ΒΑΣΙΚΟΙ ΤΥΠΟΙ ΔΕΔΟΜΕΝΩΝ ΣΤΗ JAVA – ΤΕΛΕΣΤΕΣ - ΜΕΤΑΒΛΗΤΕΣ

Ας δούμε το απλό πρόγραμμα που υπολογίζει το εμβαδόν ενός τριγώνου. Το πρόγραμμα διαβάζει(εισάγει) την βάση και το ύψος του τριγώνου και υπολογίζει το εμβαδόν του. Θα χρησιμοποιήσει **μεταβλητές** για να αποθηκευτούν η βάση και το ύψος του τριγώνου καθώς και το εμβαδόν του τριγώνου που θα υπολογισθεί. Επίσης θα χρησιμοποιήσει την έκφραση: Εμβαδόν = Βάση x Ύψος/2, για τον υπολογισμό του εμβαδού του τριγώνου.

Ο αλγόριθμος για αυτό το πρόγραμμα μπορεί να περιγραφεί ως εξής:

1) Διάβασε την βάση του τριγώνου.
2) Διάβασε το ύψος του τριγώνου.
3) Υπολογίστε το εμβαδόν με την έκφραση: Emvadon = Basi x Ypsos / 2.
4) Εμφάνισε το αποτέλεσμα.

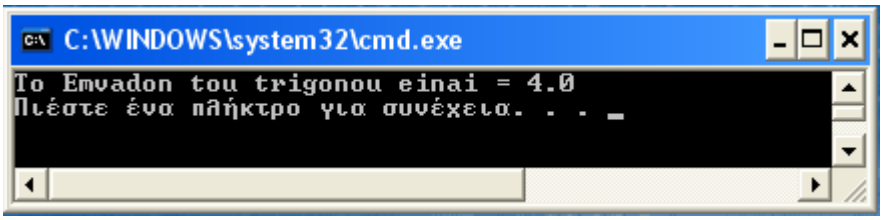
Το πρόγραμμα:

```
public class Emnadon_Trigonou
{
  public static void main(String args[])
  {
    int Basi   = 2;
    int Ypsos = 4;
    double Emvadon;

    //υπολογισμός του Εμβαδού του τριγώνου
    Emvadon = Basi * Ypsos / 2;

    //εμφάνιση του Εμβαδού του τριγώνου
    System.out.println("To Emvadon tou trigonou einai = " + Emvadon);
  }
}
```

Το αποτέλεσμα:



ΤΑ ΣΤΟΙΧΕΙΑ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ *(αναλυτικά στις επόμενες ενότητες)*

Χρησιμοποιήσαμε:

1) **Μεταβλητές: Basi, Ypsos, Emvadon**
2) **Τύπους δεδομένων: int, double**
3) **Τελεστές: =, *, / (στην έκφραση υπολογ. του Εμβαδού)**
4) **Εντολές**
5) **Σχόλια**
6) **Μπλοκ εντολών**

Μεταβλητές *(variables)*

Μεταβλητή, είναι ένας χώρος της μνήμης που θα πάρει τιμή μέσα στο πρόγραμμα ή κατά την εκτέλεση του προγράμματος. Η μεταβλητή μπορεί να αλλάξει τιμή μέσα στο πρόγραμμα. Πριν χρησιμοποιήσουμε μία μεταβλητή θα πρέπει να ορίσουμε το όνομα και τον τύπο των δεδομένων που θα λάβει. Η δήλωση της μεταβλητής έχει την μορφή:

<Τύπος δεδομένων> <Όνομα Μεταβλητής>;

Με την δήλωση μιας μεταβλητής μπορεί να καταχωρηθεί και η τιμή της μεταβλητής *(αρχικοποίηση της μεταβλητής)*:

<Τύπος δεδομένων> <Όνομα Μεταβλητής> = Τιμή;

Παραδείγματα:

```
int y;
int metritis=0;
double sum=0.0;
int x = 8, y = 13;
```

Ονόματα Μεταβλητών

Το όνομα της μεταβλητής αρχίζει με ένα γράμμα, την underscore (_) ή το $ και μπορεί να περιέχει συνδυασμούς γραμμάτων και αριθμών. Ένα πρόγραμμα αναφέρεται στην τιμή μιας μεταβλητής, χρησιμοποιώντας το όνομα της. Το όνομα της μεταβλητής πρέπει να αποτελείται από σειρές καταλλήλων Unicode χαρακτήρων. Μια μεταβλητή δεν πρέπει να έχει ως όνομα μία από τις δεσμευμένες λέξεις της Java:

Abstract	continue	for	new	switch
Assert	default	goto	package	synchronized
Boolean	do	if	private	this
Break	double	implements	protected	throw
Byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile
const	float	native	super	while

Το όνομα της μεταβλητής πρέπει να είναι μοναδικό μέσα στο μπλοκ προγράμματος που ορίζεται *(scope)*.

