το υπόλοιπο διαίρεσης ακεραίων**
- Formatting δεκαδικών κάνουμε περίπου όπως στους ακεραίους
- Με **%.2f** εννοούμε εμφάνιση με 2 δεκαδικά ψηφία, **%5.2f** είναι 5 θέσεις εκ των οποίων δύο δεκαδικά και **%05.2f** εννοούμε όπως πριν 5 θέσεις αλλά το left padding θα είναι με 0 αντί για κενά

Διαίρεση (2)


```java
package gr.aueb.cf.ch5;

/**
 * Διαίρεση και υπόλοιπο δεκαδικών
 */
public class DivisionApp {

    public static void main(String[] args) {
        double num1 = 12.5;
        double num2 = 2.0;
        double result = 0.0;
        double remaining = 0.0;

        result = num1 / num2;
        remaining = num1 % num2;

        System.out.printf("%.2f / %.2f = %05.2f\n", num1, num2, result);
        System.out.printf("%.2f / %.2f = %5.2f", num1, num2, remaining);
    }
}
```

```
Run:  DivisionApp
"C:\Program Files\Amazon Corretto\jdk11.0.10_9\bin\java.exe" "-javaagent
12,50 / 2,00 = 06,25
12,50 / 2,00 =  0,50
```

- Παρατηρούμε ότι η διαίρεση δεκαδικών επιστρέφει δεκαδικό
- Το υπόλοιπο της διαίρεσης (mod) επιστρέφει όπως αν ήταν διαίρεση ακεραίων

Διαίρεση με το 0


```java
package gr.aueb.cf.ch5;

/**
 * Διαίρεση με το 0.
 */
public class DivideZeroApp {

    public static void main(String[] args) {
        double dividend = 17.0;
        double divisor = 0.0;
        double result = 0.0;

        result = dividend / divisor;

        System.out.println("Result = " + result);
    }
}
```

```
Run: DivideZeroApp
"C:\Program Files\Amazon Corretto\jdk11.0.10_9\bir
Result = Infinity
```

- Η διαίρεση με το 0 δεν δημιουργεί εξαίρεση όπως στους ακέραιους, αλλά επιστρέφει imfinity

Καλούπωμα – Typecasting (1)

- Μετατροπή από float/double σε int
  - απώλεια κλασματικού μέρους
- Μετατροπή από int σε float/double καθώς και από float σε double
  - επιτρέπεται
- Από double σε float
  - απώλεια ακρίβειας

Καλούπωμα – Typecasting (1)

byte to short, int, long, float, or double

short to int, long, float, or double

char to int, long, float, or double

int to long, float, or double

long to float or double

float to double

- Γενικά, η Java κάνει αυτόματα type conversion σύμφωνα με τους κανόνες αριστερά

- Επίσης σε παραστάσεις γίνεται αυτόματο widening των τύπων, όπως φαίνεται παραπάνω στα primitive conversions (typecasts)

Είσοδος/Εξοδος

- Είσοδος με Scanner
  - .nextFloat()
  - .nextDouble()
- Έξοδος με
  - System.out.println();
  - System.out.printf("%f"); // simple float, double
  - System.out.printf("%e"); // scientific notation
  - System.out.printf("%g"); // the shorter of //above is selected

Σφάλματα αναπαράστασης & υπολογισμού (1)

- Ο αριθμός 0.1 του δεκαδικού συστήματος, στο δυαδικό είναι ο 0.0001100110011001100110011... (repeating)
- Παρατηρούμε ότι στο δυαδικό σύστημα δεν μπορεί με ακρίβεια να αναπαρασταθεί ο 0.1
- Γενικά, οι δεκαδικοί αριθμοί δεν μπορούν να αναπαρασταθούν με ακρίβεια στο δυαδικό σύστημα, και αν έχουν μεγάλο κλασματικό μέρος θα πρέπει να στρογγυλοποιηθεί με απώλεια ακρίβειας

Σφάλματα αναπαράστασης & υπολογισμού (2)

- Επειδή **οι δεκαδικοί αριθμοί δεν μπορούν να αναπαρασταθούν με απόλυτη ακρίβεια σε ένα σύστημα Η/Υ**, οι πράξεις ανάμεσα σε δεκαδικούς αριθμούς γίνονται με βάση τα σημαντικά ψηφία
- Αν για παράδειγμα έχουμε δύο αριθμούς:
  - x=1.0002 και
  - Y=1.0001
- Οι x και y είναι ίσοι αν λάβουμε 3 σημαντικά ψηφία μετά την υποδιαστολή και άνισοι αν λάβουμε 4 σημαντικά ψηφία

Σφάλματα Αναπαράστασης


```java
package gr.aueb.cf.ch5;

public class WithoutEpsilonApp {

    public static void main(String[] args) {
        double actual = 0.0;
        double expected = 1.0;

        for (int i = 1; i <= 10; i++) {
            actual += 0.1;
        }

        if (actual == expected) System.out.println("EQUAL");
        else System.out.println("NOT EQUAL");

        System.out.printf("actual: %.20f, expected: %.20f", actual, expected);
    }
}
```

```
Run: WithoutEpsilonApp
"C:\Program Files\Amazon Corretto\jdk11.0.10_9\bin\java.exe" "-javaagent:C:\Pr
NOT EQUAL
actual: 0,99999999999999990000, expected: 1,00000000000000000000
```

- Το 0.1 όπως αναφέραμε δεν μπορεί να αναπαρασταθεί με ακρίβεια
- Τα διαδοχικά αθροίσματα κεφαλαιοποιούν το σφάλμα αναπαράστασης του 0.1

Σφάλματα αναπαράστασης & υπολογισμού (2)

- Για να ελέγξουμε αν δύο float ή double είναι ίσοι / άνισοι / μεγαλύτερος / μικρότερος ***θα πρέπει να συγκρίνουμε*** αφού ορίσουμε πρώτα ένα επίπεδο σημαντικότητας μέσω μίας σταθεράς έστω EPSILON

