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

Τύποι Δεδομένων
Ο τύπος Ακέραιος

Αθ. Ανδρούτσος

Παράδειγμα - printf


```java
package gr.aueb.cf.chapter2;

/**
 * Προσθέτει δύο ακεραίους και εμφανίζει το
 * αποτέλεσμα στην κονσόλα (standard output)
 *
 * @author αβαπα
 */
public class AddApp {

    public static void main(String[] args) {

        // Δήλωση και αρχικοποίηση μεταβλητών
        int num1 = 12;
        int num2 = 5;
        int sum = 0;

        // Εντολές
        sum = num1 + num2;

        // Εκτύπωση αποτελεσμάτων
        System.out.println("Το άθροισμα των: " + num1 + " + " + num2 + " = " + sum );
        System.out.printf("Το άθροισμα των: %d + %d = %d", num1, num2, sum);
    }
}
```

- Η **printf** χρησιμοποιεί placeholders (π.χ. **%d** συμβολίζει ακέραιο) και τους αντιστοιχεί στη λίστα μεταβλητών που ακολουθεί

Στις αρχικοποιήσεις, εκχωρήσεις τιμών και παραστάσεις, ανάμεσα στα σύμβολα (τελεστές) = , + κλπ. και τις μεταβλητές ***μπορούν να υπάρχουν κενά διαστήματα***
Αυτό έχει να κάνει με το readability την εύκολη δηλαδή αναγνωσιμότητα του κώδικα και την συντήρησή του. ***Ο κώδικας είναι πιο αναγνώσιμος όταν υπάρχουν κενά ανάμεσα στα σύμβολα***
Όταν θέλουμε να εκτυπώσουμε και κείμενο και μεταβλητές μαζί μπορούμε να χρησιμοποιούμε την **printf** αντί για πολλαπλά + στην println

Ο τύπος δεδομένων Ακέραιος

- Περιλαμβάνει τον τύπο **int** και 3 υποτύπους (`byte`, `short`, `long`) που είναι ακέραιοι διαφορετικού μεγέθους
  - `int` (4 bytes – 32 bits)
  - `byte` (1 byte – 8 bits)
  - `short` (2 bytes – 16 bits)
  - `long` (8 bytes – 64 bits)

Εύρος τιμών (Min Max values) ακεραίων


```java
package gr.aueb.cf.ch2;

public class IntApp {

    public static void main(String[] args) {
        System.out.printf("Type: %s, Size: %d, Min: %,d, Max: %,d\n",
                Integer.TYPE, Integer.BYTES, Integer.MIN_VALUE, Integer.MAX_VALUE);
        System.out.printf("Type: %s, Size: %d, Min: %d, Max: %d\n",
                Byte.TYPE, Byte.BYTES, Byte.MIN_VALUE, Byte.MAX_VALUE);
        System.out.printf("Type: %s, Size: %d, Min: %,d, Max: %,d\n",
                Short.TYPE, Short.BYTES, Short.MIN_VALUE, Short.MAX_VALUE);
        System.out.printf("Type: %s, Size: %d, Min: %,d, Max: %,d\n",
                Long.TYPE, Long.BYTES, Long.MIN_VALUE, Long.MAX_VALUE);

```

```
Run:  IntApp
"C:\Program Files\Amazon Corretto\jdk11.0.10_9\bin\java.exe" "-javaagent:C:\Program
Type: int, Size: 4, Min: -2.147.483.648, Max: 2.147.483.647
Type: byte, Size: 1, Min: -128, Max: 127
Type: short, Size: 2, Min: -32.768, Max: 32.767
Type: long, Size: 8, Min: -9.223.372.036.854.775.808, Max: 9.223.372.036.854.775.807
```

- Με **%d** συμβολίζουμε ακέραιους
- Με **%s** τα αλφαριθμητικά
- Το **%,d** με κόμμα δηλαδή ενδιάμεσα σημαίνει να εμφανιστούν οι ακέραιοι με διαχωριστικό χιλιάδων ώστε οι μεγάλοι ακέραιοι να είναι πιο ευανάγνωστοι

Wrapper Κλάσεις


```java
package gr.aueb.cf.ch2;

public class IntApp {

    public static void main(String[] args) {
        System.out.printf("Type: %s, Size: %d, Min: %d, Max: %d\n",
                Integer.TYPE, Integer.BYTES, Integer.MIN_VALUE, Integer.MAX_VALUE);
        System.out.printf("Type: %s, Size: %d, Min: %d, Max: %d\n",
                Byte.TYPE, Byte.BYTES, Byte.MIN_VALUE, Byte.MAX_VALUE);
        System.out.printf("Type: %s, Size: %d, Min: %d, Max: %d\n",
                Short.TYPE, Short.BYTES, Short.MIN_VALUE, Short.MAX_VALUE);
        System.out.printf("Type: %s, Size: %d, Min: %d, Max: %d\n",
                Long.TYPE, Long.BYTES, Long.MIN_VALUE, Long.MAX_VALUE);

```

```
un:  IntApp ×
"C:\Program Files\Amazon Corretto\jdk11.0.10_9\bin\java.exe" "-javaagent:C:\Program
Type: int, Size: 4, Min: -2147483648, Max: 2147483647
Type: byte, Size: 1, Min: -128, Max: 127
Type: short, Size: 2, Min: -32768, Max: 32767
Type: long, Size: 8, Min: -9223372036854775808, Max: 9223372036854775807
```

- Για κάθε πρωταρχικό τύπο ακεραίων, υπάρχει μία αντίστοιχη Wrapper κλάση (κλάση περιτυλίγματος) που **ξεκινάει με κεφαλαίο γράμμα** (Byte, Short, Integer, Long) και μας παρέχει τον τύπο δεδομένων (TYPE), το μέγεθος σε SIZE ή BYTES, τις ελάχιστες και μέγιστες τιμές (MIN_VALUE, MAX_VALUE) για κάθε τύπο ακεραίων byte, short, int και long αντίστοιχα

Ακέραια Literals

- int num = 12; // Το ***12*** θεωρείται int (by default τα ακέραια literals είναι 32 bits)
- long num = 12L; // Το 12 είναι int αλλά με type suffix L μετατρέπεται σε long, 64 bits
- Όταν οι ακέραιες σταθερές είναι μεγάλος αριθμός, μπορούμε να διαχωρίζουμε με κάτω παύλα ώστε να φαίνεται καλύτερα (readability)

```
public static void main(String[] args) {
    long longNum = 123_456_789;
}
```

Παραστάσεις ακεραίων

- Για να κάνουμε πράξεις μεταξύ ακεραίων η Java μας παρέχει αριθμητικούς τελεστές
- Έτσι μία παράσταση ακεραίων είναι μία ακολουθία τελεσταίων (μεταβλητών ακεραίων ή ακεραίων literals) και τελεστών (+, -, *, /, κλπ.) όπως θα δούμε αναλυτικά στην επόμενη διαφάνεια

Αριθμητικοί Τελεστές

		Πράξη	Αποτέλεσμα
• ***+***	Άθροισμα	• 7 + 5	12
• ***-***	Διαφορά	• 7 – 5	2
• *****	Γινόμενο	• 7 * 5	35
• ***/***	(ακέραιο) πηλίκο	• 7 / 5	1
• ***%***	(ακέραιο) υπόλοιπο	• 7 % 5 2	
• ***++***	αύξηση κατά 1	• 7++	8
• ***--***	Μείωση κατά 1	• 7--	6

