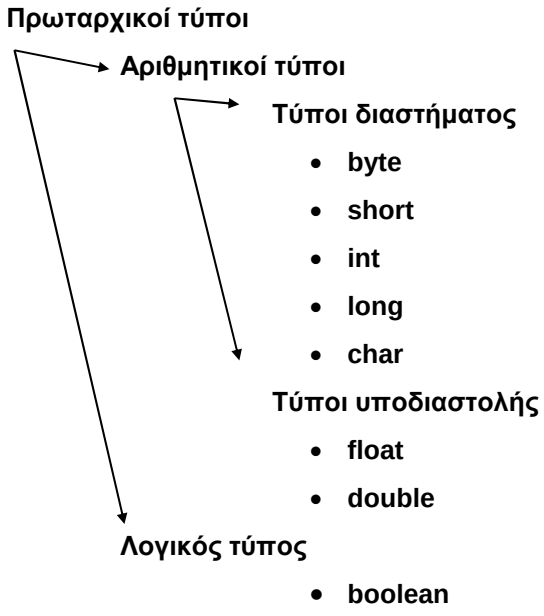
1 ΕΙΣΑΓΩΓΗ

Η γλώσσα προγραμματισμού Java είναι ισχυρά τυποποιημένη (strongly typed), που σημαίνει ότι κάθε μεταβλητή και κάθε έκφραση έχει κάποιο τύπο, ο οποίος πρέπει να είναι γνωστός κατά το χρόνο της μεταγλώττισης. Οι τύποι στη Java χωρίζονται σε δύο κατηγορίες:

- τους **πρωταρχικούς τύπους** (primitive types) και
- τους **τύπους αναφοράς** (reference types)

Πρωταρχικοί Τύποι

Οι πρωταρχικοί τύποι ονομάζονται και βασικοί και χωρίζονται στους αριθμητικούς τύπους (numeric types) και στον λογικό τύπο boolean. Οι αριθμητικοί τύποι χωρίζονται με τη σειρά τους στους τύπους περιοχής ή διαστήματος (integral types) και στους τύπους κινητής υποδιαστολής (floating-point types). Η ιεραρχία των βασικών τύπων φαίνεται παρακάτω:



Όλοι οι βασικοί τύποι δεδομένων παρέχονται αυτόματα από τον μεταγλωττιστή της Java και από πλευράς μεγέθους και εύρους τιμών είναι ίδιοι σε όλες της υλοποιήσεις της. Στον πίνακα που ακολουθεί φαίνονται το μέγεθος (σε bits) και το εύρος των τιμών του κάθε βασικού τύπου της Java.

Τύπος	Μέγεθος	Ελάχιστη τιμή	Μέγιστη τιμή	"wrapper" τύπος
char	16 bits	Unicode 0	Unicode 2^{16}-1	Character
byte	8 bits	-128	+127	Byte
short	16 bits	-2^{15}	$+2^{15}$-1	Short
int	32 bits	-2^{31}	$+2^{31}$-1	Integer
long	64 bits	-2^{63}	$+2^{63}$-1	Long
float	32 bits	1.402E-45 (*)	3.402E+38 (*)	Float
double	64 bits	4.94E-324 (*)	1.79E+308 (*)	Double
boolean				Boolean
void				Void

(*) τυποποίηση $IEEE_{754}$

Τύποι Αναφοράς

Οι τύποι αναφοράς που ορίζονται στη Java ανήκουν σε τρεις κατηγορίες:

- τους **τύπους κλάσης** (class types)
- τους **τύπους πίνακα** (array types) και
- τους **τύπους διασύνδεσης** (interface types)

Η κλάση αποτελεί το βασικό μηχανισμό παραγωγής νέων τύπων δεδομένων από το χρήστη. Ο τύπος πίνακα ορίζεται με τη βοήθεια ενός ειδικού μηχανισμού (βλέπε επόμενο κεφάλαιο) που ορίζεται από τη γλώσσα. Ο τύπος διασύνδεσης αποτελεί ένα μηχανισμό ορισμού αφηρημένων τύπων δεδομένων (abstract data types). Παραδείγματα τύπων διασύνδεσης δίνονται στο κεφάλαιο 3.