Παραδείγματα

- x == y
  - final double EPSILON = 0.00005;
    Math.abs(x-y) <= EPSILON;
    Αν ισχύει τότε x, y ίσα αν θεωρήσουμε
    5 σημαντικά ψηφία μετά την υποδιαστολή
- x == 0
  - final double EPSILON = 0.000005;
    Math.abs(x) <= EPSILON;
    Αν ισχύει τότε x ίσο με το 0 αν θεωρήσουμε 6
    σημαντικά ψηφία μετά την υποδιαστολή

EPSILON


```java
package gr.aueb.cf.ch5;

public class EpsilonApp {

    public static void main(String[] args) {
        final double EPSILON = 0.000005;    // 6 significant digits
        double actual = 0.0;
        double expected = 1.0;

        for (int i = 1; i <= 10; i++) {
            actual += 0.1;
        }

        if (Math.abs(actual-expected) <= EPSILON) System.out.println("EQUAL");
        else System.out.println("NOT EQUAL");
    }
}
```

```
Run:  EpsilonApp
"C:\Program Files\Amazon Corretto\jdk11.0.10_9\bin\java.exe" "-javaagent:C:\Pro
EQUAL
```

- Παρατηρούμε ότι θεωρούμε ίσους 1) το αποτέλεσμα του αθροίσματος 0,1 + 0,1 + 0,1 … (10 φορές) και 2) το 1.0
- Ο λόγος είναι πως θεωρούμε 6 σημαντικά δεκαδικά ψηφία μετά την υποδιαστολή και επομένως η διαφορά τους είναι μικρότερη από 0.000005

Ορθογώνιο Τρίγωνο


```java
package gr.aueb.cf.ch5;

import java.util.Scanner;

/**
 * Ελέγχει αν ένα τρίγωνο με υποτείνουσα α
 * και πλευρές β, γ είναι ορθογώνιο, δηλαδή
 * αν α^2 = β^2 + γ^2
 */
public class RightTriangleApp {

    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        final double EPSILON = 0.000005;
        double a = 0.0;
        double b = 0.0;
        double c = 0.0;
        boolean isRight = false;

        System.out.println("Please insert a, b, c");
        a = in.nextDouble();
        b = in.nextDouble();
        c = in.nextDouble();

        isRight = (Math.abs(a*a - b*b - c*c) <= EPSILON);

        System.out.printf("Triangle is right:  %b", isRight);
    }
}
```

- Πρώτα διαβάζουμε με Scanner τρεις αριθμούς τύπου double
- Στη συνέχεια ελέγχουμε αν το απόλυτο, Math.abs(), αποτέλεσμα της έκφρασης a^2 – b^2 – c^2 είναι μικρότερο ίσο από το EPSILON που έχει 6 σημαντικά ψηφία
- Αν είναι μικρότερο ή ίσο τότε το τρίγωνο είναι ορθογώνιο. Αυτό ελέγχει μεταβλητή boolean, ***isRight***

Big Decimal


```java
package gr.aueb.cf.ch5;

import java.math.BigDecimal;

public class BigDecimalApp {

    public static void main(String[] args) {
        BigDecimal actual = new BigDecimal("0.0");
        BigDecimal expected = new BigDecimal("1.0");

        for (int i = 1; i <= 10; i++) {
            actual = actual.add(BigDecimal.valueOf(0.1));
        }

        System.out.println(actual);
        System.out.println(expected);

        if (actual.equals(expected)) System.out.println("EQUAL");
        else System.out.println("NOT EQUAL");
    }
}
```

- Για έλεγχο χωρίς το EPSILON αλλά και για διαχείριση του overflow, underflow μας παρέχεται η κλάση BigDecimal που είναι ανάλογη της BigInteger

Math

- ***Όπως έχουμε ξαναδεί, η κλάση Math είναι μία κλάση της Java*** που παρέχει μεθόδους που αναπαριστούν μαθηματικές συναρτήσεις. Στον παρακάτω πίνακα παρουσιάζονται ενδεικτικά πέντε μέθοδοι της Math, που αφορούν double. Για πλήρη τεκμηρίωση της Math μπορείτε να πάτε στα Java docs:
- https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/Math.html

Μέθοδος	
Math.abs(double a)	Επιστρέφει την απόλυτη τιμή του a. Για παράδειγμα το Math.abs(-3.2) είναι 3.2
Math.max(int a, int b) ***Math.max(double a, double b)***	Επιστρέφει το μεγαλύτερο των δύο αριθμών a και b. Για παράδειγμα το Math.max(5, 12) είναι 12
Math.min(int a, int b) ***Math.min(double a, double b)***	Επιστρέφει το μικρότερο των δύο αριθμών a και b. Για παράδειγμα το Math.min(5, 12) είναι το 5
Math.sqrt(double a)	Επιστρέφει την τετραγωνική ρίζα του a Για παράδειγμα το Math.sqrt(25) είναι το 5
Math.random()	Επιστρέφει ένα τυχαίο δεκαδικό αριθμό double μεγαλύτερο ή ίσο του 0 και μικρότερο του 1

Δομημένος Προγραμματισμός (1)

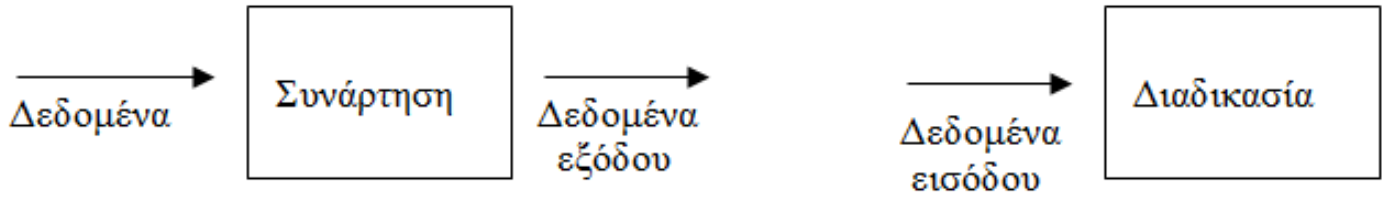
- Πολλές φορές κάποια τμήματα κώδικα ***επαναχρησιμοποιούνται*** και δεν είναι αποτελεσματικό να ξαναγράφουμε τον κώδικα όποτε τον χρειαζόμαστε
- Είναι πιο αποδοτικό να γράφουμε **ανεξάρτητα μπλοκ κώδικα**, ώστε να τα επαναχρησιμοποιούμε αλλά και να τα ελέγχουμε και συντηρούμε ευκολότερα