- Οι πράξεις με ακεραίους δίνουν πάντα αποτέλεσμα ακέραιο
- Ο τελεστής ***/*** αποκόπτει το κλασματικό μέρος
- Ο τελεστής ***%*** (που διαβάζεται ***mod***) είναι το υπόλοιπο της διαίρεσης 7 / 5 δηλ. ένας ***θετικός ακέραιος*** που όταν αφαιρεθεί από τον 7 τον κάνει ακέραιο πολλαπλάσιο του 5
- Στους μοναδιαίους τελεστές ++, -- δεν αφήνουμε κενά μεταξύ αριθμού και τελεστή

Παραστάσεις Ακεραίων (1)

- Όπως αναφέραμε οι παραστάσεις (Expressions) αποτελούνται από **Τελεστές** (*+, -, *, /, %, ++, --*) και **Τελεσταίους** (*μεταβλητές ή αριθμοί* στους οποίους εφαρμόζονται οι τελεστές)
- Κάθε παράσταση ακεραίων επιστρέφει ακέραιο. Για παράδειγμα η διαίρεση 5 / 2 επιστρέφει την τιμή 2 (και όχι 2.5)
- ***Διαίρεση με το 0 προκαλεί σφάλμα***
- Οι byte/short σε μια παράσταση μετατρέπονται σε int προτού υπολογιστεί η παράσταση

Παραστάσεις Ακεραίων (2)

- Αν υπάρχει έστω κι ένας long σε μία παράσταση, όλοι μετατρέπονται σε long
- Αν ένα αποτέλεσμα βρίσκεται έξω από την περιοχή τιμών του τύπου του, τότε συμβαίνει υπερχείλιση (overflow)
- Σε περίπτωση υπερχείλισης η Java δεν δίνει μήνυμα λάθους, αλλά **δίνει λάθος αποτελέσματα !**

Αποτίμηση παραστάσεων

- Σειρά εκτέλεσης πράξεων ανάλογα με την προτεραιότητα
- Πρώτα πολλαπλασιασμοί και διαιρέσεις και μετά προσθέσεις και αφαιρέσεις
- Για έλεγχο της παράστασης καλύτερα να χρησιμοποιούμε **παρενθέσεις**
- Οι παρενθέσεις έχουν την υψηλότερη προτεραιότητα και τότε η σειρά εκτέλεσης είναι από μέσα προς τα έξω και έχουμε καλύτερο έλεγχο της παράστασης

Προτεραιότητα Τελεστών

Precedence	Operator
1	( )
2	++, --
3	/, * , %
4	+, -
5	=, += , -=, *=, /=, %=

- Στον πίνακα αριστερά οι τελεστές κατατάσσονται κατά φθίνουσα σειρά προτεραιότητας
- Όταν οι τελεστές έχουν ίδια προτεραιότητα εκτελούνται από αριστερά προς τα δεξιά
- Όπως αναφέραμε είναι προτιμότερο να χρησιμοποιούμε παρενθέσεις για τον καλύτερο έλεγχο των προτεραιοτήτων

Παράδειγμα

- `int i = 2 + 5 * 8 – 1`, το αποτέλεσμα θα είναι 41, γιατί ***πρώτα θα εκτελεστεί ο πολ/σμός που έχει υψηλότερη προτεραιότητα***, μετά η πρόσθεση που έχει ίση προτεραιότητα με την αφαίρεση, αλλά προηγείται, και στο τέλος η αφαίρεση
- Θα ήταν καλύτερα ωστόσο αν το γράφαμε ως `int i = 2 + (5 * 8) – 1`
- Αν θέλουμε να υπολογιστεί το άθροισμα και η αφαίρεση και να πολλαπλασιαστούν μεταξύ τους, τότε χρησιμοποιούμε παρενθέσεις, ως εξής:

  `int i = (2 + 5)*(8 – 1)`. Το αποτέλεσμα θα είναι: 49
- Επίσης, θα μπορούσε να είναι: `int i = ((2 + 5) * 8) – 1`, που δίνει αποτέλεσμα 55

Expressions (1)

ExpressionsApp.java

```
1  package gr.aueb.cf.ch2;
2
3  public class ExpressionsApp {
4
5      public static void main(String[] args) {
6          int num1 = 12;
7          int num2 = 5;
8          int sum = 0, sub = 0;
9          int div = 0, mul = 0;
10         int mod = 0;
11
12         int result1 = 0;
13         int result2 = 0;
14         int result3 = 0;
15         int result4 = 0;
16
17         sum = num1 + num2;
18         sub = num1 - num2;
19         div = num1 / num2;
20         mul = num1 * num2;
21         mod = num1 % num2;
```

- Ακολουθούν μερικά παραδείγματα expressions με τη χρήση αριθμητικών τελεστών
- Παρατηρήστε καταρχάς ότι σε μια γραμμή μπορούμε να κάνουμε πολλές δηλώσεις χωρισμένες με κόμμα
- Τα expressions αποτελούνται από το αριστερό μέρος, το = και μετά το δεξί μέρος που εκτελείται η πράξη και εκχωρείται

Expressions (2)


```
23          result1 = num1++;
24          result2 = ++num1;
25          result3 += num1;
26
27          result4 = (num1 + num2) * ((num1 / 2) / (num1 % 5));
28
29          System.out.printf("sum: %d\t, sub: %d\t div: %d\t, mul: %d\t mod: %d\n", sum, sub, div, mul, mod);
30          System.out.printf("result1: %d, result2: %d, result3: %d, result4: %d", result1, result2, result3, result4);
31      }
32  }
```

- Στο result1, πρώτα εκχωρείται το num1 στο result1 και μετά γίνεται το num1++
- Στο ++num1 πρώτα γίνεται το ++ και μετά η εκχώρηση στο result2
- Υπάρχουν συντμήσεις για πράξεις όπως result3 = result3 + num1 ώστε να μη γράφουμε δύο φορές το result3, όπως result3 += num1
- Οι συντμήσεις ισχύουν για όλους τους τελεστές, π.χ. sum *= 3; Αντί για sum = sum * 3, κλπ.

Expressions (3)


```
23          result1 = num1++;
24          result2 = ++num1;
25          result3 += num1;
26
27          result4 = (num1 + num2) * ((num1 / 2) / (num1 % 5));
28
29          System.out.printf("sum: %d\t, sub: %d\t div: %d\t, mul: %d\t mod: %d\n", sum, sub, div, mul, mod);
30          System.out.printf("result1: %d, result2: %d, result3: %d, result4: %d", result1, result2, result3, result4);
31      }
32  }
```

- Οι τελεστές έχουν προτεραιότητες και επίσης τελεστές ίσης προτεραιότητας έχουν left associativity (εκτελούνται από αριστερά προς τα δεξιά)
- Για να μην έχουμε θέματα προτεραιοτήτων προτείνεται να χρησιμοποιούμε παρενθέσεις που έχουν την μεγαλύτερη προτεραιότητα και κάνουν τον κώδικά μας readable

