Generics και ArrayLists

Generics

Μία από τις σημαντικότερες δυνατότητες των αντικειμενοστρεφών γλωσσών είναι η δυνατότητα ορισμού κλάσεων και μεθόδων που έχουν σαν παραμέτρους **τύπους δεδομένων,** δηλαδή να μπορούν να λειτουργούν με αντικείμενα διαφορετικών τύπων, παρέχοντας ταυτόχρονα ασφάλεια κατά την μεταγλώττιση (από την έκδοση 5.0 και μετά). Αυτή η δυνατότητα επιτρέπει την συλλογή αντικειμένων σε μία οντότητα – Java Collection.

Generic - κλάσεις

Μια **generic κλάση** ή **παραμετρική κλάση** ορίζεται όπως και οι μη generic κλάσεις με την διαφορά ότι μετά το όνομα της πρέπει να ακολουθεί τουλάχιστον μια *τυπική παράμετρος* ανάμεσα στα σύμβολα <>. Μπορεί να υπάρχουν περισσότερες τυπικοί παράμετροι διαχωρισμένες με το σύμβολο (,).

```
class name<T1, T2, ..., Tn>
{ /* ... */ }
```

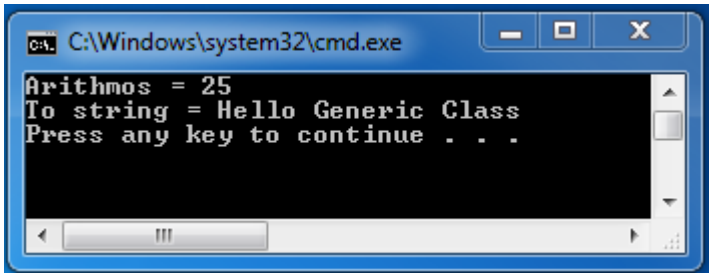
Παράδειγμα:

Θα δούμε τη χρήση μιας Generic κλάσης με δύο αντικείμενα διαφορετικών τύπων. Η κλάση έχει δύο μεθόδους την add() που αρχικοποιεί - δίνει τιμή στο αντικείμενο και την μέθοδο get() που επιστρέφει το αντικείμενο στο κυρίως πρόγραμμα. Προσέξτε επιπλέον τις εντολές δημιουργίας των δύο αντικειμένων του παραδείγματος.

```
class GenericTest1<T> {
  private T t;
  public void add(T t) {this.t = t;}
  public T get() {return t;}
```

```
public static void main(String[] args) {
   GenericTest1<Integer> i1 = new GenericTest1<Integer>();
   GenericTest1<String> s1 = new GenericTest1<String>();

   i1.add(new Integer(25));
   s1.add(new String("Hello Generic Class"));

   System.out.println("Arithmos = " + i1.get());
   System.out.println("To string = " + s1.get());
} }
```

Το αποτέλεσμα:



Generic - μέθοδοι

Γράφουμε μια generic μέθοδο που θα κληθεί με παραμέτρους διαφορετικών τύπων. Ανάλογα με τον τύπο της παραμέτρου ο μεταγλωττιστής χειρίζεται τις κλήσεις. Στην υπογραφή της μεθόδου θα υπάρχει, μέσα στα σύμβολα <>, ο τύπος του επιστρεφόμενου αποτελέσματος (στο παρακάτω παράδειγμα - <A>, δικό μας όνομα). Η μέθοδος μπορεί να δεχτεί μία ή περισσότερες παραμέτρους (χωρισμένοι με το σύμβολο (,) κόμμα).

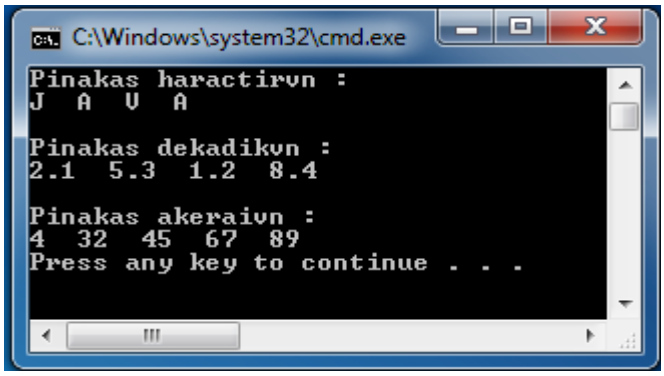
Προσοχή οι τυπικές παράμετροι μπορεί να είναι μόνο αναφορές και όχι βασικοί τύποι.

