Collections – Lists – ArrayLists

Παναγιώτης Σφέτσος, PhD
http://aetos.it.teithe.gr/~sfetsos/
sfetsos@it.teithe.gr

Δομές Δεδομένων- Container Object

Δομή δεδομένων: μια δομημένη συλλογή δεδομένων, που παρέχει αποτελεσματικές λειτουργίες πρόσβασης και χειρισμού των δεδομένων.

- Στην αντικειμενοστρέφεια (*OO)* μια δομή δεδομένων είναι η ***container*** ή ***container object,*** δηλαδή ένα αντικείμενο που αποθηκεύει άλλα αντικείμενα που ονομάζονται στοιχεία *(elements).*

- Η Java παρέχει πολλές αποτελεσματικές δομές δεδομένων μέσα από μια Αρχιτεκτονική που ονομάζεται: **Java Collections Framework *(JCF).***

- Το *Java Collections Framework* υποστηρίζει δύο τύπους containers:

  (1) ***collection*****,** για αποθήκευση μιας **συλλογής στοιχείων**

  (2) ***map*****,** για αποθήκευση ζευγών **κλειδί/τιμή** *(key/value)*

Collections (Συλλογές)

Collection *(συλλογή):* είναι ένας αποθηκευτικός χώρος *(container)* για αποθήκευση αντικειμένων που καλούνται στοιχεία.

Λόγοι ύπαρξης: **Ο δυναμικός και αποτελεσματικότερος χειρισμός των αντικειμένων**

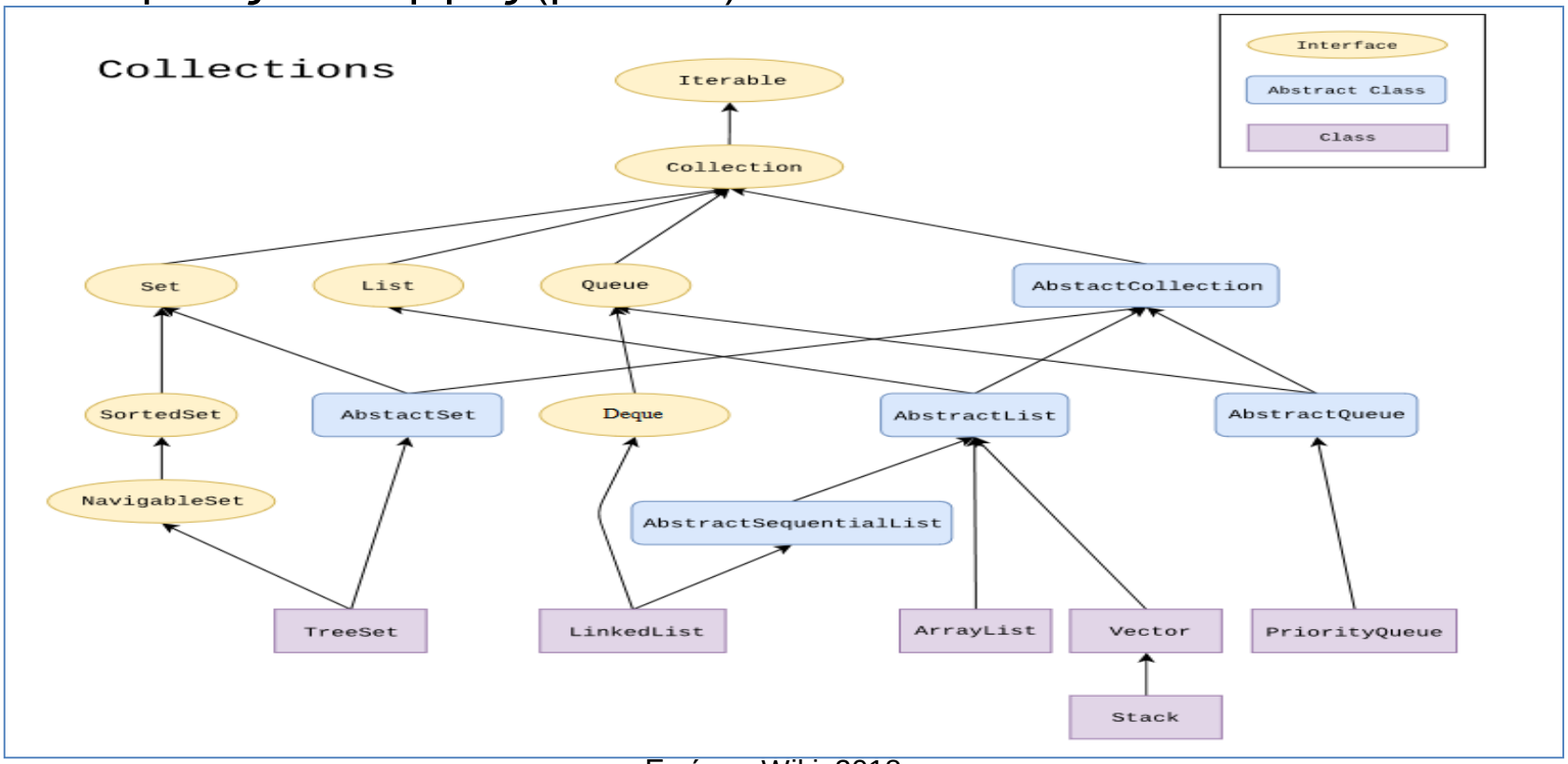
- Οι πίνακες αντικειμένων είναι στατικές δομές για αποθήκευση αντικειμένων (δεν αλλάζουν μέγεθος, κλπ.)
- Υπάρχουν περιπτώσεις όπου δεν είναι γνωστό το μέγεθος της δομής δεδομένων: μεγάλοι πίνακες (σπατάλη μνήμης, κλπ.)