Δήλωση Σταθερών *(Constants)*

Χρησιμοποιώντας την δεσμευμένη λέξη **final** μπορούμε να δηλώσουμε σταθερές τιμές μέσα στο πρόγραμμα δηλαδή, τιμές που δεν θα αλλάξουν κατά την εκτέλεση του προγράμματος. Μία παράμετρος μπορεί να δηλωθεί ως final και αυτό σημαίνει ότι δεν θα αλλάξει τιμή στη μέθοδο που χρησιμοποιείται.

```
class YpologismosFPA
{
  public static void main ( String[] arg )
  {
    final double SYNTELESTIS1 = 0.06;
    final double SYNTELESTIS2 = 0.18;
    . . . . . .
  }
}
```

Τύποι Δεδομένων

Η Java υποστηρίζει δύο τύπους δεδομένων, τα **αντικείμενα** και τους **βασικούς** τύπους (βλ. Πίνακα 1). Στους βασικούς τύπους ανήκουν:

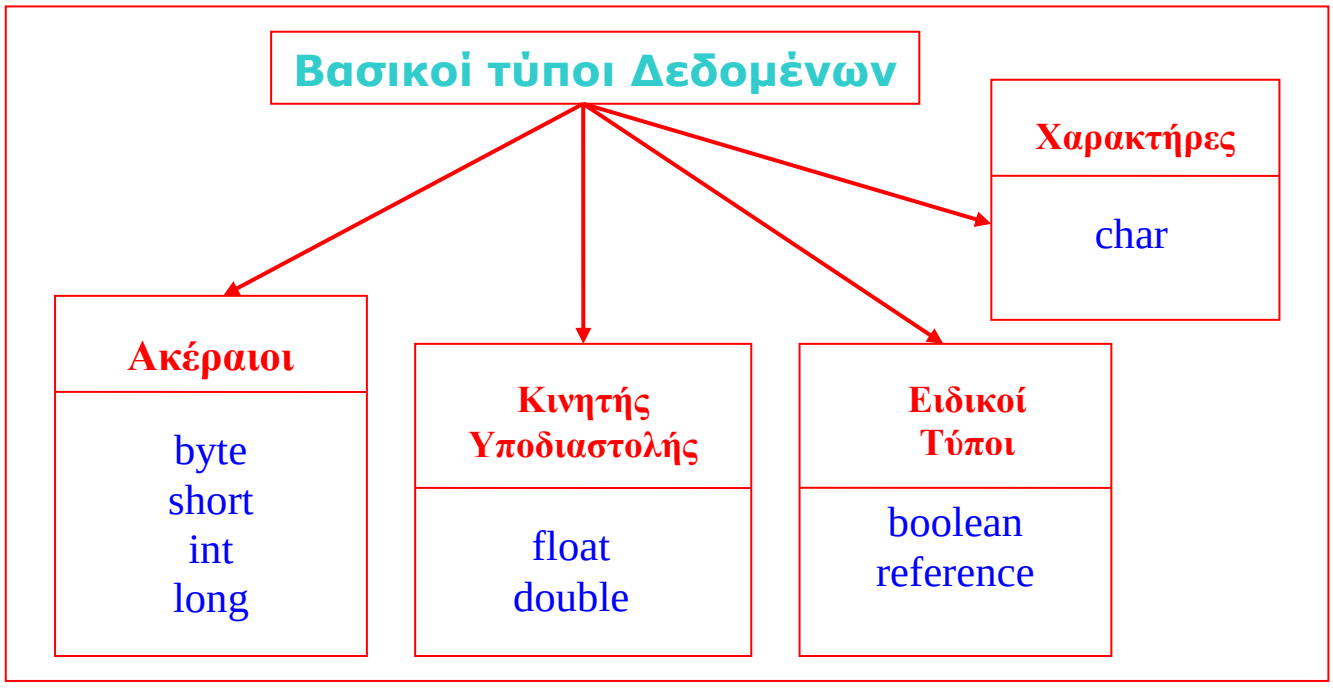
1) Ο αριθμητικός τύπος:

- τύπος ακεραίων (integer): **byte, short, int, long**
- τύπος πραγματικών ή κινητής υποδιαστολής (real): **float, double**

2) Ο τύπος χαρακτήρων (char): char

3) Άλλοι τύποι:

- Λογικός τύπος **boolean** *(true – false, λογικό αλήθεια ή ψέμα)*
- Η αναφορά σε αντικείμενο: **reference**