Στα πλαίσια της κλάσης που ορίζει έναν τύπο περιλαμβάνονται:

(α) ένα σύνολο δηλώσεων τύπων (βασικών ή/και αναφοράς). Οι τύποι αυτοί ονομάζονται μέλη δεδομένων (data members) της κλάσης και ορίζουν την εσωτερική αναπαράσταση του τύπου. Οι δηλώσεις αυτές πρέπει να είναι ιδιωτικές (private) γιατί η αναπαράσταση του τύπου δεν πρέπει να γίνεται γνωστή εκτός της κλάσης.

(β) ένα σύνολο από μεθόδους οι οποίες ορίζουν το σύνολο των πράξεων ή λειτουργιών του τύπου. Στις μεθόδους αυτές συμπεριλαμβάνονται και οι αντίστοιχοι δομητές (constructors) της κλάσης. Οι μέθοδοι που αντιστοιχούν στις πράξεις του τύπου πρέπει να είναι δημόσιες (public) γιατί αποτελούν την διασύνδεση χρήσης του τύπου.

Στο τμήμα κώδικα 1 ορίζεται ο τύπος δεδομένων παραλληλόγραμμο. Τα μέλη δεδομένων του τύπου στην περίπτωση αυτή είναι δύο ακέραιοι **height** και **width** που ορίζουν την εσωτερική αναπαράσταση του παραλληλόγραμμου.
Οι πράξεις που ορίζονται για το παραλληλόγραμμο είναι τέσσερις:

- υπολογισμός του ύψους (μέθοδος **getHeigth**)
- υπολογισμός του πλάτους (μέθοδος **getWidth**)
- υπολογισμός εμβαδού (μέθοδος **calculateArea**)
- έλεγχος εάν πρόκειται για μεγάλο παραλληλόγραμμο (μέθοδος **isBig**)

Η κλάση **Rectangle** λειτουργεί σαν μία γεννήτρια αντικειμένων που αποτελούν στιγμιότυπα του τύπου. Για τη δημιουργία των στιγμιότυπων αυτών φροντίζουν οι δύο δομητές της κλάσης. Ο ένας από αυτούς αναφέρεται στη δημιουργία ενός τυχαίου παραλληλόγραμμου και ο δεύτερος στην περίπτωση του τετραγώνου.

Αξίζει να παρατηρήσουμε ότι το ύψος και το πλάτος κάθε παραλληλόγραμμου δεν γίνεται γνωστό κατ' ευθείαν μέσω των μεταβλητών **heigth** και **width** του κάθε στιγμιότυπου, αλλά μόνο μέσω της επίκλησης των αντίστοιχων μεθόδων **getHeigth** και **getWidth** που αντιστοιχούν στις πράξεις του τύπου και πρέπει να σταλούν σαν μηνύματα προς τα αντικείμενα.

Η δημιουργία 4 διαφορετικών αντικειμένων τύπου **Rectangle** καθώς και η χρήση των αντίστοιχων πράξεων του τύπου φαίνονται στο τμήμα κώδικα 2

```
public class Rectangle
// Ορισμός του τύπου Rectangle
{
        private int height, width;

        public Rectangle (int h, int w)
        {       height = h;
                width = w;
        }
        public Rectangle (int s)
        {       height = width = s;
        }

        public int getHeigth( )
        {       return heigth;
        }

        public int getWidth( )
        {       return width;
        }

        public int calculateArea ( )
        {       return height * width;
        }

        private int perimeter ( )
        {       return height * 2 + width * 2;
        }

        public boolean isbig ( )
        {       return perimeter( ) > 100;
        }

}
```