Overflow


```java
package gr.aueb.cf.ch2;

/**
 * Προσθέτει δύο ακεραίους ενώ το αποτέλεσμα
 * δημιουργεί overflow (υπερχείλιση)
 *
 * @author a8ana
 */
public class AddApp {

    public static void main(String[] args) {
        // Δήλωση και αρχικοποίηση μεταβλητών
        int num1 = 2_147_483_647;
        int num2 = 2;
        int result = 0;

        // Εντολές
        result = num1 + num2;

        // Εκτύπωση αποτελέσματος
        System.out.println("Το αποτέλεσμα είναι: " + result);
    }
}
```

```
Run:  AddApp
"C:\Program Files\Amazon Corretto\jdk1
Το αποτέλεσμα είναι: -2147483647

Process finished with exit code 0
```

- Το αποτέλεσμα είναι απλά λάθος
- Δεν υπάρχει compile ή runtime error
- Επομένως πρέπει να είμαστε προσεκτικοί και να χρησιμοποιούμε άλλους τύπους δεδομένων για big integers

BigInteger (1)

AddBigIntsApp.java

```
1  package gr.aueb.cf.ch2;
2
3  import java.math.BigInteger;
4
5  public class AddBigIntsApp {
6
7      public static void main(String[] args) {
8          BigInteger bigNum1 = new BigInteger("2147483647");
9          BigInteger bigNum2 = new BigInteger("2147483647");
10         BigInteger result = bigNum1.add(bigNum2);
11
12         System.out.printf("%,d", result);
13     }
14 }
15
```

Run: AddBigIntsApp

```
"C:\Program Files\Amazon Corretto\jdk11.0.10_9\bin\java.exe
4.294.967.294
Process finished with exit code 0
```

- Η BigInteger είναι μία έτοιμη κλάση της Java που βρίσκεται στο package java.math
- Το πλήρες όνομά της είναι java.math.BigInteger αλλά για να μπορούμε να την χρησιμοποιούμε απλά ως BigInteger πρέπει να κάνουμε *import* την κλάση αυτή με το πλήρες όνομά της (FQN – Fully Qualified Name)

BigInteger (2)

AddBigIntsApp.java

```
1  package gr.aueb.cf.ch2;
2
3  import java.math.BigInteger;
4
5  public class AddBigIntsApp {
6
7      public static void main(String[] args) {
8          BigInteger bigNum1 = new BigInteger("2147483647");
9          BigInteger bigNum2 = new BigInteger("2147483647");
10         BigInteger result = bigNum1.add(bigNum2);
11
12         System.out.printf("%,d", result);
13     }
14 }
15
```

Run: AddBigIntsApp

```
"C:\Program Files\Amazon Corretto\jdk11.0.10_9\bin\java.exe
4.294.967.294
Process finished with exit code 0
```

- Μπορούμε να δημιουργήσουμε ένα Big integer με new BigInteger() και την τιμή ως αλφαριθμητικό
- Οι κλάσεις γενικά προσφέρουν λειτουργίες (μεθόδους) που μπορούμε να καλούμε με τον τελεστή τελεία (.)
- Όπως η μέθοδος add() που προσθέτει δύο big integers

Καλούπωμα Τύπων – Typecast

- Όταν αριστερά και δεξιά σε μια εκχώρηση υπάρχουν διαφορετικοί τύποι δεδομένων, πρέπει ο τύπος στα δεξιά να μετατραπεί στον τύπο αριστερά
- Ο λόγος της ασυμβατότητας είναι τα διαφορετικά μεγέθη των τύπων δεδομένων
- Για παράδειγμα όταν προσπαθούμε να εκχωρήσουμε long που είναι 64-bit σε int που είναι 32-bit υπάρχει ασυμβατότητα ή byte που είναι 8-bit σε short που είναι 16-bit πάλι υπάρχει ασυμβατότητα ή int που είναι 32-bit σε long που είναι 64-bit πάλι υπάρχει ασυμβατότητα κλπ.

Καλούπωμα Τύπων

- Σε αυτές τις περιπτώσεις, που υπάρχουν διαφορετικοί τύποι δεδομένων δεξιά και αριστερά σε μία εκχώρηση, για να γίνει η εκχώρηση πρέπει να γίνει ***typecast*** δηλαδή ο δεξιά τύπος να αλλάξει τύπο άρα και μέγεθος, ώστε να είναι συμβατός με τον αριστερό
- Οι κανόνες για typecast είναι οι εξής:
  - Γίνεται ***αυτόματο typecast***, όταν η παράσταση στο δεξί μέρος της εκχώρησης είναι του ίδιου ή μικρότερου μεγέθους τύπου με το αριστερό μέρος
  - Το αντίθετο, δηλαδή εκχώρηση μεγαλύτερου τύπου σε μικρότερο δεν γίνεται αυτόματα, οπότε αν θέλουμε να κάνουμε typecast ***πρέπει να το κάνουμε ρητά***
- Ρητό typecast -> Βάζουμε τον τύπο μέσα σε παρενθέσεις `(type)`

π.χ.: `int a = (int) b;`

Typecast

- Γενικώς, ***οποτεδήποτε έχουμε εκχώρηση μεταξύ διαφορετικών τύπων χρειάζεται typecast***
  - Όπως είπαμε, αν εκχωρούμε μικρότερου μεγέθους τύπο σε μεγαλύτερου μεγέθους τύπο (π.χ. byte σε int) ***τότε το typecast γίνεται αυτόματα από την Java, γιατί δεν έχουμε απώλεια δεδομένων***
  - Το αντίθετο, δηλαδή εκχώρηση μεγαλύτερου τύπου σε μικρότερο (π.χ. int σε short) ***χρειάζεται ρητά να γίνει typecast, γιατί έχουμε απώλεια δεδομένων***

Typecast int - long


```java
package gr.aueb.cf.ch2;

/**
 * Typecast Demo.
 */
public class TypecastApp {

    public static void main(String[] args) {
        int num1 = 10;
        long num2 = 20L;

        num1 = num2;
    }
}
```

TypecastApp.java

```java
package gr.aueb.cf.ch2;

/**
 * Typecast Demo.
 */
public class TypecastApp {

    public static void main(String[] args) {
        int num1 = 10;
        long num2 = 20L;

        num1 = (int) num2;
    }
}
```

- Χωρίς typecast το Type System της γλώσσας δίνει error γιατί ο num2 έχει μεγαλύτερο μέγεθος (64-bit) από τον num1 (32-bit)
- Αν κάνουμε typecast δεν δημιουργείται συντακτικό λάθος, αλλά μπορεί να έχουμε απώλεια πληροφορίας

Typecast error-prone

TypecastApp.java

```java
package gr.aueb.cf.ch2;

/**
 * Typecast Demo.
 */
public class TypecastApp {

    public static void main(String[] args) {
        int num1 = 10;
        long num2 = 20L;

        num1 = (int) num2;
    }
}
```

```java
package gr.aueb.cf.ch2;

/**
 * Typecast Demo.
 */
public class TypecastApp {

    public static void main(String[] args) {
        int num1 = 10;
        int num2 = 20;

        num1 = num2;
    }
}
```

- Επειδή το typecast είναι γενικά error-prone, μπορεί δηλαδή να δημιουργήσει λάθη, πρέπει να χρησιμοποιείται πολύ περιορισμένα
- Στο παράδειγμα θα μπορούσαμε να χρησιμοποιήσουμε int αντί για long, σε όλες τις ακέραιες μεταβλητές. Γενικά, καλύτερα να χρησιμοποιούμε int όταν θέλουμε ακέραιους ή long όταν έχουμε μεγάλους ακέραιους και να μην χρησιμοποιούμε ρητά typecast

Typecast – Type suffix (1)