Πίνακας 1. Οι βασικοί τύποι δεδομένων της Java

Τύποι Ακεραίων (Integer)			
Τύπος	***Μέγεθος***	***Όρια τιμών(min – max)***	
byte	8 bits (1 byte)	-128	+127
short	16 bits (2 bytes)	-2^{15} (-32,768)	$+2^{15}$ - 1 (+32,768)
Int	32 bits (4 bytes)	-2^{31} (-2147483648)	$+2^{31}$ - 1 (+2147483648)
long	64 bits (8 bytes)	-2^{63} (-9223372036854775808) -9.223E18	$+2^{63}$ – 1 (+9223372036854775808) +9.223E18
Τύποι Κινητής Υποδιαστολής (Floating Point)			
Τύπος	***Μέγεθος***	***Όρια τιμών (min – max)***	
float	32 bits (4 bytes)	-3.4E+38 (7 δεκ. ψηφία)	+3.4E+38
double	64 bits (8 bytes)	-1.8E+308 (15 δεκ. ψηφία)	+1.8E+308
Τύπος χαρακτήρων			
char	16-bit	Ένας χαρακτήρας (ISO – Unicode). Από 0 ('\u0000') έως 65536 ('\uffff').	
Άλλοι τύποι			
boolean	1 bit	Μία λογική τιμή true, false	

Αρχικές τιμές των βασικών τύπων:

Τύπος	**Αρχική τιμή**
byte	0
short	0
Int	0
long	0L
float	0.0f
double	0.0d
char	'\u0000'
String (ή άλλο αντικείμενο)	null
boolean	false

Πέρα από τους οκτώ βασικούς τύπους η Java υποστηρίζει τις **συμβολοσειρές** *(Strings)* μέσω της κλάσης java.lang.String. Για τη δημιουργία ενός αντικειμένου String αρκεί να γράψουμε κείμενο μέσα σε διπλά εισαγωγικά, π.χ. **String s = " My Textx";** Για τις συμβολοσειρές θα αναφερθούμε σε επόμενο κεφάλαιο.

Χρήση ακεραίων αριθμών

- Γράφουμε ακέραιους: 123, 0, -34 (Integer literals, χωρίς υποδιαστολή).

- Οι ακέραιοι αριθμοί τύπου **int** μπορούν να γίνουν τύπου **long** προσθέτοντας στο τέλος τους ένα L ή l.

Ακρίβεια και χρήση πραγματικών αριθμών

- Ο τύπος δεδομένων double έχει ακρίβεια περίπου 15 - δεκαδικών ψηφίων. Ένας αριθμός που περιέχει υποδιαστολή ή εκθέτη θεωρείται τύπου double. Τέτοιου είδους αριθμοί είναι για παράδειγμα οι: **9.876, 2.35e-19**

- Ο τύπος δεδομένων float έχει ακρίβεια 7 - δεκαδικών ψηφίων

- Η μετατροπή ενός **double** αριθμού σε **float** γίνεται με την τοποθέτηση του 'F' ή του 'f ' στο τέλος του αριθμού. Π.χ. **2.567F ή 2.567f**

- Γράφουμε δεκαδικούς με τα δεκαδικά ψηφία να χωρίζονται με την τελεία:

```
123.0      -123.5      -198234.234      0.00000381
```

- Γράφουμε επίσης δεκαδικούς με επιστημονική μορφή *(scientific notation)* χρησιμοποιώντας το E δηλαδή, 10 εις τη δύναμη:

```
1.23E+02    -1.235E+02    -1.98234234E+05    3.81E-06
```

Για παράδειγμα ο αριθμός 1.2345E+03 είναι ο 1234.5 σε κανονική μορφή.

Ο βασικός τύπος απεικόνισης χαρακτήρων *(char)*

- Ο char είναι ένας 16-bit τύπος για την απεικόνιση Unicode - χαρακτήρων (χαρακτήρες όλων των γλωσσών, π.χ. Κινέζικων, Ελληνικών, κλπ.). Οι τιμές που λαμβάνει ο char είναι από 0 έως 65536. Ο κλασικός πίνακας των Λατινικών

χαρακτήρων, γνωστός και σαν ASCII-πίνακας, εξακολουθεί να υπάρχει στις θέσεις 0 έως 127, αλλά το εκτεταμένο σύνολο χαρακτήρων ISO-Latin-1 (8 - bit) καταλαμβάνει τις θέσεις 0 έως 255. Προσοχή στην απεικόνιση των χαρακτήρων, καθότι χαρακτήρες θεωρούνται και τα σημεία στίξης αλλά και το κενό διάστημα. Τα κεφαλαία και τα πεζά γράμματα θεωρούνται τελείως διαφορετικοί χαρακτήρες γι' αυτό πρέπει να προσέχουμε ιδιαίτερα κατά τη σύνταξη των προγραμμάτων. Ένας χαρακτήρας γράφεται μέσα σε μονούς αποστρόφους. Για παράδειγμα '**A'**, '**o'**.