Τμήμα Κώδικα 1

```
public static void main(String[ ] args) throws IOException
        {
            int x,y;
            Rectangle square1 = new Rectangle (4);
            Rectangle square2 = new Rectangle (8);
            x = 9; y = 9;
            Rectangle box1 = new Rectangle (x,y);
            Rectangle box2 = new Rectangle (3,5);
            System.out.println("AreaA="+square1.calculateArea( ));
            System.out.println("AreaB="+square2.calculateArea( ));
            System.out.println("AreaC="+box1.calculateArea( ));
            System.out.println("AreaD="+box2.calculateArea( ));
        }
}
```

Τμήμα Κώδικα 2

Μετάπτωση Τύπου (Type Casting)

Σε μερικές περιπτώσεις είναι αναγκαία η μετατροπή του τύπου μιας μεταβλητής σε κάποιον άλλο τύπο. Έτσι για παράδειγμα μπορεί η τιμή μιας μεταβλητής ακέραιου τύπου να χρειάζεται να καταχωρηθεί σε μία μεταβλητή πραγματικού τύπου, ή να περάσει σαν παράμετρος σε μία μέθοδο η οποία είναι ορισμένη μόνο για πραγματικές μεταβλητές.

Η μετατροπή του τύπου μιας μεταβλητής σε έναν άλλο ονομάζεται **μετάπτωση** (***casting***). Για να γίνει αυτό τοποθετούμε τον τύπο στον οποίο θέλουμε να μεταπέσει η μεταβλητή μέσα σε παρενθέσεις μπροστά από το όνομα της μεταβλητής ή το όνομα της μεθόδου. Για παράδειγμα η επόμενη πρόταση έχει σαν αποτέλεσμα την μετάπτωση της ακέραιας μεταβλητής που επιστρέφεται από τη μέθοδο **calculateArea** σε πραγματική και την ανάθεση της τιμής της στην πραγματική μεταβλητή **f**.

```
float f = (float) calculateArea( );
```

Η διαδικασία της μετάπτωσης δεν είναι πάντοτε ασφαλής καθώς είναι πιθανόν να χαθεί μέρος της πληροφορίας ή να αλλοιωθεί η ακρίβεια μιας τιμής. Παραδείγματος χάριν όταν έχουμε μετάπτωση μίας τιμής τύπου **double** σε μία τιμή τύπου **float** ο μεταγλωττιστής στρογγυλοποιεί αναγκαστικά την τιμή.

Για να υπάρχει ασφάλεια κατά τη διαδικασία της μετάπτωσης τύπων θα πρέπει ο τύπος μετάπτωσης να είναι κατ' ελάχιστον ίσος σε μέγεθος με τον αρχικό τύπο. Ο πίνακας που ακολουθεί δείχνει ασφαλείς περιπτώσεις μετάπτωσης πρωταρχικών τύπων σε άλλους

Αρχικός Τύπος	Τύπος Μετάπτωσης
byte	**short, char, int, long, float, double**
short	**int, long, float, double**
char	**int, long, float, double**
int	**long, float, double**
long	**float, double**
float	**double**

Είναι δυνατόν η διαδικασία της μετάπτωσης να ενεργοποιηθεί αυτόματα από τον μεταγλωττιστή χωρίς να έχει προβλεφθεί από τον κώδικα του προγράμματος. Στο επόμενο παράδειγμα η τιμή της μεταβλητής **ch** μεταπίπτει σε ακέραιο (τον αριθμό που κωδικοποιεί τον **'a'**) και στη συνέχεια υπολογίζεται η boolean τιμή της σύγκρισης στην **if**:

```
int x = 10;
char ch = 'a';
if (x > 'a') { . . . }
```

Σε περιπτώσεις σαν τις παραπάνω μιλάμε για διαδικασία **έμμεσης μετάπτωσης** (***implicit casting***). Η έμμεση μετάπτωση κατά κανόνα θα πρέπει να αποφεύγεται σαν επικίνδυνη.

Χώρος αποθήκευσης δεδομένων στη Java.

Η στοίβα (stack)

Ο σωρός (heap)

Στατική περιοχή μνήμης (static storage)

Περιοχή σταθερών (constant storage)

Καταχωρητές (registers)

Δευτερεύουσα μνήμη