TypeSuffixApp.java

```
1  package gr.aueb.cf.ch2;
2
3  /**
4   * Type suffix Demo.
5   */
6  public class TypeSuffixApp {
7
8      public static void main(String[] args) {
9          /*
10          * int literals are integers (32-bit )
11          * by default. In order to typecast to
12          * long we have to insert a type suffix L or l.
13          */
14         long num1 = 20L;
15
16         /*
17          * For short and byte data types
18          * no type-suffixes are provided.
19          */
20         short num2 = 12;
21         byte b1 = 2;
22     }
23 }
```

- Όπως αναφέραμε όλα τα integer literals θεωρούνται int, δηλαδή αριθμοί 32-bits. Αν δώσουμε ένα αριθμό όπως το 20, από το JVM θεωρείται int
- Παρότι το typecast μπορεί να γίνει αυτόματα, για να ‘διευκολύνουμε’ το JVM, μπορούμε να κάνουμε typecast το 20 σε long, με suffix L ή l (προτιμότερο το L, γιατί το l συγχέεται με το 1)

Typecast – Type suffix (2)


```java
package gr.aueb.cf.ch2;

/**
 * Type suffix Demo.
 */
public class TypeSuffixApp {

    public static void main(String[] args) {
        /*
         * int literals are integers (32-bit )
         * by default. In order to typecast to
         * long we have to insert a type suffix L or l.
         */
        long num1 = 2147483648L;
```

- Όταν έχουμε τιμές μεγαλύτερες από το μέγεθος του int, πρέπει αναγκαστικά να κάνουμε typecast διαφορετικά το Type System δίνει error

Έξοδος ακεραίων (1)

- Έστω ***int num1 = 10;***
- Μπορούμε να εκτυπώσουμε το num1 με τρεις τρόπους:

System.out.print(num1); // Εκτυπώνει 10

System.out.println(num1); // Εκτυπώνει 10
// και μία αλλαγή γραμμής

System.out.printf("%d", num1); // Εκτυπώνει 10
// Το %d σημαίνει int και αντικαθίσταται με το num1

Έξοδος ακραίων (2)

- Μπορούμε να εκτυπώνουμε ακεραίους μαζί με κείμενο (συμβολοσειρές)
- Τις συμβολοσειρές τις βάζουμε μέσα σε “ ... ”
- Οι παραστάσεις ακεραίων μαζί με συμβολοσειρές συνενώνονται με τον τελεστή συνένωσης: **+**

Έξοδος ακεραίων (3)


```
1  package gr.aueb.cf;
2
3  /**
4   * Prints integers and text
5   *
6   * @author thanos
7   * @version 0.1
8   *
9   */
10 public class IntPrintDemo {
11
12     public static void main(String[] args) {
13
14         int i = 12;
15
16         System.out.print(i);
17         System.out.println(i);
18         System.out.printf("%d", i);
19
20         System.out.print("Η τιμή του i είναι: " + i);
21         System.out.println("Η τιμή του i είναι: " + i);
22         System.out.printf("Η τιμή του i είναι: %d", i);
23     }
24 }
```

- Στο 1ο γκρουπ των print, println, printf ***εκτυπώνεται μόνο ο ακέραιος i***
- Στο 2ο γκρουπ ***συνενώνουμε (concatenate) αλφαριθμητικά με το i με τη χρήση του τελεστή +*** εκτός της τελευταίας περίπτωσης όπου έχουμε ***αντικατάσταση (substitution) του %d από το i***
- Το αποτέλεσμα είναι ο υπολογισμός των παραστάσεων και η εμφάνιση των τιμών τους ως συμβολοσειρών στην κονσόλα

Έξοδος Ακεραίων (4)


```
22          System.out.printf("Η τιμή του i είναι: %d", i);
```

- Στην τελευταία περίπτωση της printf έχουμε αντικατάσταση μεταβλητών ***(variable substitution)***
- Δηλαδή έχουμε placeholders όπως το %d μέσα στα 'αυτάκια' του αλφαριθμητικού. Οι placeholders αντικαθίστανται από τις αντίστοιχες τιμές των μεταβλητών
- Αυτό επιτρέπει ***καλύτερο και ευκολότερο formatting του τελικού αποτελέσματος*** σε σχέση με την συνένωση (concatenation) με τον τελεστή + της println, ιδιαίτερα όταν έχουμε πολλές συνενώσεις

Έξοδος Ακεραίων Παράδειγμα με println

- Στο παρακάτω παράδειγμα υπολογίζουμε και εμφανίζουμε το i + j
- Στην εντολή ***println*** του παραδείγματος το ***+ εμφανίζεται 6 φορές***

```
1  package gr.aueb.cf;
2
3  /**
4   * Calculates and prints the sum of i , j
5   *
6   * @author thanos
7   */
8  public class IntPrint2Demo {
9
10     public static void main(String[] args) {
11
12         int i = 10, j = 5;
13         int sum = 0;
14
15         sum = i + j;
16
17         System.out.println("Το άθροισμα των " + i + " και " + j + " είναι: " + sum + ".");
18     }
19 }
```

Έξοδος Ακεραίων Παράδειγμα με printf


```
1  package gr.aueb.cf;
2
3  /**
4   * Calculates and prints the sum of i , j
5   *
6   * @author thanos
7   */
8  public class IntPrint2Demo {
9
10     public static void main(String[] args) {
11
12         int i = 10, j = 5;
13         int sum = 0;
14
15         sum = i + j;
16
17         System.out.println("Το άθροισμα των " + i + " και " + j + " είναι: " + sum + ".");
18         System.out.printf("Το άθροισμα των %d και %d είναι: %d.%n", i, j, sum);
19     }
20 }
```

- Όπως βλέπουμε στις περιπτώσεις που έχουμε κείμενο και μεταβλητές, η printf() είναι προτιμότερη
- Το ***%n*** στην printf() είναι η αλλαγή γραμμής (βλ. επόμενη διαφάνεια)

Console

```
<terminated> IntPrint2Demo [Java Application] C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe (28 Ιουλ 2019, 12:20:24 μ.μ.)
Το άθροισμα των 10 και 5 είναι: 15.
Το άθροισμα των 10 και 5 είναι: 15.
```

Σύμβολο αλλαγής γραμμής

- Η αλλαγή γραμμής είναι και αυτή ένας χαρακτήρας, απλά δεν τον βλέπουμε (βλέπουμε μόνο τον κέρσορα, που μετακινείται στην επόμενη γραμμή)
- Στα συστήματα Linux και στα Mac ο χαρακτήρας αλλαγής γραμμής συμβολίζεται ως **\n** (ASCII 10, Unicode \u000a)
- Στα συστήματα Windows συμβολίζεται ως **\r\n** (ASCII 13 και 10 , Unicode \u000d και \u000a)
- Στα παλιά Mac συμβολίζεται ως **\r** (ASCII 13 , Unicode \u000d)
- Για να μπορούμε στην Java να εκτυπώνουμε ***αλλαγές γραμμής σε αρχεία*** που να είναι ***συμβατές με όλα τα συστήματα*** μπορούμε να χρησιμοποιούμε το ***%n***
- *Για αλλαγή γραμμής στο standard input (Οθόνη) μπορούμε να χρησιμοποιούμε το \n*

Η μέθοδος printf() …


```
22          System.out.printf("Η τιμή του i είναι: %d", i);
```