- Υπάρχουν και οι ειδικοί χαρακτήρες που ονομάζονται χαρακτήρες ελέγχου ή χαρακτήρες διαφυγής και συντάσσονται με ειδικό τρόπο (χρησιμοποιώντας την πλάγια κάθετο (\):

Χαρακτήρας	Περιγραφή
\b	Οπισθοδρόμηση - *Backspace*
\t	Στηλοθέτης - *Horizontal tab*
\n	Αλλαγή γραμμής - *Linefeed*
\f	Αλλαγή σελίδας - *Form feed*
\r	Επαναφορά – *Carriage return*
\"	Διπλό εισαγωγικό - *Double quote*
\'	Μονό εισαγωγικό - *Single quote*
****	Ανάστροφη κάθετος - *Backslash*

Η ιδιομορφία του τύπου char

Σαν ακέραιος χρησιμοποιείται συνήθως στην εξαγωγή χαρακτήρων:

```
class TestChar {
  public static void main(String args[]) {
   char x=65;              //αποθήκευση ακεραίου αριθμού
   char c='A';             // αποθήκευση χαρακτήρα
   System.out.println(x);
   System.out.println(c);
  }
}
```

Θα εμφανίσει:

```
A
A
```

Ο βασικός τύπος λογικών συγκρίσεων *(boolean)*

Αντιπροσωπεύει **μία τιμή αλήθειας** ή **ψεύδους** σε συγκρίσεις μεταξύ όλων των τύπων των δεδομένων (**true** | **false**).

Τελεστές

Οι τελεστές είναι τα σύμβολα που λαμβάνουν μέρος στην τέλεση μιας λειτουργίας σε ένα, δύο ή και τρεις τελεστέους *(operators)*. Υπάρχουν οι **μοναδιαίοι** *(unary)* τελεστές που επενεργούν σε ένα και μόνο τελεστέο (π.χ. ++i), οι **δυαδικοί** τελεστές *(binary)* που επενεργούν σε δύο τελεστέους (π.χ. x1+x2) και οι **τριαδικοί** τελεστές *(ternary)* που επενεργούν σε τρεις τελεστέους (στη Java χρησιμοποιείται ο τελεστής **?** που θα δούμε στην εντολή if (π.χ. a>4?a-4:4).

Αριθμητικοί τελεστές

Τελεστής	Εκτέλεση Πράξης
*	Πολλαπλασιασμός (x1*x2)
/	Διαίρεση (x1/x2)
%	Υπόλοιπο (x1%x2)
+	Πρόσθεση (x1+x2)
-	Αφαίρεση (x1-x2)

Τελεστές προθέματος *(prefix)* και επιθέματος *(postfix)*

Οι τελεστές ++ και -- συντάσσονται είτε μπροστά (αριστερά) από την μεταβλητή (προ-αύξηση και προ-μείωση) είτε πίσω (δεξιά) από την μεταβλητή (μετά-αύξηση και μετά-μείωση).

Τελεστής	Χρήση	Περιγραφή
++	i++	Αυξάνει το i κατά 1; Σε μια παράσταση θα λάβει μέρος με την τιμή που ήδη έχει και μετά θα αυξηθεί κατά 1.
++	++i	Αυξάνει το i κατά 1; Σε μια παράσταση η τιμή της θα αυξηθεί κατά 1 και με τη νέα τιμή θα λάβει μέρος στις πράξεις.
--	i--	Μειώνει το i κατά 1; Σε μια παράσταση θα λάβει μέρος με την τιμή που ήδη έχει και μετά θα μειωθεί κατά 1.
--	--i	Μειώνει το i κατά 1; Σε μια παράσταση η τιμή της θα μειωθεί κατά 1 και με τη νέα τιμή θα λάβει μέρος στις πράξεις.

Σε μία παράσταση, προ-αύξηση ή προ-μείωση μιας μεταβλητής σημαίνει ότι ***η μεταβλητή θα αλλάξει τιμή και μετά θα υπολογιστεί η παράσταση με τη χρήση της νέας τιμής.***

Π.χ. **int i=10;**

j=++i; //η μεταβλητή j θα πάρει την τιμή 11

Μετά-αύξηση ή μετά-μείωση, σε μία παράσταση, σημαίνει ότι ***η παράσταση θα υπολογιστεί με την τιμή που έχει εκείνη τη στιγμή η μεταβλητή και μετά θα αυξηθεί ή μειωθεί η τιμή της.***

Π.χ. **int i=10;**

j=i++; //η μεταβλητή j θα πάρει την τιμή 10 και μετά η i θα γίνει 11

Τελεστές εκχώρησης

Ο τελεστής (**=**) είναι ο τελεστής καταχώρισης. Στο παράδειγμα x=10, η τιμή 10 καταχωρείται στη μεταβλητή x. Στην σχέση y=x+5, πρώτα θα υπολογιστεί η έκφραση x+5 και το αποτέλεσμα θα καταχωρηθεί στη μεταβλητή y. Άλλοι τελεστές απόδοσης τιμής είναι οι συντομεύσεις για απόδοση τιμής σε μεταβλητή με ταυτόχρονη αύξηση, μείωση κτλ. αυτής της μεταβλητής. Για παράδειγμα η έκφραση i += 10 είναι ταυτόσημη με την έκφραση: i = i+10. Αναλυτικά αυτοί οι τελεστές είναι:

Τελεστής	Χρήση	Ισοδυναμεί με:
+=	i += j	i = i + j
-=	i -= j	i = i - j
*=	i *= j	i = i * j

/=	i /= j	i = i / j
%=	i %= j	i = i % j
&=	i &= j	i = i & j
|=	i |= j	i = i | j
^=	i ^= j	i = i ^ j
<<=	i <<= j	i = i << j
>>=	i >>= j	i = i >> j
>>>=	i >>>= j	i = i >>> j

Προσοχή στην προτεραιότητα εκτέλεσης πράξεων

Η σειρά εκτέλεσης πράξεων σε μία έκφραση γίνεται από αριστερά προς τα δεξιά. Έτσι στη σχέση c = 2 * 6 + 16 / 4 θα παίρναμε πρώτα το 2 * 6 = 12, ύστερα το 12 + 16 = 28, μετά το 28 / 4 = 7 και θα το είχαμε σαν τελικό αποτέλεσμα στη μεταβλητή c. Στην πραγματικότητα το αποτέλεσμα είναι διαφορετικό (2 * 6 = 12, 16 / 4 = 4, 12 + 4 = 16 και τέλος c =16), αν δούμε τις προτεραιότητες στον παρακάτω πίνακα :

1η προτερ.:	( )	[ ]	.	
2η προτερ.:	++	--	~	!
3η προτερ.:	*	/	%	
4η προτερ. :	+	-		
5η προτερ. :	>>	>>>	<<	
6η προτερ. :	>	>=	<	<=
7η προτερ. :	==	!=		
8η προτερ. :	&			
9η προτερ. :	^			
10η προτερ. :	|			
11η προτερ. :	&&			
12η προτερ. :	||			
13η προτερ. :	?:			
14η προτερ. :	=op=			

Ασκήσεις με χρήση μεταβλητών και εκτέλεση αρ. πράξεων

Στο παρακάτω παράδειγμα:

- ορίζουμε δύο μεταβλητές τις ar1 και ar2
- στη συνέχεια δίνουμε τιμές στις μεταβλητές
- εκτελούμε απλές πράξεις
- εμφανίζουμε τα αποτελέσματα

```
class Example1 {
  public static void main(String args[]) {
    int ar1;            //Ορίζει τη μεταβλητή ar1 που θα πάρει ακέραια τιμή
    int ar2;            //Ορίζει τη μεταβλητή ar2 που θα πάρει ακέραια τιμή
    ar1 = 45;           // δίνουμε στη ar1 την τιμή 45
    ar2 = 20;           // δίνουμε στη ar2 την τιμή 20
    System.out.println("Arithmos1 : " + ar1);
    System.out.println("Arithmos2 : " + ar2);
    ar1 = ar1 * 2;
    ar2 = ar2 * 3;
    System.out.print("Apotelesma1 : ar1 * 2 = ");
    System.out.println(ar1);
    System.out.print("Apotelesma2 : ar2 * 3 = ");
    System.out.println(ar2);
  }
}
```

Αν εκτελέσουμε το πρόγραμμα θα πάρουμε τα αποτελέσματα:

```
Arithmos1 : 45
Arithmos2 : 20
Apotelesma1 : ar1 * 2 = 90
Apotelesma2 : ar2 * 3 = 60
```

Ένα ακόμη απλό παράδειγμα εκτέλεσης αριθμ. πράξεων με ακέραιους αριθμούς:

```
class Arithmetic {
  public static void main(String args[]) {
    int x = 17, y = 5;
    System.out.println("x = " + x);
```

```
    System.out.println("y = " + y);
    System.out.println("x + y = " + (x + y));
    System.out.println("x - y = " + (x - y));
    System.out.println("x * y = " + (x * y));
    System.out.println("x / y = " + (x / y));
    System.out.println("x % y = " + (x % y));
  }
}
```

Αν εκτελέσουμε το πρόγραμμα θα πάρουμε τα αποτελέσματα:

```
x = 17
y = 5
x + y = 22
x - y = 12
x * y = 85
x / y = 3
x % y = 2
```

Ένα δεύτερο απλό παράδειγμα με πραγματικούς αριθμούς:

```
class FloatMath {
 public static void main(String args[]) {
    float x = 23.5f, y = 7.3f;
    System.out.println("x = " + x);
    System.out.println("y = " + y);
    System.out.println("x + y = " + (x + y));
    System.out.println("x - y = " + (x - y));
    System.out.println("x * y = " + (x * y));
    System.out.println("x / y = " + (x / y));
    System.out.println("x % y = " + (x % y));
  }
}
```