Δομημένος Προγραμματισμός (2)

- Τα βασικό μπλοκ κώδικα στον αντικειμενοστραφή προγραμματισμό είναι η ***μέθοδος (method)***
- Ιστορικά οι μέθοδοι προέρχονται ως **συνέχεια δύο βασικών δομών κώδικα** που υπήρχαν στις παλαιότερες γλώσσες συναρτησιακού / διαδικασιακού προγραμματισμού (functional / procedural programming), των ***Συναρτήσεων*** (Functions) και των ***Διαδικασιών*** *(Procedures)*

Συναρτήσεις - Διαδικασίες

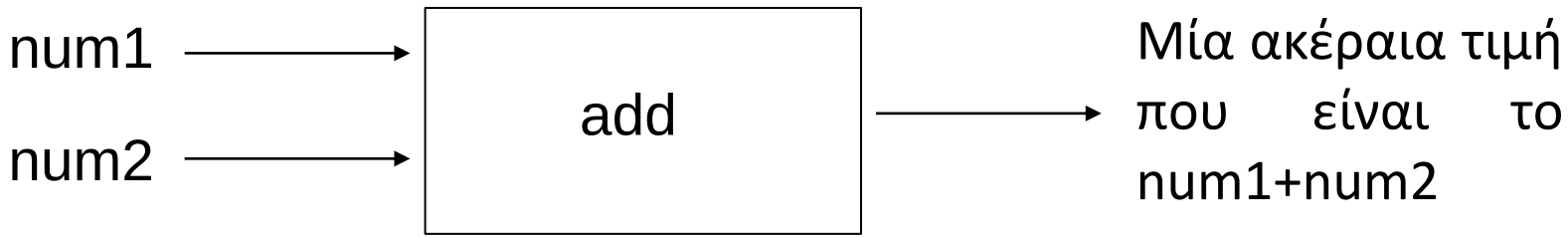
- Όπως φαίνεται (σχήμα 1) μία ***συνάρτηση*** μπορεί να λαμβάνει δεδομένα εισόδου και δίνει μία τιμή εξόδου ενώ μία ***διαδικασία*** λαμβάνει δεδομένα εισόδου και απλά επιτελεί μία λειτουργία, χωρίς να παράγει τιμή εξόδου



Σχήμα 1: Διαδικασίες και συναρτήσεις

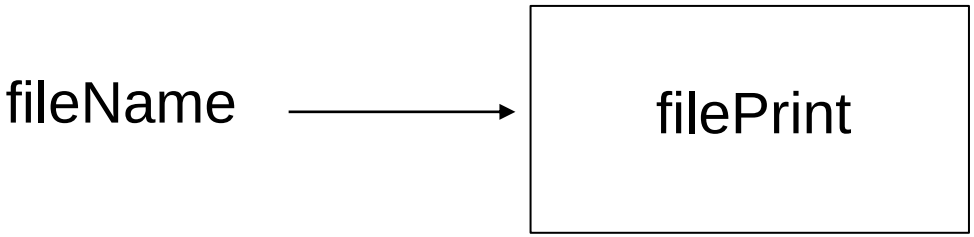
Μέθοδοι – Παράδειγμα (1)

- Οι μέθοδοι είναι αυτόνομα τμήματα κώδικα, που μπορεί να λαμβάνουν ως είσοδο 0 ή περισσότερες τιμές εισόδου (input data) και αφού τα επεξεργαστούν, να παράγουν 0 ή μία τιμή εξόδου
- Για παράδειγμα μία μέθοδος ***add*** λαμβάνει ***ως είσοδο δύο (2) ακεραίους, έστω num1 και num2***, επιτελεί την πράξη της άθροισης και παράγει ***ως έξοδο μία ακέραια τιμή*** του αθροίσματος



Μέθοδοι – Παράδειγμα (2)

- Έστω, για παράδειγμα η εκτύπωση ενός αρχείου. Έστω η μέθοδος ***filePrint*** που λαμβάνει ***ως παράμετρο εισόδου ένα String έστω fileName***, που είναι το όνομα του αρχείου, και στη συνέχεια εκτυπώνει το αρχείο ***χωρίς να επιστρέφει τιμή***



Μορφή Μεθόδων (1)

- Ο γενικός τύπος μίας μεθόδου είναι:
- ***<επιστρεφόμενος τύπος> <όνομα μεθόδου> (λίστα τυπικών παραμέτρων αν υπάρχουν) εξαιρέσεις***

```
{
    //σώμα της μεθόδου
}
```

- ***Επιστρεφόμενος τύπος*** της μεθόδου είναι ο τύπος της τιμής που επιστρέφεται (π.χ. int, double, String, κλπ.) ή αν δεν επιστρέφει τιμή, τότε ο επιστρεφόμενος τύπος είναι ***void***

Μορφή Μεθόδων (2)

- Το ***όνομα της μεθόδου*** το επιλέγουμε εμείς ώστε να αντανακλά τη λειτουργία της μεθόδου
- Το όνομα της μεθόδου ***είναι ρήμα υποδηλώνοντας ότι κάνει κάτι*** και ακολουθεί τους κανόνες των identifiers, ενώ ***κατά σύμβαση ξεκινά με μικρό γράμμα και κάθε επόμενη λέξη ξεκινά με κεφαλαίο γράμμα***

Μορφή Μεθόδων (3)