Στο παρακάτω παράδειγμα θα δούμε πώς μπορούμε να εμφανίσουμε τρείς - διαφορετικού τύπου - πίνακες με την ίδια μέθοδο.

```
class GenericMethod {
   public static < A > void printArray( A[] inputArray ) //ypografi tis methodou
  {
      for ( A element : inputArray )
```

```
        {
          System.out.print(element + "  ");
        }
        System.out.println();
    }

    public static void main( String args[] ) {
        Character[] charArray = { 'J', 'A', 'V', 'A'};
        Double[] doubleArray = { 2.1, 5.3, 1.2, 8.4 };
        Integer[] intArray = { 4, 32, 45, 67, 89 };

        System.out.println( "Pinakas haractirvn : " );
        printArray( charArray );
        System.out.println();
        System.out.println( "Pinakas dekadikvn : " );
        printArray( doubleArray );
        System.out.println();
        System.out.println( "Pinakas akeraivn : " );
        printArray( intArray  );
    } }
```

Το αποτέλεσμα:

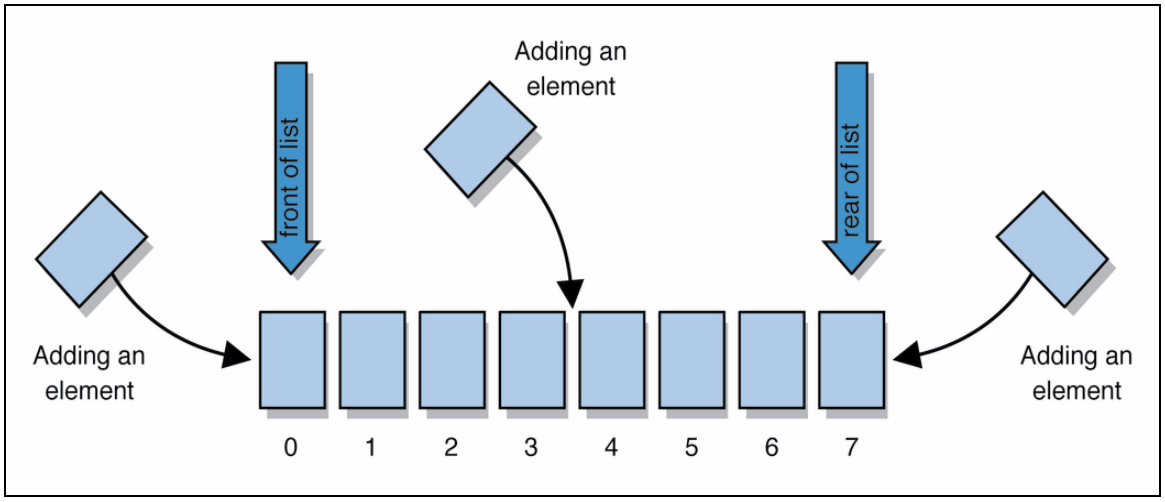


Λίστες - Lists

- **Λίστα :** μια συλλογή (collection) από διατεταγμένα στοιχεία

- Κάθε στοιχείο ανιχνεύεται από ένα δείκτη (0 έως n-1)
- Η λίστα έχει δυναμικά αυξανόμενο μέγεθος (size) – το πλήθος των στοιχείων της
- Τα στοιχεία μπορούν να εισαχθούν/διαγραφούν σε/από οποιαδήποτε θέση (αρχή, μέση, τέλος, κλπ.).

Μια σημαντική λίστα είναι η συλλογή ArrayList.



ArrayList

Μια σημαντική συλλογή (collection) αντικειμένων, που ακολουθεί την generics – φιλοσοφία, είναι ο δυναμικός πίνακας ή δυναμική λίστα - ArrayList. Είναι ένας δυναμικός χώρος μνήμης που μπορεί να επεκταθεί κατά την εκτέλεση του προγράμματος, σε αντίθεση με τον πίνακα σταθερού μήκους (array προκαθορισμένων κελιών) που δεν μπορεί να επεκταθεί.