Τέτοιες συλλογές είναι οι:

Λίστες (Lists)*, *Σύνολα (Sets)*, *Ουρές (Queues)*, *Απεικόνισεις (Maps)

Collections *(2 /14)*

- Οι collections *(συλλογές)* είναι interfaces *(διεπαφές).*
- Η καθεμία διεπαφή έχει μια ή περισσότερες υλοποιήσεις.
- Όταν υλοποιείται μια διεπαφή υπάρχει η δέσμευση να υλοποιούνται οι απαιτούμενες λειτουργίες (μεθόδοι).



Εικόνα - Wiki, 2018

Collections *(3 /14)*

- **Sets:** αποθηκεύουν ομάδα από μη επαναλαμβανόμενα στοιχεία.
- **Lists:** αποθηκεύουν μία ταξινομημένη συλλογή από στοιχεία.
- **Stacks:** αποθηκεύουν αντικείμενα που τα επεξεργαζόμαστε με την τεχνική last-in/first-out.
- **Queues:** αποθηκεύουν αντικείμενα που τα επεξεργαζόμαστε με την τεχνική first-in/first-out.
- **PriorityQueues:** αποθηκεύουν αντικείμενα που τα επεξεργαζόμαστε σύμφωνα με τις προτεραιότητές τους.

Collections *(4 /14)*

- Όλες οι διεπαφές και οι κλάσεις που ορίζονται στο πλαίσιο – JCF βρίσκονται στο πακέτο **java.util**
- Το πλαίσιο-JCF αποτελεί ένα άριστο παράδειγμα της χρήσης διεπαφών, αφηρημένων και απλών κλάσεων.
- Οι ***διεπαφές*** ορίζουν το πλαίσιο υλοποίησης
- Οι ***αφηρημένες*** κλάσεις παρέχουν μερική υλοποίηση των διεπαφών που βοηθούν τον χρήστη να γράψει αποτελεσματικό κώδικα.
- Οι ***απλές κλάσεις*** υλοποιούν τις διεπαφές με συγκεκριμένες δομές. Ο χρήστης ορίζει μια απλή κλάση που **κληρονομεί** την αφηρημένη κλάση αντί να υλοποιεί όλες τις μεθόδους της διεπαφής.
- Τέτοιες αφηρημένες κλάσεις, όπως η **AbstractCollection**, παρέχονται για ευκολία του χρήστη καθώς υλοποιούν όλες μεν τις μεθόδους της διεπαφής εκτός από τις **add, size,** και **iterator** που υλοποιούνται σε **συγκεκριμένες υποκλάσεις**.