Αν εκτελέσουμε το πρόγραμμα θα πάρουμε τα αποτελέσματα:

```
x = 23.5
y = 7.3
x + y = 30.8
x - y = 16.2
x * y = 171.55
x / y = 3.219178
x % y = 1.5999994
```

ΑΝΑΛΥΣΗ ΤΗΣ ΕΚΤΕΛΕΣΗΣ ΑΡΙΘΜΗΤΙΚΩΝ ΠΡΑΞΕΩΝ

Πρόσθεση και αφαίρεση ακεραίων *(int)*

Στο παρακάτω παράδειγμα:

- Ορίζουμε τρεις ακέραιες μεταβλητές a, b, c. Δίνουμε τιμές στις a = 1 και b = 2
- Εκτελούμε πρόσθεση και αφαίρεση των αριθμών a και b και τοποθετούμε το αποτέλεσμα στη μεταβλητή c.
- Τέλος εμφανίζουμε τα αποτελέσματα.

```
class Example2 {
  public static void main(String args[]) {
    int a = 1;
    int b = 2;
    int c;
    System.out.println("a = " + a);
    System.out.println("b = " + b);
    c = a + b;
    System.out.println("a + b = " + c);
    c = a - b;
    System.out.println("a - b = " + c);
  }
}
```

Αν εκτελέσουμε το πρόγραμμα θα πάρουμε τα αποτελέσματα:

```
a = 1
b = 2
a + b = 3
a - b = -1
```

Πρόσθεση και αφαίρεση πραγματικών - Double

Στο παρακάτω παράδειγμα:

- Ορίζουμε τρεις μεταβλητές, που θα πάρουν πραγματικές τιμές, τις a, b, c. Δίνουμε τιμές στις a = 8.5 και b = 6.5
- Εκτελούμε πρόσθεση και αφαίρεση των αριθμών a και b και τοποθετούμε το αποτέλεσμα στη μεταβλητή c.
- Τέλος εμφανίζουμε τα αποτελέσματα.

```
class Example3 {
  public static void main(String args[]) {
    double a = 8.5;
    double b = 6.5;
    double c;
    System.out.println("a = " + a);
    System.out.println("b = " + b);
    c = a + b;
    System.out.println("a + b = " + c);
    c = a - b;
    System.out.println("a - b = " + c);
  }
}
```

Αν εκτελέσουμε το πρόγραμμα θα πάρουμε τα αποτελέσματα:

```
a = 8.5
b = 6.5
a + b = 15.0
a - b =  2.0
```

Πολλαπλασιασμός και Διαίρεση ακεραίων

Στο παρακάτω παράδειγμα:

- Ορίζουμε τρεις ακέραιες μεταβλητές a, b, c. Δίνουμε τιμές στις : a = 20 και b = 2.
- Εκτελούμε πολλαπλασιασμό και διαίρεση των αριθμών a και b και τοποθετούμε το αποτέλεσμα στη μεταβλητή c.
- Τέλος εμφανίζουμε τα αποτελέσματα.

```
class Example4 {
  public static void main(String args[]) {
    int a = 20;
    int b = 2;
    int c;
    System.out.println("a = " + a);
    System.out.println("b = " + b);
    c = a/b;
    System.out.println("a / b = " + c);
    c = a * b;
    System.out.println("a * b = " + c);
  }
}
```

Αν εκτελέσουμε το πρόγραμμα θα πάρουμε τα αποτελέσματα:

```
a = 20
b = 2
a / b = 10
a * b = 40
```

Προσοχή στη διαίρεση ακεραίων

Στη διαίρεση δύο ακεραίων, όπου υπάρχει υπόλοιπο (το αποτέλεσμα είναι ένας δεκαδικός αριθμός), θα πρέπει ο ένας από τους διαιρέτη ή διαιρετέο να είναι τύπος κινητής υποδιαστολής, το δε αποτέλεσμα θα πρέπει να αποθηκευτεί σε μεταβλητή τύπου κινητής υποδιαστολής. Παράδειγμα:

```
class TestDivision {
public static void main(String args[]) {
   float a = 10f;
   int b = 3;
   float c;
   System.out.println("a = " + a);
   System.out.println("b = " + b);
   c = a/b;
   System.out.println("a / b = " + c);
}
}
```

Αν εκτελέσουμε το πρόγραμμα θα πάρουμε:

```
a = 10.0
b = 3
a / b = 3.3333333
```

Ο Τελεστής - υπόλοιπο διαίρεσης (%)

Στο παρακάτω παράδειγμα:

- Ορίζουμε τρεις ακέραιες μεταβλητές a, b, c. Δίνουμε τιμές στις : a = 10 και b = 3.
- Εκτελούμε την διαίρεση των αριθμών a και b με τον τελεστή % και τοποθετούμε το αποτέλεσμα στη μεταβλητή c.
- Τέλος εμφανίζουμε τα αποτελέσματα.

```
class Example5 {
  public static void main(String args[]) {
   int a = 10;
   int b = 3;
   int c;
   System.out.println("a = " + a);
   System.out.println("b = " + b);
   c = a%b;
   System.out.println("a % b = " + c);
  }}
```