- Η ***λίστα τυπικών παραμέτρων*** δηλώνει τους τύπους και τα ονόματα των μεταβλητών εισόδου μέσα σε παρενθέσεις
- Αν οι παράμετροι είναι περισσότεροι από μία τις διαχωρίζουμε με κόμμα

Μορφή Μεθόδων (4)

- Οι ***εξαιρέσεις*** είναι μία λίστα με τους τύπους των σφαλμάτων που μπορεί να προκληθούν κατά την εκτέλεση της μεθόδου, όπως είχαμε δει με την ***System.in.read() και την main()*** που έκανε ***throws java.io.IOException***

Σώμα μεθόδων

- Το σώμα μία μεθόδου είναι **μέσα στα άγκιστρα { }**
- Ο τρόπος συγγραφής είναι παρόμοιος με τον τρόπο που ακολουθήσαμε στην main() με μόνη ***σημαντική διαφορά, ότι εάν η μέθοδος επιστρέφει τιμή πρέπει στο τέλος να έχει μία εντολή return***
- //Επικεφαλίδα Μεθόδου
  Επιστρεφόμενος-Τύπος όνομα-μεθόδου(παράμετροι εισόδου)
  // Σώμα μεθόδου
  {
  1. Δηλώσεις και Αρχικοποίηση μεταβλητών
  2. Εντολές / επεξεργασία (συνήθως επαναληπτικές)
  3. Εκτύπωση αποτελεσμάτων *-- Αν η μέθοδος εκτυπώνει αποτελέσματα*
  4. Return τιμή *--Αν η μέθοδος επιστρέφει τιμή*

  }

Δήλωση και ορισμός μεθόδων

- Οι μέθοδοι δηλώνονται και ορίζονται ***μέσα σε μία κλάση***
- Οι **μεταβλητές που δηλώνονται μέσα σε μια μέθοδο είναι τοπικές μεταβλητές** της μεθόδου
- Οι **τυπικές παράμετροι μια μεθόδου είναι επίσης τοπικές μεταβλητές** της μεθόδου

Κλήση Μεθόδων

- Οι μέθοδοι ορίζονται μια φορά και στη συνέχεια ***μπορούν να καλούνται σε διάφορα σημεία ενός προγράμματος***
- Καλούνται με το όνομά τους και μία λίστα με τις τιμές των πραγματικών ορισμάτων (actual arguments) χωρισμένες με κόμμα
- Η τιμή ενός πραγματικού ορίσματος μπορεί να είναι μία τιμή, μία μεταβλητή, μία παράσταση, με τύπο συμβατό με αυτόν της αντίστοιχης τυπικές παραμέτρου

Επιστροφή ελέγχου

- Αν μία μέθοδος που ***δεν επιστρέφει αποτέλεσμα*** κληθεί σε ένα σημείο του προγράμματος, τότε ***ο έλεγχος του προγράμματος επιστρέφει*** (στο σημείο που κλήθηκε η μέθοδος) ***είτε όταν τελειώσει η μέθοδος ή όταν εκτελεστεί μία εντολή return***
- Αν μία μέθοδος ***που επιστρέφει αποτέλεσμα*** κληθεί σε ένα σημείο του προγράμματος, τότε ***ο έλεγχος του προγράμματος επιστρέφει*** όταν εκτελεστεί μία εντολή ***return παράσταση*** που επιστρέφει την τιμή της παράστασης στην καλούσα μέθοδο

Παράδειγμα Μέθοδος add(int a, int b)


```java
package gr.aueb.cf.ch5;

public class AddApp {

    public static void main(String[] args) {
        int a = 10;
        int b = 20;
        int sum = 0;

        sum = add(a, b);

        System.out.println("Sum = " + sum);
    }
    1 usage
    public static int add(int a, int b) {
        // Δήλωση και αρχικοποίηση
        int sum = 0;

        // Εντολές
        sum = a + b;

        // Επιστροφή αποτελέσματος
        return sum;
    }
}
```

- Η μέθοδος add(int a, int b) υπολογίζει και επιστρέφει το άθροισμα a + b
- Αποτελείται από: 1) την επικεφαλίδα και 2) το σώμα της μεθόδου μέσα σε { }
- Η επικεφαλίδα αποτελείται από τον επιστρεφόμενο τύπο (int), το όνομα της μεθόδου (add) και μέσα σε παρενθέσεις τις τυπικές παραμέτρους εισόδου (a, b)
- Οι τυπικές παράμετροι δηλώνονται όπως οι δηλώσεις μεταβλητών (int a, int b) χωρισμένες με κόμμα
- Μέσα στη main() η μέθοδος καλείται με το όνομά της και με πραγματικές παραμέτρους τα a, b της main, που είναι διαφορετικές μεταβλητές από τις τυπικές παραμέτρους a, b της add (παρότι έχουν το ίδιο όνομα)

Πλήρες όνομα static μεθόδων


```java
package gr.aueb.cf.ch5;

public class AddApp {

    public static void main(String[] args) {
        int a = 10;
        int b = 20;
        int sum = 0;

        sum = AddApp.add(a, b);

        System.out.println("Sum = " + sum);
    }
    1 usage
    public static int add(int a, int b) {
        // Δήλωση και αρχικοποίηση
        int sum = 0;

        // Εντολές
        sum = a + b;

        // Επιστροφή αποτελέσματος
        return sum;
    }
}
```

- Οι static μέθοδοι όπως η add() ανήκουν στην κλάση και γιαυτό το **πλήρες όνομά** τους είναι ***το όνομα της κλάσης ακολουθούμενο από τον τελεστή τελεία (.) ακολουθούμενο από το όνομα της μεθόδου***
- Γιαυτό το πλήρες όνομα της add() είναι ***AddApp.add()***
- Μπορούμε να καλούμε ***μία static μέθοδο με το πλήρες όνομά της*** ή αν καλούμε ***μέσα από την ίδια την κλάση*** μπορούμε να αναφερόμαστε με το απλό όνομα της, όπως το add(), (βλ. στην προηγούμενη διαφάνεια)