Collections *(5 /14)*

Δημιουργία μιας Collection

Όλες οι υλοποιήσεις θα έχουν δύο δομητές:

- ένα δομητή χωρίς παραμέτρους για ένα άδειο collection
- ένα δομητή με παράμετρο ένα άλλο collection

- **Προσοχή: μόνο για τα collections, γιατί στις δικές μας συλλογές δεν μπορούμε να έχουμε διεπαφές με δομητές.**

- Η διεπαφή collection παρέχει τις βασικές λειτουργίες όπως της προσθήκης και διαγραφής στοιχείων, κλπ.

Collections *(6 /14)*

Collection: Βασικές Λειτουργίες

```
boolean add(Object element);
boolean remove(Object element);
boolean isEmpty( );
boolean contains(Object element);
int size( );
Iterator iterator( );
```

Collections *(7/14)*

Collection: 'Μαζικές λειτουργίες' *(bulk)* *(1/2)*

boolean containsAll(Collection c);

Επιστρέφει true εάν η συλλογή περιέχει όλα τα στοιχεία της συλλογής c που δίδεται σαν παράμετρος.

boolean addAll(Collection c);

Εισάγει όλα τα στοιχεία της συλλογής c που δίδεται σαν παράμετρος στην τρέχουσα συλλογή.

boolean removeAll(Collection c);

Διαγράφει από την συλλογή τα στοιχεία τα οποία εμπεριέχονται στην συλλογή c που δίδεται σαν παράμετρος

Collections *(8/14)*

Collection: 'Μαζικές λειτουργίες' *(bulk)* *(2/2)*

boolean retainAll(Collection c);

Διατηρεί στη συλλογή τα στοιχεία τα οποία εμπεριέχονται στην συλλογή c που δίδεται σαν παράμετρος και διαγράφει τα υπόλοιπα.

void clear();

Διαγράφει όλα τα στοιχεία από τη συλλογή.

Collection: Λειτουργίες διανυσμάτων

Η διεπαφή collection παρέχει την μέθοδο **toArray()** που επιστρέφει ένα αντίστοιχο πίνακα για την τρέχουσα συλλογή.

Π.χ. **Object[ ] pin = c.toArray();** *(μετατροπή σε πίνακα)*

Collections *(9/14)*

Collection: Αλγόριθμοι *(1/3)*

Στην κλάση Collections περιέχονται διάφοροι γενικής χρήσης ***αλγόριθμοι-λειτουργίες*** υπό την μορφή στατικών μεθόδων:

- **static int binarySearch(List list, Object key)**
  αναζητά το συγκεκριμένο στοιχείο στην συγκεκριμένη λίστα *(binary search algorithm)*.
- **static void copy(List dest, List src)**
  αντιγράφει όλα τα στοιχεία μιας λίστας σε μια άλλη.
- **static void fill(List list, Object o)**
  αντικαθιστά όλα τα στοιχεία της λίστας με το συγκεκριμένο στοιχείο.

Collections *(10/14)*

Collection: Αλγόριθμοι *(2/3)*

- **static Object max(Collection coll)**
  επιστρέφει το μεγαλύτερο στοιχείο της λίστας, σύμφωνα με την φυσική ταξινόμηση των στοιχείων.