- Η printf() είναι μία γενικής χρήσης μέθοδος διαμόρφωσης εξόδου, όπου κάθε **%** δείχνει τη θέση που θα καταλάβει κάθε ένα από τα ορίσματα (1ο, 2ο, 3ο, κλπ.) Στο παράδειγμα υπάρχει μόνο ένα όρισμα, το i
- Με την printf() μπορούμε επίσης να μορφοποιήσουμε την έξοδο. Μπορούμε για παράδειγμα να προσθέσουμε στο %d το πλάτος πεδίου, π.χ. **%4d** και οι εμφανιζόμενοι αριθμοί να είναι στοιχισμένοι δεξιά μέσα στο πεδίο τους με κενά διαστήματα αριστερά αν ο αριθμός έχει λιγότερα από 4 ψηφία
- Επίσης, αν προσθέσουμε το 0 πριν τον αριθμό, π.χ. **%04d** τότε αντί για κενά διαστήματα, εμφανίζονται μηδενικά. Έτσι, για παράδειγμα, αν το 5 συμβόλιζε τον μήνα και είχαμε %02d θα εμφανιζόταν ως 05.

Η μέθοδος printf()

%d	Εμφάνιση ως ακέραιος
%6d	Εμφάνιση ως ακέραιος με πλάτος τουλάχιστον 6 διαστήματα. Ο αριθμός στοιχίζεται δεξιά και τα διαστήματα που δεν έχουν ψηφία μένουν κενά
%06d	Εμφάνιση ως ακέραιος με πλάτος τουλάχιστον 6 διαστήματα. Ο αριθμός στοιχίζεται δεξιά και τα διαστήματα που δεν έχουν ψηφία συμπληρώνονται από το 0
%,d	Εμφάνιση ως ακέραιος. Εμφανίζεται διαχωριστικό χιλιάδων αν ο αριθμός είναι μεγαλύτερος ή ίσος από 1000 (το διαχωριστικό χιλιάδων είναι το κόμμα ή η τελεία ανάλογα με τις ρυθμίσεις του Η/Υ μας. Στα Ελληνικά Windows είναι η τελεία, στα Αγγλικά είναι το κόμμα)
%,02d	Εμφάνιση ως ακέραιος. Εμφανίζεται διαχωριστικό χιλιάδων αν ο αριθμός είναι μεγαλύτερος ή ίσος από 1000 (το διαχωριστικό χιλιάδων είναι το κόμμα ή η τελεία ανάλογα με τις ρυθμίσεις του Η/Υ μας. Στα Ελληνικά Windows είναι η τελεία, στα Αγγλικά είναι το κόμμα). Εμφάνιση ως ακέραιος με πλάτος τουλάχιστον 2 διαστήματα και left padding με 0 (αν είναι μονοψήφιος)

Η έτοιμη κλάση Math

- Η Java μας παρέχει ένα έτοιμο τύπο, την κλάση Math που μπορούμε να χρησιμοποιούμε για ευκολία όταν θέλουμε κάποιες μαθηματικές συναρτήσεις
- Η κλάση Math βρίσκεται μέσα σε μία βιβλιοθήκη (package) της Java, το java.lang
- Η Oracle μας παρέχει τεκμηρίωση για όλους τους τύπους (κλάσεις, κλπ.) στα λεγόμενα Javadocs. Δείτε για παράδειγμα
- https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/Math.html

Κλάση Math


```java
package gr.aueb.cf.chapter2;

/**
 * Χρησιμοποιεί την έτοιμη κλάση Math.
 */
public class MathDemo {

    public static void main(String[] args) {
        System.out.println("Το απόλυτο του -5 είναι: " + Math.abs(-5));
        System.out.println("Το απόλυτο του -5 είναι: " + Math.min(1, 10));
        System.out.println("Το απόλυτο του -5 είναι: " + Math.max(1, 10));
    }
}
```

- Παρατηρήστε το dot notation με τον τελεστή τελεία για τις λειτουργίες της Math. Για παράδειγμα, το **Math.abs()** σημαίνει ότι η **abs()** είναι μία λειτουργία (σσ. μέθοδος) **που ανήκει στην κλάση Math** (επιστρέφει το απόλυτο ενός αριθμού που παίρνει ως παράμετρο μέσα στις παρενθέσεις)
- Η **τελεία δηλαδή είναι τελεστής εμβέλειας** και το πλήρες όνομα της abs() είναι Math.abs()

Κλάση Math - random

- Η μέθοδος **Math.random()** επιστρέφει ένα ψευδοτυχαίο δεκαδικό αριθμό μεταξύ 0 – 0.99, δηλ. 0 <= randomNum < 1
- Για να μετατρέψουμε το διάστημα μπορούμε να πολλαπλασιάσουμε. Για παράδειγμα να πολ/σουμε με 10, δηλαδή Math.random() * 10, τότε το διάστημα γίνεται 0 <= randomNum < 9.99
- Οπότε αν εκχωρήσουμε το αποτέλεσμα της Math.random() σε ένα int με typecast τότε ο int θα έχει μία τυχαία τιμή μεταξύ 0 και 9 μιας και το δεκαδικό μέρος αποκόπτεται όταν εκχωρούμε δεκαδικό αριθμό σε int με typecast
- Αν θέλουμε να ξεκινάμε, όχι από το 0 αλλά από μεγαλύτερο ακέραιο, π.χ. το 1, μπορούμε να έχουμε (Math.random() * 10) + 1, οπότε τότε το range είναι 1 – 10 αντί για 0 - 9
- Γενικά αν θέλουμε αριθμούς σε ένα range **min – max** , τότε έχουμε **Math.random() * ((max – min) + 1) + min**. Αν θέλουμε για παράδειγμα αριθμούς 1 – 6 (ζάρια) τότε έχουμε Math.random() * 6 + 1

Random - ζάρι


```java
package gr.aueb.cf.chapter2;

/**
 * Χρησιμοποιεί την Math.random() που παράγει
 * τυχαίους δεκαδικούς αριθμούς μεταξύ 0 - 0.9,
 * δηλαδή, 0 <= randomNum < 1, για να μας δώσει
 * τυχαίες τιμές μεταξύ 1 - 6, όπως δηλαδή ένα ζάρι
 */
public class RandomDemo {

    public static void main(String[] args) {
        int die = (int) (Math.random() * 6) + 1;
        System.out.println(die);
    }
}
```

- Το max – min + 1 είναι **6**, αφού το max είναι 6, το min είναι 1 (= 6 - 1 + 1)
- Ενώ το min είναι 1
- Άρα η κλήση είναι Math.random() * 6 + 1

Εισαγωγή δεδομένων

- Στο προηγούμενο παράδειγμα δώσαμε τις ***τιμές των μεταβλητών μέσα στον κώδικα***
- Κάτι τέτοιο δεν είναι πολύ χρήσιμο γιατί δεν υπάρχει αλληλεπίδραση με τον χρήστη
- Θα ήταν χρήσιμο αν υπήρχε αλληλεπίδραση με τον χρήστη ώστε να δίνει ο χρήστης διαφορετικά δεδομένα, δηλ. ***να μπορεί το πρόγραμμα να διαβάζει δεδομένα από το πληκτρολόγιο (standard input)***