Η μέθοδος add πιο αναλυτικά


```java
public class AddApp2 {

    public static void main(String[] args) {
        int a = 10;
        int b = 20;
        int sum = 0;

        sum = add(a, b);

        System.out.println("Sum = " + sum);
    }

    /**
     * Add two integers.
     *
     * @param a     the first int to add
     * @param b     the second int to add
     * @return      the sum of a and b
     */
    1 usage
    public static int add(int a, int b) {
        return a + b;
    }
}
```

- Εδώ η add περιέχει στο σώμα της μόνο μία `return a + b;` αντί για τρεις στα προηγούμενα παραδείγματα. Είναι συντομότερος τρόπος.
- Επίσης έχουμε εισάγει Javadoc σε επίπεδο μεθόδου όπου εξηγούμε τη λειτουργία της μεθόδου και αναφέρουμε τη σημασία των παραμέτρων εισόδου και της τιμής εξόδου

Μενού Επιλογών με μέθοδο (1)


```
38      /**
39       * Prints a CRUD menu.
40       */
41      public static void printMenu() {
42          System.out.println("Επιλέξτε ένα από τα παρακάτω");
43          System.out.println("1. Εισαγωγή");
44          System.out.println("2. Διαγραφή");
45          System.out.println("3. Αναζήτηση");
46          System.out.println("4. Ενημέρωση");
47          System.out.println("5. Έξοδος");
48      }
```

- Η ***printMenu()*** δεν επιστρέφει τιμή και για αυτό ***ο επιστρεφόμενος τύπος είναι void.*** Η printMenu() δεν επιστρέφει τιμή μιας και δεν κάνει κάτι παρά μόνο εκτυπώνει το Μενού επιλογών
- Επίσης ***δεν έχει τυπικές παραμέτρους*** δεδομένου ότι δεν λαμβάνει τιμές εισόδου ***(ωστόσο πρέπει να έχει παρενθέσεις)***

Μενού Επιλογών με μέθοδο (2)


```
1  package gr.aueb.cf.ch5;
2
3  import java.util.Scanner;
4
5  public class MenuApp {
6
7      public static void main(String[] args) {
8          Scanner in = new Scanner(System.in);
9          int choice = 0;
10
11         do {
12             printMenu();
13             choice = in.nextInt();
14
```

- Παρατηρήστε την κλήση της μεθόδου *printMenu*()
- Καλούμε με το όνομα της μεθόδου καθώς και τις παρενθέσεις
- Η μέθοδος εκτελείται, εμφανίζει το μενού και επιστρέφει στο σημείο που κλήθηκε, οπότε και συνεχίζει στην επόμενη εντολή

Μενού Επιλογών με μέθοδο (3)


```
15          switch (choice) {
16              case 1:
17                  System.out.println("Επιτυχής εισαγωγή");
18                  break;
19              case 2:
20                  System.out.println("Επιτυχής διαγραφή");
21                  break;
22              case 3:
23                  System.out.println("Επιτυχής ενημέρωση");
24                  break;
25              case 4:
26                  System.out.println("Επιτυχής αναζήτηση`");
27                  break;
28              case 5:
29                  System.out.println("Επιτυχής έξοδος");
30                  break;
31              default:
32                  System.out.println("Λάθος επιλογή");
33                  break;
34          }
35      } while (choice != 5);
36  }
```

- Στη συνέχεια ελέγχουμε το choice με μία switch / case

Υπολογιστής τσέπης (1)


```java
package gr.aueb.cf.ch5;

import java.util.Scanner;

/**
 * Απλός υπολογιστής τσέπης με μεθόδους.
 */
public class CalculatorApp {
    static Scanner in = new Scanner(System.in);

    public static void main(String[] args) {
        int choice = 0;
        int num1 = 0, num2 = 0;
        int result = 0;

        do {
            printMenu();
            choice = getUserChoice();
```

- Η main() ***απλά υλοποιεί την βασική επαναληπτική διαδικασία*** και καλεί μέσα στο σώμα της διάφορες μεθόδους, στην αρχή την ***printMenu()***, στη συνέχεια την **getUserChoice()** καθώς και άλλες μεθόδους που θα δούμε στη συνέχεια

Υπολογιστής τσέπης (2)


```
16          do {
17              printMenu();
18              choice = getUserChoice();
19
20              if (choice < 1 || choice > 6) {
21                  System.out.println("Λάθος επιλογή");
22                  continue;
23              }
24
25              System.out.println("Please insert two ints");
26              num1 = getOneNumber();
27              num2 = getOneNumber();
28              result = getResult(choice, num1, num2);
29              System.out.println("Result: " + result);
30
31          } while (choice != 6);
32          System.out.println("Thanks for using our app");
33      }
```

- Αν το choice δεν ανήκει στις έγκυρες επιλογές, τότε κάνουμε continue και δεν προχωράμε
- Διαβάζουμε ένα – ένα τους ακεραίους με *getOneNumber*() και μετά καλούμε την *getResult*()

Υπολογιστής τσέπης (3)


```
35      public static void printMenu() {
36          System.out.println("Επιλέξτε ένα από τα παρακάτω: ");
37          System.out.println("1. Πρόσθεση");
38          System.out.println("2. Αφαίρεση");
39          System.out.println("3. Πολλαπλασιασμός");
40          System.out.println("4. Διαίρεση");
41          System.out.println("5. Υπόλοιπο Διαίρεσης");
42          System.out.println("6. Έξοδος");
43      }
44
45      public static int getUserChoice() {
46          return in.nextInt();
47      }
48
49      public static int add(int a, int b) {
50          return a + b;
51      }
52
53      public static int sub(int a, int b) {
54          return a - b;
55      }
```

- Υλοποίηση των μεθόδων του Μενού, της εισαγωγής της επιλογής από τον χρήστη καθώς και μεθόδων υλοποίησης των αριθμητικών πράξεων