- Η ArrayList μπορεί να οριστεί με αρχικό μέγεθος, που μπορεί να αυξάνει (προσθήκη στοιχείων) ή να μειώνεται ανάλογα (διαγραφή στοιχείων). Για να χρησιμοποιήσουμε την κλάση ArrayList θα πρέπει να εισάγουμε το πακέτο java.util (**import java.util.***).
- Με τον ορισμό μιας λίστας πρέπει να καθορίζεται ο **τύπος των στοιχείων** της λίστας μέσα στις τριγωνικές αγκύλες <>:

```
ArrayList<Type> name = new ArrayList<Type>();
```

π.χ ArrayList<String> list = new ArrayList<String>(20);

- Ο τύπος των στοιχείων πρέπει να είναι τύπος – object και όχι βασικός δηλ., δεν μπορεί να υπάρξει ένας τέτοιος ορισμός:

ArrayList<int> list = new ArrayList<int>(); *// Λάθος ορισμός*

Όμως μπορούμε να έχουμε βασικούς τύπους χρησιμοποιώντας αντικείμενα των κλάσεων των βασικών τύπων *(wrapper classes)*, δηλ.:

```
ArrayList<Integer> list = new ArrayList<Integer>();
```

Wrapper classes

Τα αντικείμενα αυτών των κλάσεων κρατούν τιμές των βασικών τύπων.

Βασικός Τύπος	Wrapper Τύπος
int	**Integer**
double	**Double**
char	**Character**
boolean	**Boolean**

Μετά τον ορισμό της λίστας με τύπο – wrapper, μπορούμε να χειριστούμε τις βασικές τιμές σε μεθόδους και κώδικα:

```
ArrayList<Double> vathmoi = new ArrayList<Double>();
vathmoi.add(3.2);
vathmoi.add(2.7);
...
```

- Μια μέθοδος μπορεί να δεχτεί σαν παράμετρο μια ArrayList ή να επιστρέψει μια λίστα (return):

```
public static void name(ArrayList<Type> name)
```

Παράδειγμα

Μέθοδος που διαγράφει τους ζυγούς αριθμούς από την λίστα list.

```
public static void DiagrafiZygonArithmon(ArrayList<Integer> list) {
    for (int i = list.size() - 1; i >= 0; i--) {
        int n = list.get(i);
```

```
        if (n % 2 == 0) {
            list.remove(i); } }
}
```

Τρείς διαφορετικοί δομητές:

ArrayList() - μια κενή λίστα.

ArrayList(Collection c) – η λίστα αρχικοποιείται με τα αντικείμενα της συλλογής c.

ArrayList(int capacity) – η λίστα θα έχει αρχικό μέγεθος – capacity.

Σημαντικές Μέθοδοι της ArrayList

boolean add(Object o)	Προσθέτει το αντικείμενο - ο στο τέλος της λίστας
void add(int index, Object o)	Εισάγει το αντικείμενο-ο στην θέση index
void clear()	Διαγράφει όλα τα στοιχεία της λίστας
int indexOf(Object o)	Επιστρέφει την θέση της πρώτης θέσης εύρεσης του στοιχείου (-1 αν δεν το βρει)
Object get(int index)	Επιστρέφει το στοιχείο της συγκεκριμένης θέσης
Object remove(int index)	Διαγράφει το στοιχείο της συγκεκριμένης θέσης
Object set(int index, Object element)	Αντικαθιστά το στοιχείο στην συγκεκριμένη θέση με το συγκεκριμένο στοιχείο
int size()	Επιστρέφει το πλήθος των στοιχείων της λίστας
String toString()	Επιστρέφει ένα String που αντιπροσωπεύει την λίστα με την μορφή "[3, 42, -7, 15]"
boolean addAll(Collection c) **boolean addAll**(index, Collection c)	Προσθέτει όλα τα στοιχεία της συλλογής c στο τέλος της λίστας ή τα εισάγει στην συγκεκριμένη θέση
boolean contains(Object o)	Επιστρέφει true αν η λίστα περιέχει το στοιχείο
boolean containsAll(Collection c)	Επιστρέφει true αν η λίστα περιέχει όλα τα στοιχεία της συλλογής c
boolean equals(Collection c)	Επιστρέφει true αν η συλλογή c περιέχει τα στοιχεία της λίστας

int lastIndexOf(Object o)	Επιστρέφει την τελευταία θέση του στοιχείου στη λίστα (-1 αν δεν το βρει)
Object remove(int index)	Διαγράφει το στοιχείο στην συγκεκριμένη θέση από την λίστα
protected void removeRange(int fromIndex, int toIndex)	Διαγράφει τα στοιχεία της λίστας που βρίσκονται στα όρια fromIndex έως toIndex
Object[] toArray()	Επιστρέφει τα στοιχεία της λίστας σαν πίνακα

.....Περισσότερα στην τεκμηρίωση

Προσοχή!!