Είσοδος ακεραίου με Scanner

- Για είσοδο ακεραίων από το πληκτρολόγιο κατά την εκτέλεση του προγράμματος χρησιμοποιούμε τον έτοιμο τύπο της Java, **Scanner**
- θα πρέπει να δηλώσουμε μία ***μεταβλητή τύπου Scanner*** και να την συνδέσουμε με το πληκτρολόγιο (standard input) που στην Java είναι το ***System.in***

```
Scanner in = new Scanner(System.in);
```

- Ο τελεστής ***new δημιουργεί ένα αντικείμενο*** τύπου ***Scanner*** και το συσχετίζει με το ***System.in*** δηλ. το πληκτρολόγιο
- Έτσι μπορούμε να χρησιμοποιούμε την ***μεταβλητή με όνομα `in`*** για να διαβάζουμε από το πληκτρολόγιο

Είσοδος ακεραίων με Scanner

- Για να διαβάσουμε ακέραιους χρησιμοποιούμε τη μέθοδο ***`in.nextInt()`*** που διαβάζει ένα (τον επόμενο) ακέραιο από το πληκτρολόγιο
- Ο συμβολισμός ***in.nextInt()*** με την τελεία (dot notation) σημαίνει, όπως έχουμε αναφέρει, πως η nextInt() ανήκει στην μεταβλητή Scanner, in

Scanner

- Η κλάση Scanner είναι έτοιμη κλάση της Java που βρίσκεται στο package **java.util** Επομένως το πλήρες όνομά της είναι **java.util.Scanner**
- Για να μπορούμε μέσα στον κώδικά μας να την χρησιμοποιούμε απλώς ως Scanner χωρίς να χρειάζεται να δίνουμε το πλήρες όνομα java.util.Scanner, δίνουμε στην αρχή την οδηγία: ***import java.util.Scanner;***

Παράδειγμα - Άθροισμα


```java
package gr.aueb.cf.ch2;

import java.util.Scanner;

/**
 * Scanner Demo. Reads two ints from standard
 * input (keyboard) and gets/prints the sum.
 */
public class ScannerApp {

    public static void main(String[] args) {
        // Δήλωση (declaration) μεταβλητών
        Scanner in = new Scanner(System.in);
        int num1, num2, sum;

        // Εντολές
        System.out.println("Please insert two ints");
        num1 = in.nextInt();
        num2 = in.nextInt();

        sum = num1 + num2;

        // Εμφάνιση αποτελεσμάτων
        System.out.println(num1 + "+" + num2 + "=" + sum);
    }
```

- Δηλώνουμε τρεις ακεραίους, num1, num2 και sum καθώς και τον Scanner in
- Στη συνέχεια, δίνουμε μήνυμα προτροπής, διαβάζουμε με **nextInt()**, υπολογίζουμε το sum και εκτυπώνουμε
- Παρατηρήστε το import java.util.Scanner ώστε να μπορούμε να χρησιμοποιούμε τον Scanner χωρίς το πλήρες όνομά του

Σφάλματα Προγραμματισμού

- Συντακτικά & Σημασιολογικά λάθη
- Συντακτικά ή γραμματικά οφείλονται στην παραβίαση των συντακτικών κανόνων της γλώσσας java
- Για παράδειγμα ξεχνάμε ένα άγκιστρο, ή μία παρένθεση, ή γράφουμε λάθος έναν identifier ή μία εντολή

Συντακτικά λάθη

- Ο μεταγλωττιστής καθώς γράφουμε κοκκινίζει τα σημεία όπου υπάρχουν συντακτικά λάθη. Αν **βάλουμε το ποντίκι πάνω στο λάθος** εμφανίζεται το μήνυμα λάθους
- Στο παρακάτω παράδειγμα ***έχουμε ξεχάσει τα «αυτάκια» στο κείμενο της println()*** και το μήνυμα λάθους είναι ότι υπάρχουν περισσότερα από ένα λάθη (multiple markers) σε αυτή τη γραμμή. Τα μηνύματα μπορεί να μην είναι πάντα 100% αντιπροσωπευτικά του λάθους (άλλες φορές είναι) αλλά σε κάθε περίπτωση δίνουν μια ένδειξη για το λάθος

```
19  Multiple markers at this line                          στε δύο ακεραίους);
20    - Παρακαλώ cannot be resolved to a variable
21    - ακεραίους cannot be resolved to a variable
22    - Syntax error, insert ";" to complete Statement
23    - Syntax error, insert ")" to complete MethodInvocation
24    - Syntax error on token "δύο", ( expected          των %d και %d είναι %d", i, j, sum);
25          in.close();
26      }
27  }
```

Σημασιολογικά Σφάλματα (1)

- Σφάλματα που αφορούν κυρίως το Type System της γλώσσας
- Για παράδειγμα αν προσθέσουμε ακέραιο με αλφαριθμητικό, αν διαιρέσουμε ακέραιο με το μηδέν, κλπ.
- Τα λογικά λάθη κάποιες φορές δεν εντοπίζονται από τον μεταγλωττιστή και εμφανίζονται κατά το χρόνο εκτέλεσης από το run time της γλώσσας

Έλεγχος προγράμματος

- Οι προγραμματιστές πρέπει πάντα να εκτελούν το πρόγραμμά τους με διαφορετικά δοκιμαστικά δεδομένα (test data) και να συγκρίνουν με τα αναμενόμενα αποτελέσματα που έχουν υπολογίσει από πριν
- Αν τα αναμενόμενα αποτελέσματα δεν συμφωνούν με τα πραγματικά αποτελέσματα τότε υπάρχει λογικό λάθος

Δομή Προγραμμάτων


```
1  package gr.aueb.cf.chapter2;
2
3  public class AdderDemo {
4
5      public static void main(String[] args) {
6
7          // Δήλωση και Αρχικοποίηση Μεταβλητών
8          int num1 = 10;
9          int num2 = 20;
10         int sum = 0;
11
12         // Εντολές
13         sum = num1 + num2;
14
15         // Εκτύπωση αποτελεσμάτων
16         System.out.println("Το άθροισμα των " + num1 + " και " + num2 + " είναι " + sum);
17         System.out.printf("Το άθροισμα των %d και %d είναι %d", num1, num2, sum);
18     }
19 }
```

(1)

(2)

(3)

- Όπως ίσως παρατηρήσατε, σε όλα τα προγράμματα που έχουμε γράψει μέχρι τώρα και θα γράψουμε στη συνέχεια, ακολουθούμε μία συγκεκριμένη **δομή**, που αποτελείται από τρία (3) βασικά μέρη:

1. **Δήλωση και αρχικοποίηση μεταβλητών**
2. **Εντολές**
3. **Εκτύπωση (δηλ. εμφάνιση) αποτελεσμάτων**

Αυτή τη δομή θα πρέπει να την ακολουθούμε κατά τη συγγραφή προγραμμάτων!

Παράδειγμα - Μετατροπή Ευρώ σε Δολάρια US


```java
import java.util.Scanner;

/**
 * Euro To USD Converter.
 */
public class EuroUsdConverter {

    public static void main(String[] args) {

        Scanner in = new Scanner(System.in);
        int inputEuros;
        final int PARITY = 99;  //USA cents
        int totalUsaCents, usaCents, usaDollars;

        System.out.println("Please insert an amount (in Euros)");
        inputEuros = in.nextInt();

        totalUsaCents = inputEuros * PARITY;
        usaDollars = totalUsaCents / 100;
        usaCents = totalUsaCents % 100;

        System.out.printf("%d euros = %d usa cents = %d usa dollars & %d usa cents",
                inputEuros, totalUsaCents, usaDollars, usaCents);
    }
}
```

- ***Σταθερές***: Αντί να έχουμε μέσα στον κώδικά μας τον αριθμό 99, που θα ήταν σαν ‘μαγικός αριθμός’, δημιουργούμε μία σταθερά με όνομα που έχει νόημα (PARITY).