Υπολογιστής τσέπης (4)


```
57      public static int mul(int a, int b) {
58          return a * b;
59      }
60
61      public static int div(int a, int b) {
62          if (b == 0) {
63              System.out.println("Invalid number: 0");
64              return 0;
65          }
66
67          return a / b;
68      }
69
70      public static int mod(int a, int b) {
71          if (b == 0) {
72              System.out.println("Invalid number: 0");
73              return 0;
74          }
75
76          return a % b;
77      }
```

- Υλοποίηση και άλλων μεθόδων υπολογισμού των αριθμητικών πράξεων

Υπολογιστής τσέπης (5)


```
79      public static int getResult(int choice, int num1, int num2) {
80          int result = 0;
81
82          switch (choice) {
83              case 1:
84                  result = add(num1, num2);
85                  break;
86              case 2:
87                  result = sub(num1, num2);
88                  break;
89              case 3:
90                  result = mul(num1, num2);
91                  break;
92              case 4:
93                  result = div(num1, num2);
94                  break;
95              case 5:
96                  result = mod(num1, num2);
97                  break;
98              case 6:
99                  System.out.println("Επιλέξατε έξοδο");
100                 break;
101             default:
102                 System.out.println("Λάθος Επιλογή");
104         }
105         return result;
106     }
```

- Υλοποίηση της getResult() με κλήση μέσα στη ***switch/case*** των μεθόδων:

add(),
sub()
mul()
div()
mod()

Υπολογιστής τσέπης (6)


```
108     public static int getOneNumber() {
109         return in.nextInt();
110     }
111 }
```

- Μπορούμε στο ***return να έχουμε παράσταση***, όπως εδώ, για να συντομεύσουμε τον κώδικα

Παραγοντικό του n

- Θα δούμε σε αυτό το παράδειγμα, κάτι που έχουμε ξαναδεί, τον υπολογισμό του n!, αλλά αυτή τη φορά με μορφή μεθόδου.
- Θυμίζουμε πως το n! υπολογίζεται ως το γινόμενο 1*2* … * n. Για παράδειγμα, το παραγοντικό το 4 είναι: ***4! = 1*2*3*4 = 24***
- Ορίζουμε το παραγοντικό του 0 να είναι: 0! = 1

Παραγοντικό του n ως μέθοδος


```
public static int factorial(int n) {
    int facto = 1;

    for (int i = 1; i <= n; i++) {
        facto = facto * i;
    }

    return facto;
}
```

- **Επιστρεφόμενος τύπος:** int
  Όνομα: factorial
  Τυπική παράμετρος: int n
- Δήλωση και Αρχικοποίηση μεταβλητών
- Επαναληπτική δομή ελέγχου for Όσο το i <= n πολλαπλασίασε το facto επί το i
- Επιστροφή τιμής με την return. Επιστρέφουμε (στην καλούσα μέθοδο) το αποτέλεσμα της factorial που είναι int

Παραγοντικό του n (1)


```java
package gr.aueb.cf.ch5;

import java.util.Scanner;

/**
 * Ορίζει μία μέθοδο που υπολογίζει το
 * παραγοντικό του n (n!).
 */
public class FactoApp {

    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int n = 0;

        System.out.println("Please insert an int");
        n = in.nextInt();

        System.out.printf("%d! = %d", n, facto(n));
    }
```

- Μέσα στην κλάση ***FactoApp*** ορίζουμε δύο μεθόδους:
  - την ***main***, και
  - την ***facto*** *(βλ. επόμενη διαφάνεια)*
- Η ***main*** λειτουργεί ως driver method δηλαδή απλά ελέγχει την ορθότητα της factorial: (i) δίνοντας τιμή στο n από τον χρήστη, (ii) καλώντας την factorial, και (iii) εκτυπώνοντας το αποτέλεσμα

Παραγοντικό του n (2)


```
21      /**
22       * Iterative version of n!
23       *
24       * @param n      the n number of n!
25       * @return       1*2*..*n
26       */
27      public static int facto(int n) {
28          int facto = 1;
29
30          for (int i = 1; i <= n; i++) {
31              facto *= i;
32          }
33
34          return facto;
35      }
36  }
```

- Η ***facto*** παίρνει ως τυπική παράμετρο ένα ***int n*** και ***επιστρέφει int***

Ύψωση σε δύναμη (1)


```
22      /**
23       * Power a^n.
24       *
25       * @param a      the base
26       * @param n      the power to raise
27       * @return       the power of a to n
28       */
29      public static int pow(int a, int n) {
30          int power = 1;
31
32          for (int i = 1; i <= n; i++) {
33              power *= a;
34          }
35
36          return power;
37      }
38  }
```

- Η pow παίρνει δύο ***τυπικές παραμέτρους***, τις ***int a*** και ***int n***
- Στα ***javadoc*** τα @param αφορούν τις τυπικές παραμέτρους, όπου εξηγούμε με λόγια τι είναι η κάθε μια και το @return αφορά το τι επιστρέφει αυτή η μέθοδος

Ύψωση σε δύναμη (2)


```
1  package gr.aueb.cf.ch5;
2
3  import java.util.Scanner;
4
5  /**
6   * Ορίζει μέθοδο που υπολογίζει το a^n.
7   */
8  public class PowerApp {
9
10     public static void main(String[] args) {
11         Scanner in = new Scanner(System.in);
12         int a = 0;
13         int n = 0;
14
15         System.out.println("Please insert a and n");
16         a = in.nextInt();
17         n = in.nextInt();
18
19         System.out.printf("%d^%d = %d", a, n, pow(a, n));
20     }
```

- Η ***pow*** καλείται μέσα στην ***main()*** στην γραμμή 19, παίρνοντας ως ***πραγματικές παραμέτρους*** τις μεταβλητές ***a*** και ***n*** οι οποίες αντικαθιστούν τις τυπικές παραμέτρους ***a*** και ***n*** στο σώμα της ***pow***