Μια generic-ArrayList δέχεται αντικείμενα του τύπου που ορίζει, διαφορετικά λαμβάνουμε λάθος κατά την μεταγλώττιση. Π.χ.:

```
ArrayList<String> stringList = new ArrayList<String>(); //generic ArrayList μόνο για Strings
stringList.add("Skiathos"); //δεν υπάρχει λάθος γιατί το αντικείμενο είναι τύπουString
stringList.add(new Integer(2)); //λάθος στην μεταγλώττιση (compilation error)
```

Διαφορές με τα arrays

1) Στην **κατασκευή**:

```
String[] names = new String[3];
ArrayList<String> list = new ArrayList<String>();
```

2) Στην **αποθήκευση τιμών**:

```
names[0] = "Sakis";
list.add("Sakis");
```

3) Στην **Ανάκτηση τιμών**:

```
String s = names[0];
```

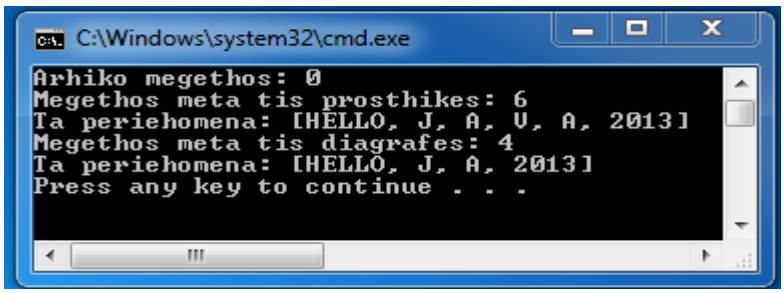
```
String s = list.get(0);
```

ΑΣΚΗΣΕΙΣ

Στην παρακάτω άσκηση δοκιμάζουμε τις μεθόδους size(), add() και remove(). Επειδή δεν χρησιμοποιούμε Τύπο (generics) μας βγάζει μήνυμα κατά την μεταγλώττιση (μη ασφαλής μεταγλώττιση).

```
import java.util.*;
class ArrayList1 {
  public static void main(String args[]) {
    ArrayList al = new ArrayList();
    System.out.println("Arhiko megethos: " + al.size());
    // prosthiki stoiheivn
    al.add("J");
    al.add("A");
    al.add("V");
    al.add("A");
    al.add(0, "HELLO");
    al.add(new Integer(2013));
    System.out.println("Megethos meta tis prosthikes: " + al.size());
    // emfanisi tou array list
    System.out.println("Ta periehomena: " + al);
    // diagrafi stoiheivn tou array list
    al.remove("V");
    al.remove(2);
    System.out.println("Megethos meta tis diagrafes: " + al.size());
    System.out.println("Ta periehomena: " + al);
  }
}
```

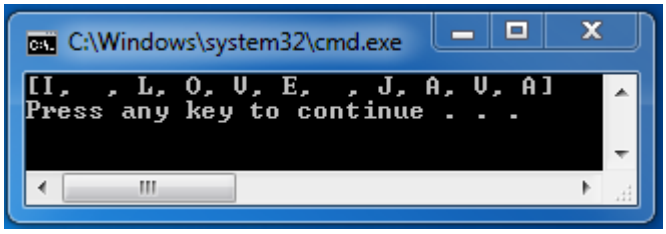
Το αποτέλεσμα:



Μια πιο ασφαλής παραλλαγή (για ασφαλή μεταγλώττιση) είναι να χρησιμοποιήσουμε generics με την χρήση τύπου στην δημιουργία του ArrayList. Έτσι δεν θα λάβουμε προειδοποιητικό μήνυμα κατά την μεταγλώττιση.

```
import java.util.*;
class arraylist2
{
 public static void main(String args[])
 {
  ArrayList<String> arr = new ArrayList<String>(10);
 arr.add("I");
 arr.add(" ");
 arr.add("L");
 arr.add("O");
 arr.add("V");
 arr.add("E");
 arr.add(" ");
 arr.add("J");
 arr.add("A");
 arr.add("V");
 arr.add("A");
 System.out.println(arr); } }
```

Το αποτέλεσμα:



Προσπέλαση στα στοιχεία της ArrayList με τον Iterator και την While

Ένας άλλος τρόπος προσπέλασης των στοιχείων ενός ArrayList (εκτός από την εντολή - for) είναι με την χρήση της **Iterator** ή **ListIterator** σε συνδιασμό με την εντολή while. Έχουν δύο μεθόδους που χρησιμοποιούμε με την while για να προσπελάσουμε τα στοιχεία: την μέθοδο **hasNext()**, που επιστρέφει true όσο υπάρχει επόμενο στοιχείο στην λίστα και η μέθοδος **next()** που επιστρέφει το επόμενο στοιχείο.