Διαίρεση και υπόλοιπο

- Η διαίρεση και το υπόλοιπο ***(το div και το mod) λειτουργούν 'συνεργατικά'***
- Η μεν διαίρεση επιστρέφει τα μεγαλύτερης τάξης ψηφία, το δε mod τα μικρότερης τάξης ψηφία.
- Αν για παράδειγμα έχουμε ένα αριθμό, τον num = 517 και ***θέλουμε να χωρίσουμε σε εκατοντάδες, δεκάδες και μονάδες***, δηλ. 5 εκατοντάδες, 1 δεκάδα και 7 μονάδες, τότε ξεκινώντας από την μεγαλύτερη τάξη, που είναι οι εκατοντάδες μπορούμε να κάνουμε ***num / 100 (αυτό μας κάνει 5) και να πάρουμε τις 5 εκατοντάδες*** και ***num % 100 και να πάρουμε τις εναπομένουσες μονάδες (δηλαδή 17 και έστω ότι αποθηκεύουμε το 17 στην μεταβλητή remaining)***
- Στην συνέχεια αν κάνουμε ***remaining / 10 (κάνει 1)*** θα πάρουμε τις δεκάδες και ***remaining % 10 (=7)*** θα πάρουμε τις μονάδες

Παράδειγμα – Μετατροπή δευτερολέπτων (1)


```java
package gr.aueb.cf.chapter2;

import java.util.Scanner;

/**
 * Διαβάζει από τον χρήση ένα ακέραιο που αντιστοιχεί στο
 * πλήθος δευτερολέπτων και μετατρέπει σε ημέρες, ώρες, λεπτά
 * και δευτερόλεπτα και εκτυπώνει το αποτέλεμα στην οθόνη
 */
public class SecondsDemo {

    public static void main(String[] args) {

        // Δήλωση και αρχικοποίηση μεταβλητών
        Scanner in = new Scanner(System.in);
        final int DAY = 24 * 3600;
        final int HOUR = 3600;
        final int MINUTE = 60;
        int inputSeconds = 0;
        int days = 0, hours = 0, minutes = 0;
        int remainingSeconds = 0;
```

- Δηλώνουμε σταθερές και μεταβλητές

Παράδειγμα – Μετατροπή δευτερολέπτων (2)


```
23          // Εντολές
24          System.out.println("Please insert the count of Seconds");
25          inputSeconds = in.nextInt();
26          remainingSeconds = inputSeconds;
27
28          days = remainingSeconds / DAY;
29          remainingSeconds = remainingSeconds % DAY;
30
31          hours = remainingSeconds / HOUR;
32          remainingSeconds = remainingSeconds % HOUR;
33
34          minutes = remainingSeconds / MINUTE;
35          remainingSeconds = remainingSeconds % MINUTE;
36
37          // Εκτύπωση αποτελεσμάτων
38          System.out.printf("Input Seconds: %d = %d Days, %d Hours, %d Minutes, %d Seconds",
39                  inputSeconds, days, hours, minutes, remainingSeconds);
40
41      }
42  }
```

- Ξεκινάμε διαβάζοντας με ***in.nextInt()*** από τον χρήστη, το πλήθος των δευτερολέπτων

Στη συνέχεια βρίσκουμε τις μέρες με διαίρεση με το 3600 * 24 (που το έχουμε κάνει σταθερά) και τα εναπομείναντα δευτερόλεπτα με mod
Με παρόμοιο τρόπο βρίσκουμε με τη σειρά, τις ώρες, τα λεπτά και απομένουν και τα εναπομείναντα δευτερόλεπτα

Παράδειγμα – Μετατροπή σε δευτερόλεπτα


```java
import java.util.Locale;
import java.util.Scanner;

/**
 * Μετατρέπει μέρες, ώρες, λεπτά και δευτερόλεπτα
 * σε συνολικά δευτερόλεπτα.
 *
 * @author A. Androutsos
 */
public class DHMToSec {

    public static void main(String[] args) {
        final int SEC_PER_DAY = 3600 * 24;
        final int SEC_PER_HOUR = 3600;
        final int SEC_PER_MIN = 60;

        Scanner sc = new Scanner(System.in);
        long days, hours, mins, secs;
        long totalSec;

        System.out.println("Δώστε Μέρες, Ώρες, Λεπτά, Δευ/πτα");
        days = sc.nextLong();
        hours = sc.nextLong();
        mins = sc.nextLong();
        secs = sc.nextLong();

        totalSec = (days * SEC_PER_DAY) + (hours * SEC_PER_HOUR) + (mins * SEC_PER_MIN) + (secs);

        System.out.printf(Locale.US, "Total Sec: %,8d", totalSec);
    }
}
```

- Αφού διαβάσουμε ώρες, μέρες, λεπτά και δεύτερα από τον χρήστη, υπολογίσουμε με προσοχή και παρενθέσεις τον συνολικό αριθμό των δευτερολέπτων
- Εκτυπώνουμε με printf() χρησιμοποιώντας ***Locale.US*** ώστε το διαχωριστικό χιλιάδων που υπάρχει στην ***%,d*** να είναι το κόμμα και όχι η τελεία

Locale για Ελληνικά


```
1   package gr.aueb.cf.chapter2;
2
3   import java.util.Locale;
4
5   public class TypeSuffixDemo {
6
7       public static void main(String[] args) {
8           System.out.println(Integer.MAX_VALUE);
9           long num = 2_147_483_648L;  //2_147_483_647 is the MAX_INT
10          System.out.printf(Locale.forLanguageTag("el"), "%,d", num);
11      }
12  }
```

- Το Locale για ελληνικά μπορούμε να το πάρουμε με τον τρόπο στο παράδειγμα, ώστε να χρησιμοποιηθεί ως διαχωριστικό χιλιάδων η τελεία

Euros App (1)

- Θα αναπτύξουμε μία εφαρμογή που μπορεί να λειτουργήσει σαν ένα ATM μηχάνημα που ανάλογα το ποσό ανάληψης δίνει αντίστοιχα χαρτονομίσματα 500, 100, 50, 20, 10 ευρώ καθώς και κέρματα ενός ευρώ

Euros App (2)


```java
package gr.aueb.cf.ch2;

import java.util.Scanner;

public class EurosApp {

    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int inputEuros;
        int euros500, euros100, euros50, euros20, euros10, euros5, euros1;
        int remainingEuros;

        System.out.println("Please insert the amount (in Euros)");
        inputEuros = in.nextInt();

        remainingEuros = inputEuros;

        euros500 = remainingEuros / 500;
        remainingEuros = remainingEuros % 500;

        euros100 = remainingEuros / 100;
        remainingEuros = remainingEuros % 100;

```

- Ξεκινάμε από την ανώτερη τάξη που είναι τα Euro-500 και συνεχίζουμε σταδιακά στα Euro-100, Euro-50, Euro-20, Euro-10 και Euro-1, όπου ***κάθε φορά βρίσκουμε με διαίρεση το αντίστοιχο πλήθος και με mod το εναπομείναν υπόλοιπο*** *remainingEuros*