Τυπικές παράμετροι

- Πρέπει να ***διαχωρίσουμε*** τις τυπικές από τις πραγματικές παραμέτρους
- Οι ***τυπικές παράμετροι*** μπορεί να είναι οποιουδήποτε απλού ή σύνθετου τύπου δεδομένων και **σκοπό έχουν να δηλώσουν τις μεταβλητές εκείνες που χρειάζεται να λάβει ως είσοδο η μέθοδος**
- Οι τυπικές παράμετροι επέχουν θέση τοπικών μεταβλητών στις μεθόδους στις οποίες ορίζονται

Πραγματικά ορίσματα (1)

- **Οι πραγματικές παράμετροι σκοπό έχουν να περάσουν στην μέθοδο τιμές πραγματικές εισόδου**
- Η τιμή ενός πραγματικού ορίσματος μπορεί να είναι μία τιμή (literal), μία μεταβλητή, μία παράσταση, με τύπο συμβατό με αυτόν της αντίστοιχης τυπικές παραμέτρου

Πραγματικά ορίσματα (2)

- Οι πραγματικές παράμετροι ή ορίσματα αντικαθιστούν τις τυπικές παραμέτρους με πραγματικές τιμές
- Πρέπει να υπάρχει ***συμβατότητα τύπων***
- Ισχύουν όσα έχουμε πει για το typecast (μικρότεροι τύποι μετατρέπονται αυτόματα, μεγαλύτεροι χρειάζονται ρητό typecast)

Τυπικές παράμετροι

- Γενικότερα, στις μεθόδους οι τιμές των παραμέτρων εισόδου μπορούν να περάσουν με τρεις τρόπους:
  - κατά τιμή (***by value***),
  - κατ' αναφορά (***by reference***)
  - κατά τιμή και αναφορά (***by value and reference***)
- Τους τρόπους by reference και by value and reference θα τους συζητήσουμε σε επόμενα κεφάλαια

Πέρασμα κατά τιμή

- Όταν οι τιμές μεταβλητές περνάνε κατά τιμή (by value), τότε **δημιουργούνται τοπικά αντίγραφα** των μεταβλητών και **όλες οι πράξεις γίνονται με αυτά τα τοπικά αντίγραφα, όχι με τις πραγματικές μεταβλητές**
- Επομένως, κατά την έξοδο από τη μέθοδο οι πραγματικές μεταβλητές δεν περιέχουν τις αλλαγές που έχουν γίνει (στα τοπικά αντίγραφα)

Java πέρασμα κατά τιμή

- Η Java περνάει τις τυπικές παραμέτρους πρωταρχικών τύπων **πάντα κατά τιμή**
- Ας δούμε ένα παράδειγμα: Θέλουμε μία μέθοδο που να λαμβάνει ως είσοδο δύο παραμέτρους, έστω int x, int y και να ανταλλάσσει αμοιβαία τις τιμές τους, δηλ., το x να γίνεται y και το y να γίνεται x (βλ. επόμενη διαφάνεια)

swap


```
22      /**
23       * Swaps a, b.
24       *
25       * @param a      the first int
26       * @param b      the second int
27       */
28      public static void swap(int a, int b) {
29          int tmp = a;
30          a = b;
31          b = tmp;
32      }
33  }
```

- Η swap λαμβάνει ως είσοδο δύο ακεραίους a, b.
- Στο σώμα της με τη χρήση μίας ενδιάμεσης μεταβλητής tmp, ανταλλάσσει αμοιβαία τα a, b

swap – πέρασμα κατά τιμή


```java
package gr.aueb.cf.ch5;

/**
 * Χρησιμοποιεί τη μέθοδο swap() για
 * να ανταλλάξει αμοιβαία τις τιμές των a, β.
 * Ο τρόπος αυτός αμοιβαίας ανταλλαγής δε
 * δουλεύει γιατί τα a, b περνάνε ως input
 * (readonly) και όχι ως output.
 *
 */
public class SwapApp {

    public static void main(String[] args) {
        int a = 5;
        int b = 7;

        swap(a, b);

        System.out.printf("a = %d, b = %d",a, b);
    }
```

- Στο σώμα της main, καλούμε την swap με a=5 και b=7.
- Θα περιμέναμε η printf να εκτυπώσει a=7, b=5. Όμως οι τιμές που εκτυπώνει η printf είναι α=5, b=7.
- Αυτό συμβαίνει γιατί όταν καλούμε τη swap στη γραμμή 17, οι τιμές a , b της main() αντιγράφονται στις τυπικές παραμέτρους a, b αντίστοιχα της swap
- Μέσα στη swap τα a, b της swap ανταλλάσσουν πράγματι τις τιμές τους. ***Τα a, b όμως της swap είναι τοπικές μεταβλητές***. Έξω από τη swap ***οι αλλαγές στα a, b δεν ισχύουν*** ενώ τα a, b της main() διατηρούν τις τιμές τους, όπως φαίνεται από τα αποτελέσματα της printf

Υπερφόρτωση μεθόδων


```java
package gr.aueb.than.ch5;

/**
 * Δείχνει τη χρήση του method overloading,
 * δηλ. μέδοδοι με ΙΔΙΟ ΟΝΟΜΑ αλλά διαφορετικές
 * παραμέτρους.
 *
 * @author A. Androutsos
 */
public class MethodOverload {

    public static void main(String[] args) {
        int a = 5, b = 15;

        int sum1, sum3;
        long sum2, sum4;

        sum1 = add(1, 2);
        sum2 = add(1L, 2L);
        sum3 = add(1, 2, 12);
        sum4 = add(1, 2, 12L);

        System.out.println(sum1 + " " + sum2 + " " + sum3 + " " + sum4);
    }
```

- Μπορούμε να έχουμε μεθόδους με το ***ίδιο όνομα αλλά διαφορετικές τυπικές παραμέτρους***?
- Ναι, με method overloading

Υπογραφή μεθόδων


```
26  @      public static int add(int a, int b) {
27             return a + b;
28         }
29
30  @      public static long add(long a, long b) {
31             return  a + b;
32         }
33
34  @      public static int add(int a, int b, int c) {
35             return a + b + c;
36         }
37
38  @      public static long add(int a, int b, long c) {
39             return a + b + c;
40         }
41 }
```

- Method Overloading = Μέθοδοι με διαφορετική υπογραφή

Η ***υπογραφή (signature) μιας μεθόδου*** είναι ***το όνομά της μαζί με τις τυπικές της παραμέτρους***. Ο επιστρεφόμενος τύπος δεν επηρεάζει την υπογραφή της μεθόδου.