- **static Object min(Collection coll)**
  επιστρέφει το μικρότερο στοιχείο της λίστας, σύμφωνα με την φυσική ταξινόμηση των στοιχείων.

- **static void reverse(List l)**
  αντιστρέφει τη διάταξη των στοιχείων της λίστας.

Collections *(11/14)*

Collection: Αλγόριθμοι *(3/3)*

- **static void shuffle(List list)**
  τυχαία μετάθεση των στοιχείων της λίστας *(default source of randomness)*.

- **static void sort(List list)**
  ταξινομεί την λίστα κατά αύξουσα τάξη.

- **static Set singleton(Object o)**
  επιστρέφει ένα αμετάβλητο set που περιέχει μόνο το συγκεκριμένο στοιχείο.

Collections *(12/14)*

Collection: Iterator *(Διαπροσπελαστής)* *(1/2)*

```
public interface Iterator {
    boolean hasNext( );
        // true - αν υπάρχει άλλο στοιχείο στη συλλογή

    Object next( );
        // επιστρέφει το επόμενο στοιχείο της συλλογής

     void remove( );
    // διαγράφει το στοιχείο που επιστρέφει η  next
}
```

Collections *(13/14)*

Collection: Iterator - *Χρήση* *(2/2)*

```
static void printAll (Collection coll) {
    Iterator it = coll.iterator( );
    while (it.hasNext( )) {
       System.out.println(it.next( ) );
    }
}
```

Πολυμορφική χρήση: Δουλεύει για οποιαδήποτε συλλογή (θα την δούμε αναλυτικά στην συλλογή ArrayList)

Collections *(14/14)*

Υλοποιήσεις συλλογών δεδομένων Java

- **List:**
  - **ArrayList,** σταθερός χρόνος προσπέλασης
  - **LinkedList,** γρήγορη εισαγωγή στοιχείων
- **Set:**
  - **HashSet,** γρήγορη δομή
  - **TreeSet,** ταξινομημένη δομή
- **Map:**
  - **HashMap**, γρήγορη δομή
  - **TreeMap,** ταξινομημένη δομή

Collections *Παράδειγμα (1/6)*

Στο παρακάτω παράδειγμα θα υλοποιήσουμε τις περισσότερες από τις συλλογές. Χρήση της **Iterator** για την εμφάνιση των στοιχείων.

```
import java.util.*;
public class Main {
   public static void main(String[] args) {
      List lnkLst = new LinkedList();
      lnkLst.add("LinkedList1");
      lnkLst.add("LinkedList2");
      lnkLst.add("LinkedList3");
      displayAll(lnkLst);
     System.out.println("_____");
```

Collections *Παράδειγμα (2/6)*

```
List aryLst = new ArrayList();
aryLst.add("ArrayList1");
 aryLst.add("ArrayList1");
aryLst.add("ArrayList2");
aryLst.add("ArrayList3");
aryLst.add("ArrayList4");
displayAll(aryLst);
System.out.println("_____");

Set hashSet = new HashSet();
hashSet.add("HashSet1");
hashSet.add("HashSet2");
displayAll(hashSet);
System.out.println("_____");
```

Collections *Παράδειγμα (3/6)*

```
SortedSet treeSet = new TreeSet();
 treeSet.add("TreeSet1");
 treeSet.add("TreeSet2");
 treeSet.add("TreeSet3");
 treeSet.add("TreeSet4");
 displayAll(treeSet);
 System.out.println("_____");

 LinkedHashSet lnkHashset = new LinkedHashSet();
 lnkHashset.add("LinkedHashSet1");
 lnkHashset.add("LinkedHashSet2");
 lnkHashset.add("LinkedHashSet3");
 displayAll(lnkHashset);
 System.out.println("_____");
```


System.out.println(" ");