Αν εκτελέσουμε το πρόγραμμα θα πάρουμε το αποτέλεσμα:

```
a = 10
b = 3
a % b = 1
```

Πράξεις με μεικτούς τύπους

```
class Example6  {
   public static void main(String[] args) {
      //Μεταβλητές διάφορων τύπων
      int i = 37;
      int j = 42;
      double x = 27.475;
      double y = 7.22;

      //Πρόσθεση αριθμών
      System.out.println("Adding...");
      System.out.println("    i + j = " + (i + j));
      System.out.println("    x + y = " + (x + y));

      //Αφαίρεση αριθμών
      System.out.println("Subtracting...");
      System.out.println("    i - j = " + (i - j));
      System.out.println("    x - y = " + (x - y));

      //Πολλαπλασιασμός αριθμών
      System.out.println("Multiplying...");
      System.out.println("    i * j = " + (i * j));
      System.out.println("    x * y = " + (x * y));

      //Διαίρεση αριθμών
      System.out.println("Dividing...");
      System.out.println("    i / j = " + (i / j));
      System.out.println("    x / y = " + (x / y));
```

```
 //Υπολογισμός υπολοίπου διαίρεσης
       System.out.println("Computing the remainder...");
       System.out.println("    i % j = " + (i % j));
       System.out.println("    x % y = " + (x % y));

       //Μεικτοί τύποι
       System.out.println("Mixing types...");
       System.out.println("    j + y = " + (j + y));
       System.out.println("    i * x = " + (i * x));
   }
}
```

Αν εκτελέσουμε το πρόγραμμα θα πάρουμε τα αποτελέσματα:

```
Adding...
    i + j = 79
    x + y = 34.695
Subtracting...
    i - j = -5
    x - y = 20.255000000000003
Multiplying...
    i * j = 1554
    x * y = 198.36950000000002
Dividing...
    i / j = 0
    x / y = 3.805401662049862
Computing the remainder...
    i % j = 37
    x % y = 5.815000000000002
Mixing types...
    j + y = 49.22
    i * x = 1016.575
```

Στην παρακάτω άσκηση θα υπολογίσουμε την ενέργεια ενός Ηλεκτρονίου σύμφωνα με την σχέση του Αϊνστάιν E = mc2. Δίνονται οι τιμές της μάζας ηλεκτρονίου mass = 9.1096E-25 και της ταχύτητας c = 2.998E8 του.

```
class mc2 {
  public static void main (String args[]) {
    double mass = 9.1096E-25;
    double c = 2.998E8;
    double E = mass * c * c;
    System.out.println(E);
  }
}
```

Αν εκτελέσουμε το πρόγραμμα θα πάρουμε το αποτέλεσμα:

```
8.18771e-08
```

Μετατροπή τύπων δεδομένων *(Type Conversion and Casting)*

Η Java υποστηρίζει τρεις τύπους μετατροπών: με **εκχώρηση** *(assignment),* με **προαγωγή** *(promotion)* και με **διανομή** *(casting)*. Η μετατροπή με εκχώρηση γίνεται όταν η τιμή μιας μεταβλητής εκχωρείται σε μια μεταβλητή άλλου τύπου. Μόνο μετατροπές από μικρότερο τύπο σε μεγαλύτερο μπορεί να λάβει χώρα. Π.χ. **double ar = 987.65F;**

Η μετατροπή με προαγωγή εκτελείται αυτόματα όταν οι τελεστές μιας παράστασης μετατρέπουν τις τιμές των τελεστέων. Για παράδειγμα σε μία πρόσθεση ακεραίου και πραγματικού τύπου float, ο ακέραιος θα μετατραπεί σε αριθμό τύπου float πριν την εκτέλεση της πράξης. Π.χ.

```
int x = 15;
double y = 20.346;
System.out. println("x + y = "+(x+y));   // Το αποτέλεσμα θα είναι: x+y = 35.346
```

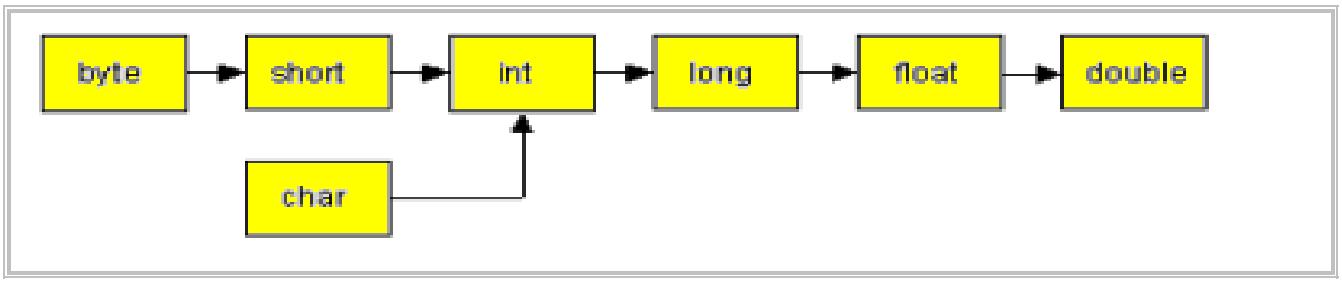
Οι επιτρεπτές μετατροπές με προαγωγή ακολουθούν τη διάταξη του Σχ. 1. Προσοχή στις μετατροπές τύπων γιατί μπορεί να έχουμε απώλειες σημαντικών ψηφίων. Π.χ. μια μετατροπή από long σε float μπορεί να οδηγήσει σε απώλεια ψηφίων.