Το method overloading είναι απλά μέθοδοι με ίδιο όνομα αλλά διαφορετική υπογραφή.

Άρτιοι και Περιττοί (1)


```java
package gr.aueb.cf.ch5;

import java.util.Scanner;

/**
 * Ορίζει τις μεθόδους isEven και isOdd
 * για τον έλεγχο άρτιων και περιττών
 * ακεραίων.
 */
public class EvenOddApp {

    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int a = 0;

        System.out.println("Please insert an int");
        a = in.nextInt();

        System.out.printf("Ο %d%sείναι άρτιος",a,  isEven(a) ? " " : " δεν ");
    }
```

- Ελέγχει τον αριθμό που δίνει ο χρήστης αν είναι άρτιος ή περιττός και εμφανίζει κατάλληλο μήνυμα

Άρτιοι και Περιττοί (2)


```
22      /**
23       * Evaluates a as even.
24       *
25       * @param a      the int to evaluate
26       * @return       true, if is even, false otherwise
27       */
28      public static boolean isEven(int a) {
29          return (a % 2) == 0;
30      }
31
32      /**
33       * Evaluates a as odd.
34       *
35       * @param a      the int to evaluate
36       * @return       true, if is even, false otherwise
37       */
38      public static boolean isOdd(int a) {
39          return !isEven(a);
40      }
41  }
```

- Ορίζουμε δύο μεθόδους, την ***isEven(int a)*** και την ***isOdd(int a)***
- Και οι δύο μέθοδοι παίρνουν ως είσοδο ένα ακέραιο και επιστρέφουν true ή false
- Η isEven(int a) ελέγχει αν ο αριθμός διαιρείται ακριβώς με το 2, οπότε είναι άρτιος
- Η isOdd(int a) ελέγχει επιστρέφει το αντίθετο της isEven()

Αναδρομή

- Όταν μία μέθοδος ***καλεί μέσα στο σώμα της τον εαυτό της***, αυτό καλείται αναδρομή
- Συνήθως μέσα στο σώμα καλούμε την μέθοδο ***με μικρότερο μέγεθος δεδομένων***, διαιρούμε δηλ. το πρόβλημα σε μικρότερα μέρη (διαίρει και βασίλευε – Divide & Conquer)

Παραγοντικό αναδρομικό


```java
package gr.aueb.than.ch5;

public class FactorialRecursive {

    public static void main(String[] args) {
        int n = 4;
        System.out.println(factorial(n));
    }

    public static int factorial(int n) {
        if ( n <= 1) {
            return 1;
        } else {
            return n * factorial(n-1);
        }

        /* Πιο σύντομα θα μπορούσαμε να γράψουμε:
         * return (n <= 1) ? 1 : (n * factorial(n-1));
         */
    }
}
```

- Παρατηρούμε ότι η factorial καλεί κάθε φορά τον εαυτό της **με μικρότερο μήκος δεδομένων (n-1)**
- Πρέπει ωστόσο να υπάρχει μία τελική συνθήκη. Αν το n είναι 1 ή 0 επιστρέφουμε 1

Η βασική ιδέα εδώ είναι πως το παραγοντικό(n) μπορεί να εκφραστεί ως n * παραγοντικό(n-1), το παραγοντικό(n-1) = (n-1) * παραγοντικό(n-2), κοκ.

Δύναμη αναδρομική


```java
package gr.aueb.than.ch5;

/**
 * Ορίζει την power(int a, int n) που
 * υπολογίζει το a^n.|
 *
 * @author A. Androutsos
 */
public class PowerRecursive {

    public static void main(String[] args) {
        int a = 2, n = 4;
        System.out.println(power(a, n));
    }

    /**
     * Υπολογίζει το a^n με αναδρομή.
     *
     * @param a η βάση
     * @param n η δύναμη
     * @return το a^n
     */
    public static int power(int a, int n) {
        if (n == 0) {
            return 1;
        } else {
            return a * power(a, n-1);
        }
    }
}
```

- Η ιδέα εδώ είναι πως η δύναμη a^n = a * a^n-1 = a * a * a^n-2, κοκ.
- Επομένως κάθε φορά ***μειώνεται το μέγεθος της εισόδου στην αναδρομική κλήση*** power(a, n-1)
- Όταν n == 0 τότε η δύναμη του a^0 είναι 1

Μικρή εργασία (cont.)

- Δημιουργήστε μία εφαρμογή που να εμφανίζει ένα μενού με 6 επιλογές:

  1. Εμφάνισε n αστεράκια οριζόντια
  2. Εμφάνισε n αστεράκια κάθετα
  3. Εμφάνισε n γραμμές με n αστεράκια
  4. Εμφάνισε n γραμμές με αστεράκια 1 – n
  5. Εμφάνισε n γραμμές με αστεράκια n – 1
  6. Έξοδος από το πρόγραμμα

  Δώσε επιλογή:

- Και μετά: Δώστε αριθμό για αστεράκια

Μικρή εργασία

- Κάθε μία από τις πέντε πρώτες επιλογές θα πρέπει να **υλοποιηθεί με μέθοδο**
- Ο χρήστης θα δίνει ένα αριθμό για αστεράκια
- Οι επιλογές 3, 4, και 5 θα πρέπει να υλοποιηθούν με τη βοήθεια της μεθόδου της επιλογής 1
- Το πρόγραμμα θα πρέπει να τρέχει μέχρι ο χρήστης να επιλέξει την Επιλογή 6