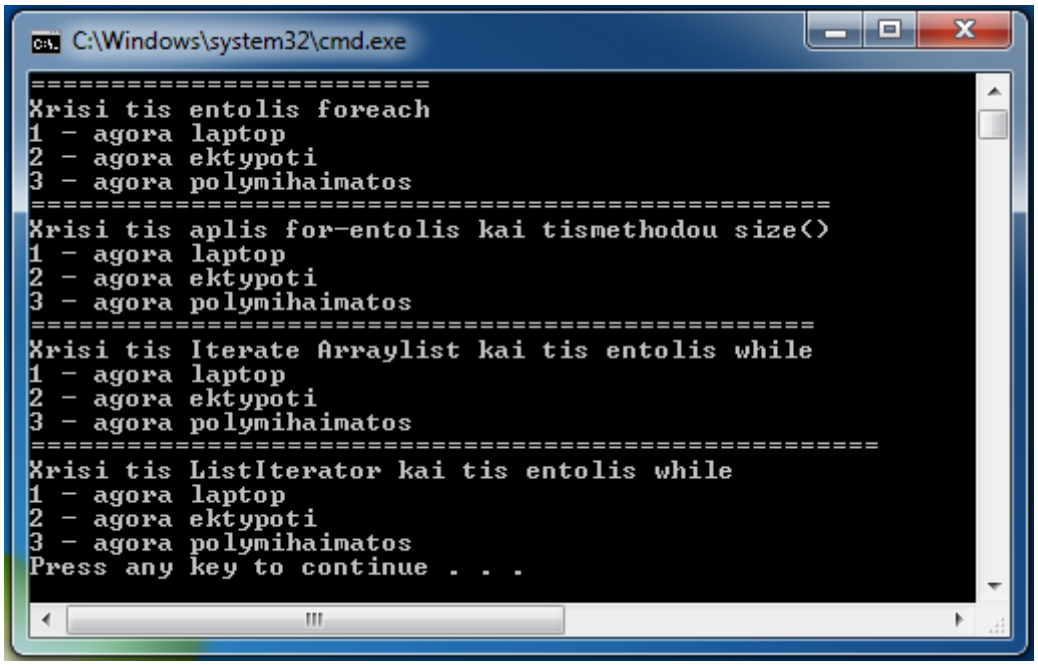
Παράδειγμα:

Στο παρακάτω παράδειγμα θα χρησιμοποιήσουμε την εντολή for και την listIterator για να προσπελάσουμε τα στοιχεία μιας λίστας.

```
import java.util.*;
class IterarorArrayList {
   public static void main(String args[]){

      //mia lista me douleies pou prepei na kanv (todo list)
       ArrayList<String> loopList = new ArrayList<String>();

      //Stoiheia tis listas
```

```
        loopList.add("1 - agora laptop");
        loopList.add("2 - agora ektypoti  ");
        loopList.add("3 - agora polymihaimatos");

        //Xrisi tis entolis foreach
        System.out.println("============================");
        System.out.println("Xrisi tis entolis foreach");
        for(String element: loopList){
            System.out.println(element);
        }



        //Xrisi tis aplis for-entolis kai tismethodou size

System.out.println("=======================================");
        System.out.println("Xrisi tis aplis for-entolis kai tismethodou size()");
        for(int i=0; i<loopList.size(); i++){
            System.out.println(loopList.get(i));
        }

        //Xrisi tis Iterator kai tis entolis while

System.out.println("=======================================");
        System.out.println("Xrisi tis Iterate Arraylist kai tis entolis while");
        Iterator<String> iterator = loopList.iterator();
        while(iterator.hasNext()){
            System.out.println(iterator.next());
        }

        //Xrisi tis ListIterator kai tis entolis while

System.out.println("=======================================");
        System.out.println("Xrisi tis ListIterator kai tis entolis while");
        ListIterator<String> listIterator = loopList.listIterator();
        while(listIterator.hasNext()){
            System.out.println(listIterator.next());
        }
    }
}
```

Το αποτέλεσμα:


C:\Windows\system32\cmd.exe
==========
Xrisi tis entolis foreach
1 - agora laptop
2 - agora ektypoti
3 - agora polymihaimatos
==========
Xrisi tis aplis for-entolis kai tismethodou size()
1 - agora laptop
2 - agora ektypoti
3 - agora polymihaimatos
==========
Xrisi tis Iterate Arraylist kai tis entolis while
1 - agora laptop
2 - agora ektypoti
3 - agora polymihaimatos
==========
Xrisi tis ListIterator kai tis entolis while
1 - agora laptop
2 - agora ektypoti
3 - agora polymihaimatos
Press any key to continue . . .