Euros App (3)


```
24          euros50 = remainingEuros / 50;
25          remainingEuros = remainingEuros % 50;
26
27          euros20 = remainingEuros / 20;
28          remainingEuros = remainingEuros % 20;
29
30          euros10 = remainingEuros / 10;
31          remainingEuros = remainingEuros % 10;
32
33          euros5 = remainingEuros / 5;
34          remainingEuros = remainingEuros % 5;
35
36          System.out.printf("%d Euros = euros500: %d, euros100: %d, euros50: %d\n",
37                  inputEuros, euros500, euros100, euros50);
38          System.out.printf("euros20: %d, euros10: %d, euros5: %d, euros: %d",
39                  euros20, euros10, euros5, remainingEuros);
40
41      }
42  }
```

- Τελικά εκτυπώνουμε

Παράδειγμα – Εύρεση τετραγώνου και κύβου


```
1  package gr.aueb.than.ch2;
2
3  import java.util.Scanner;
4
5  /**
6   * Υπολογίζει το τετράγωνο και τον κύβο ενός αριθμού
7   * που δίνει ο χρήστης. Χρησιμοποιεί την έτοιμη μέθοδο
8   * Math.pow() της κλάσης Math.
9   *
10  * @author A. Androutsos
11  */
12 public class MathPowers {
13
14     public static void main(String[] args) {
15         Scanner sc = new Scanner(System.in);
16         int num;
17
18         System.out.println("Παρακαλώ δώστε ένα αριθμό");
19         num = sc.nextInt();
20
21         System.out.printf("Square: %d, Cube: %d", (int) Math.pow(num, 2),
22                 (int) Math.pow(num, 3));
23
24     }
25 }
```

- Η ***Math.pow()*** παίρνει δύο ορίσματα, τον αριθμό και την δύναμη που τον υψώνουμε και επιστρέφει την δύναμη του αριθμού. Για παράδειγμα η ***Math.pow(num, 2)*** επιστρέφει το τετράγωνο του num, και η **Math.pow(num, 3)** επιστρέφει τον κύβο του num.
- Παρατηρήστε επίσης στην printf() ότι κάνουμε ***typecast σε int*** γιατί η Math.pow() επιστρέφει δεδκαδικό (double) καθώς επίσης και έχουμε χωρίσει την εντολή σε δύο γραμμές διακόπτοντας μετά το κόμμα και αφήνοντας 8 κενά στην επόμενη γραμμή

Άθροισμα Ψηφίων Ακεραίου


```java
package gr.aueb.cf.ch2;

/**
 * Βρίσκει τα ψηφία ενός διψήφιου αριθμού
 * και τα προσθέτει.
 */
public class IntDigits {

    public static void main(String[] args) {
        int num = 19;
        int leftDigit = 0;
        int rightDigit = 0;
        int sum = 0;

        leftDigit = num / 10;
        rightDigit = num % 10;

        sum = leftDigit + rightDigit;

        System.out.printf("LeftDigit: %d, RightDigit: %d, Sum: %d", leftDigit, rightDigit, sum);
    }
}
```

- Με % 10 μπορούμε να πάρουμε το τελευταίο ψηφίο και με / 10 μπορούμε να πάρουμε όλο το μέρος του αριθμού εκτός το τελευταίο ψηφίο. Στο παρόν παράδειγμα, στη συνέχεια προσθέτουμε

Μέγιστος τριών ακεραίων


```java
package gr.aueb.cf.ch2;

import java.util.Scanner;

/**
 * Βρίσκει τον μέγιστο τριών ακεραίων
 * που δίνει ο χρήστης.
 */
public class MaxOfThree {

    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int num1 = 0;
        int num2 = 0;
        int num3 = 0;

        System.out.println("Please insert three nums to find the max");
        num1 = in.nextInt();
        num2 = in.nextInt();
        num3 = in.nextInt();

        System.out.printf("Max between %d, %d, %d is %d",
                num1, num2, num3, Math.max(num1, Math.max(num2, num3)));
    }
}
```

- Καλούμε την Math.max με παραμέτρους τον ένα ακέραιο και την Math.max των άλλων δύο ακεραίων

Δύο ζάρια


```java
package gr.aueb.cf.ch2;

/**
 * Ρίχνει μία τυχαία ζαριά με δύο ζάρια
 */
public class RandomDiceApp {

    public static void main(String[] args) {
        int die1 = 0;
        int die2 = 0;

        die1 = (int) (Math.random() * 6) + 1;
        die2 = (int) (Math.random() * 6) + 1;

        System.out.printf("Die1: %d, Die2: %d", die1, die2);
    }
}
```

- Καλούμε την Math.random() για δύο ζάρια από 1 - 6

Απλός υπολογιστής τσέπης


```java
package gr.aueb.cf.ch2;

import java.util.Scanner;

/**
 * Εκτελεί απλές αριθμητικές πράξεις σε δύο ακεραίους.
 */
public class MenuOpApp {

    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int num1 = 0;
        int num2 = 0;

        System.out.println("Please provide two ints");
        num1 = in.nextInt();
        num2 = in.nextInt();

        System.out.printf("Sum: %d\n", num1 + num2);
        System.out.printf("Sub: %d\n", num1 - num2);
        System.out.printf("Mul: %d\n", num1 * num2);
        System.out.printf("Div: %d\n", num1 / num2);
        System.out.printf("Mod: %d\n", num1 % num2);
    }
}
```

- Απλά εκτελεί αριθμητικές πράξεις σε δύο ακέραιους που δίνει ο χρήστης ως input

Ασκήσεις (1) cont.

- Θέλουμε να αναπτύξουμε ένα πρόγραμμα που να μετατρέπει ακέραιες θερμοκρασίες Φαρενάιτ (Fahrenheit) στην κλίμακα Κελσίου.
- Το πρόγραμμα θα εκτυπώνει κατάλληλο μήνυμα και θα διαβάζει στη συνέχεια την ακέραια θερμοκρασία από τον χρήστη
- Θα την μετατρέπει στην κλίμακα Κελσίου σύμφωνα με τον τύπο (5/9)(F-32)
- Επειδή όλα τα μέλη της παράστασης είναι int το αποτέλεσμα θα είναι int αποκόπτοντας τα δεκαδικά και δεν θα είναι ακριβές, αλλά αυτό δεν μας πειράζει σε αυτό το στάδιο

Ασκήσεις (1)

- Προσοχή στον τρόπο που θα δώσετε την παράσταση μετατροπής
- Αν δώσετε (5/9)(F-32) τότε επειδή 5/9 είναι 0 το τελικό αποτέλεσμα θα είναι 0 (επειδή οι 5 και 9 είναι int, το αποτέλεσμα της διαίρεσης 5/9 θα είναι int δηλ. 0)
- Καλύτερα λοιπόν να δώσουμε την παράσταση ως: 5 * (F-32) / 9 που είναι ακριβώς το ίδιο, αλλά αποφεύγουμε το 5/9 που δίνει 0

Ασκήσεις (2)

- Γράψτε ένα πρόγραμμα που διαβάζει από τον χρήστη 3 ακέραιους αριθμούς που αναπαριστούν ημέρα, μήνα, έτος και την εμφανίζει σε μορφή:

  HH/MM/ΕΕ

Π.χ. αν δώσουμε 5 12 2022 θα εμφανίσει 05/12/22