Collections *Παράδειγμα (4/6)*

```
Map map1 = new HashMap();
map1.put("HashMap1", "HM1");
map1.put("HashMap2", "HM2");
map1.put("HashMap3", "HM3");
displayAll(map1.keySet());
displayAll(map1.values());
System.out.println("_____");

SortedMap map2 = new TreeMap();
map2.put("TreeMap1", "TM1");
map2.put("TreeMap2", "TM2");
map2.put("TreeMap3", "TM3");
displayAll(map2.keySet());
displayAll(map2.values());
System.out.println("_____");
```

Collections *Παράδειγμα (5/6)*

```
LinkedHashMap map3 = new LinkedHashMap();
map3.put("Panos", "1");
map3.put("Roulis", "2");
map3.put("Sakis", "3");
map3.put("Takis", "4");
displayAll(map3.keySet());
displayAll(map3.values()); }

static void displayAll(Collection col) {
Iterator itr = col.iterator();
while (itr.hasNext()) {
   String str = (String) itr.next();
   System.out.print(str + " ");
}
System.out.println(); } }
```

Collections *Παράδειγμα (6/6)*

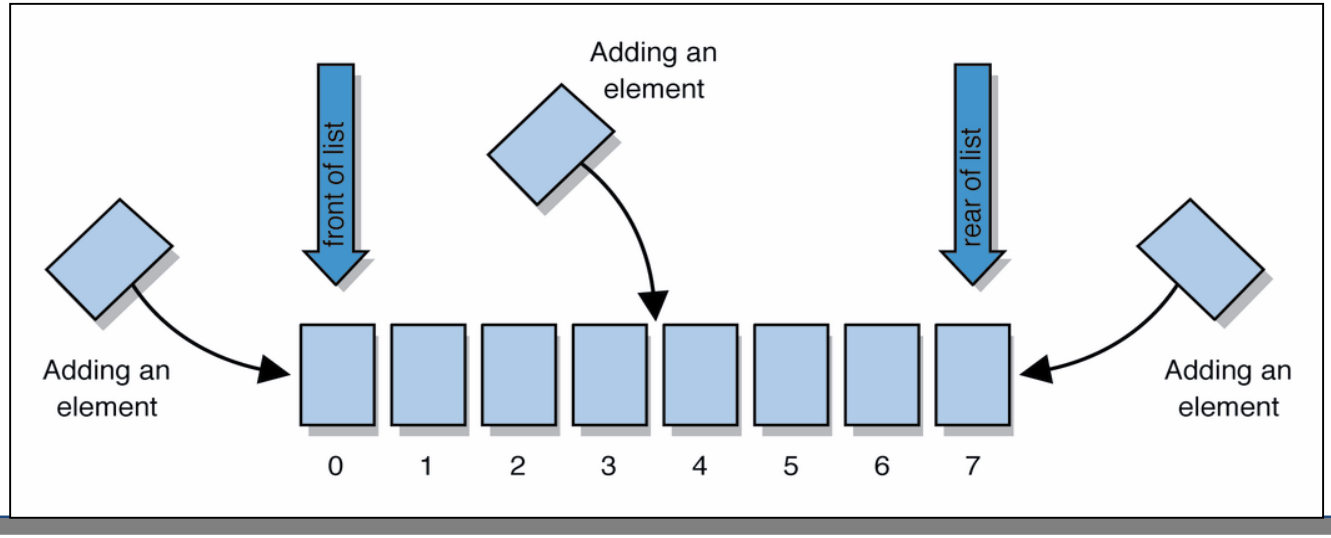
Το αποτέλεσμα:

```
run:
LinkedList1 LinkedList2 LinkedList3

_____________________________________________________
ArrayList1 ArrayList2 ArrayList3 ArrayList4

_____________________________________________________
HashSet2 HashSet1

_____________________________________________________
TreeSet1 TreeSet2 TreeSet3 TreeSet4

_____________________________________________________
LinkedHashSet1 LinkedHashSet2 LinkedHashSet3

_____________________________________________________
HashMap2 HashMap3 HashMap1
HM2 HM3 HM1

_____________________________________________________
TreeMap1 TreeMap2 TreeMap3
TM1 TM2 TM3

_____________________________________________________
Panos Roulis Sakis Takis
1 2 3 4
BUILD SUCCESSFUL (total time: 0 seconds)
```