Η μετατροπή με διανομή είναι η μετατροπή που εντέλει τη Java να προσπαθήσει να αποθηκεύσει μία τιμή σύμφωνα με ένα επιθυμητό τύπο. Η σύνταξη μιας διανομής – casting είναι:

<μεταβλητή> = <επιθυμητός τύπος> τιμή

Π.χ.
int i;
double d=987.654;
τότε **i = (int)d**; *(το αποτέλεσμα i=987, χάνονται τα δεκαδικά ψηφία)*

Σχ. 1. Οι επιτρεπτές μετατροπές βασικών τύπων της java



Το παρακάτω παράδειγμα δείχνει τέτοιες μετατροπές:

```
class Conversion {
  public static void main(String args[]) {

    byte b;
    int i = 257;
    double d = 323.142;
    System.out.println("\nConversion of int to byte.");
    b = (byte) i;
    System.out.println("i and b " + i + " " + b);
    System.out.println("\nConversion of double to int.");
    i = (int) d;
    System.out.println("d and i " + d + " " + i);
    System.out.println("\nConversion of double to byte.");
    b = (byte) d;
    System.out.println("d and b " + d + " " + b);
  }
}
```

Αν εκτελέσουμε το πρόγραμμα θα πάρουμε τα αποτελέσματα:

```
Conversion of int to byte.
i and b 257 1                (το 1 είναι το υπόλοιπο της διαίρεσης 257/256 – του byte)
Conversion of double to int.
d and i 323.142 323          (το ακέραιο μέρος, χωρίς τα δεκαδικά ψηφία)
Conversion of double to byte.
d and b 323.142 67           (323%256 = 67)
```

Τελεστές σύγκρισης

Οι τελεστές αυτοί συγκρίνουν δύο τιμές και καθορίζουν τη σχέση αλήθειας ή ψεύδους (true ή false) μεταξύ τους.

Τελεστής	Έλεγχος που εκτελεί
>	true αν το x1 είναι μεγαλύτερο του x2 (x1>x2)
>=	true αν το x1 είναι μεγαλύτερο ή ίσο του x2 (x1>=x2)
<	true αν το x1 είναι μικρότερο του x2 (x1<x2)
<=	true αν το x1 είναι μικρότερο ή ίσο του x2 (x1<=x2)
==	True αν το x1 είναι ίσο με το x2 (x1==x2)
!=	true αν το x1 είναι διάφορο του x2 (x1!=x2)

Λογικοί τελεστές

Είναι οι τελεστές που τους χρησιμοποιούμε για την εκτέλεση σύνθετων λογικών εκφράσεων.

Τελεστής	Έλεγχος που εκτελεί
&&	true αν και η x1 και η x2 είναι true (x1 && x2). Η x2 δεν αποτιμάται αν η x1 είναι false.
||	false αν και η x1 και η x2 είναι false (x1 || x2). Η x2 δεν αποτιμάται αν η x1 είναι true.
!	Η λογική άρνηση της έκφρασης x (!x).

&	Το ίδιο με τον && μόνο που η x2 θα αποτιμηθεί ακόμα και αν η x1 είναι false (x1&x2).
|	Το ίδιο με τον || μόνο που η x2 θα αποτιμηθεί ακόμα και αν η x1 είναι true (x1|x2).

Δυαδικοί τελεστές

Με τους δυαδικούς τελεστές εκτελούμε πράξεις σε επίπεδο bit.

Τελεστής	Περιγραφή
>>	Ολίσθηση x2 bits δεξιά στο x1 (x1 >> x2)
<<	Ολίσθηση x2 bits αριστερά στο x1 (x1<<x2)
>>>	Όπως ο >> αλλά χωρίς πρόσημο (x1 >>> x2)
&	Δυαδικό (ΚΑΙ) των x1 και x2 (x1&x2)
|	Δυαδικό (Ή) των x1 και x2 (x1 | x2)
^	Αποκλειστικό (Ή) (XOR) των x1 και x2 (x1 ^ x2)
~	Δυαδικό συμπλήρωμα του x (~x)