Η διεπαφή List *(1/6)*

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 το αντικείμενο **ArrayList**



Η διεπαφή List *(2/6)*

- Η List είναι **ταξινομημένη** και μπορεί να έχει **διπλότυπα** στοιχεία
- Οι λειτουργίες είναι οι ίδιες με τις συλλογές.

```
int size( );
boolean isEmpty( );
boolean contains(Object e);
boolean add(Object e);
boolean remove(Object e);
Iterator iterator( );
```

```
boolean containsAll(Collection c);
boolean addAll(Collection c);
boolean removeAll(Collection c);
boolean retainAll(Collection c);
void clear( );
```

```
Object[ ] toArray( );
Object[ ] toArray(Object a[ ]);
```

Η διεπαφή List *(3/6)*

- Η List προσφέρει δύο υλοποιήσεις:
  - την **ArrayList,** που παρέχει ταχύτερες εισόδους και διαγραφές στοιχείων
  - την **LinkedList**, που παρέχει ταχύτερη τυχαία πρόσβαση

- Η List είναι διεπαφή δεν μπορούμε να πούμε: new List()

Καλός ορίσμός υλοποίησης:

```
List list = new ArrayList();
```

Κακός ορισμός - στυλ:

```
ArrayList list = new ArrayList();
```

Η διεπαφή List *(4/6)*

Μέθοδοι που κληρονομεί η List :

- **list.remove(e),** διαγράφει το πρώτο στοιχείο e
- **add** και **addAll**, προσθέτει στο τέλος της list
- Προσθήκη μιας list σε άλλη: **list1.addAll(list2);**
- Προσθήκη δύο-lists σε μια νέα :

```
List list3 = new ArrayList(list1);
list3.addAll(list2);
```

Η διεπαφή List *(5/6)*

```
public interface List extends Collection {
    //Positional access
    Object get(int index);
    Object set(int index, Object element);
    void add(int index, Object element);
    Object remove(int index);
    abstract boolean addAll(int index, Collection c);

    // Search access
    int indexOf(Object o);
    int lastIndexOf(Object o);

    ListIterator listIterator();
    ListIterator listIterator(int index);
    List subList(int from, int to); //range-view
}
```

Η διεπαφή List *(6/6)*

Iteration:

ListIterator listIterator( );

ListIterator listIterator(int index);

//ξεκινά από την θέση index(0 θέση το 1ο στοιχείο)

- ***Μέθοδοι που κληρονομεί:***

  boolean hasNext( );

  Object next( );

  void remove( );

- ***Πρόσθετες μέθοδοι:***

  boolean hasPrevious() *//Iterating backwards*

  Object previous()

ArrayList *(1)*

Είναι ένας **δυναμικός χώρος μνήμης** για συλλογή *(collection)* διατεταγμένων αντικειμένων/στοιχείων.

- Ακολουθεί την *generics* – φιλοσοφία.
- Μπορεί να οριστεί με αρχικό μέγεθος, να αυξάνει (προσθήκη στοιχείων) ή να μειώνεται ανάλογα (διαγραφή στοιχείων).
- Πρέπει να καθορίζεται ο τύπος των στοιχείων της λίστας μέσα στις τριγωνικές αγκύλες **<>**:

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

παράδειγμα:

```
ArrayList<String> list = new ArrayList<String>(20);
```

ArrayList *(2)*

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

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

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

- Πρέπει να εισάγεται *(import)* το πακέτο **java.util** (δηλ.

  import java.util.* ή μόνο το πακέτο ArrayList (δηλ.

  import java.ArrayList

ArrayList *(3)*

- Μετά τον ορισμό της λίστας με τύπο – wrapper, μπορούμε να χειριστούμε τις βασικές τιμές με 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 ----→

ArrayList *(4)*

```
public static void DiagrafiZygon(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 *(1/2)*

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

Σημαντικές Μέθοδοι της ArrayList *(2/2)*

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

Χρήση των σημαντικότερων μεθόδων *(1/4)*

Το μέγεθος της ArrayList

- Μέγεθος μιας ArrayList είναι ο συνολικός αριθμός των στοιχείων της: **int size = s.size();**

Εύρεση της θέσης ενός στοιχείου της ArrayList

- Με την χρήση της indexOf():
  int index = s.indexOf("Skiathos");
  //εύρεση του στοιχείου Skiathos στη λίστα

Χρήση της απλής for ή της foreach για την προσπέλαση μιας λίστας

```
for (int i = 0; i < stringList.size(); i++)
        String stoiheio = s.get(i);
        System.out.println("Stoiheio " + i + " : " + stoiheio);  }
```

Χρήση των σημαντικότερων μεθόδων *(2/4)*

```
for(String stoiheio : s){
        System.out.println("Stoiheio : " + stoiheio); }
```

Έλεγχος αν ένα ArrayList είναι άδειο

Με δύο τρόπους:

1) με την μέθοδο ***isEmpty()***

```
boolean result = s.isEmpty();
//η isEmpty() επιστρέφει true αν η λίστα είναι άδεια
```

2) με την μέθοδο ***size():***

```
if(s.size()==0) {System.out.println("H lista einai adeia");}
```

Χρήση των σημαντικότερων μεθόδων *(3/4)*

Διαγραφή στοιχείου από μια ArrayList

*Με δύο τρόπους χρήσης της **remove()**:*

1) διαγραφή στοιχείου σε συγκεκριμένη θέση (όταν την γνωρίζουμε): π.χ. **s.remove(0);**
2) διαγραφή συγκεκριμένου στοιχείου: π.χ. **s.remove(stoiheio);**

Αντιγραφή όλων των στοιχείων μιας λίστας σε μια άλλη

*Με την μέθοδο **addAll(collection c)**, π.χ.:*

```
ArrayList<String> clone1 = new ArrayList<String>();
clone1.addAll(s);
```

Αντικατάσταση ενός στοιχείου σε συγκεκριμένη θέση

*Με την μέθοδο **set(int i, O object)**,π.χ.:* **s.set(7, "Santorini");**

Χρήση των σημαντικότερων μεθόδων *(4/4)*

Διαγραφή όλων των στοιχείων της λίστας

Με την μέθοδο ***clear(),*** π.χ.: **s.clear();**

Δημιουργία μιας ArrayList από απλό Array

Με την μέθοδο ***Arrays.asList(T... a),*** π.χ.:

```
ArrayList s=Arrays.asList(new String[]{"Skiathos", "Kriti", "Halkidiki"});
//μετά την δημιουργία μπορούμε να αλλάξουμε και τα στοιχεία
```

Μετατροπή μιας ArrayList σε απλό Array

Με την μέθοδο ***toArray(T[] a),*** π.χ.:

```
String[] listArray = new String[s.size()];
String[] aploArray = s.toArray(listArray);
```

Προσπέλαση στα στοιχεία της ArrayList με τον Iterator και την While

Η **Iterator** και **ListIterator** 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
- την μέθοδο ***next()*** που επιστρέφει το επόμενο στοιχείο

```
Iterator<String> iterator = Lista.iterator();
while(iterator.hasNext()){
    System.out.println(iterator.next());
}
ListIterator<String> listIterator = Lista.listIterator();
while(listIterator.hasNext()){
    System.out.println(listIterator.next());
}
```

Διαφορές με τα απλά arrays

Στην κατασκευή:

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

Στην αποθήκευση τιμών:

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

Στην ανάκτηση τιμών:

```
String s = names[0];
String s = list.get(0);
```

Παραδείγματα *(1/3)*

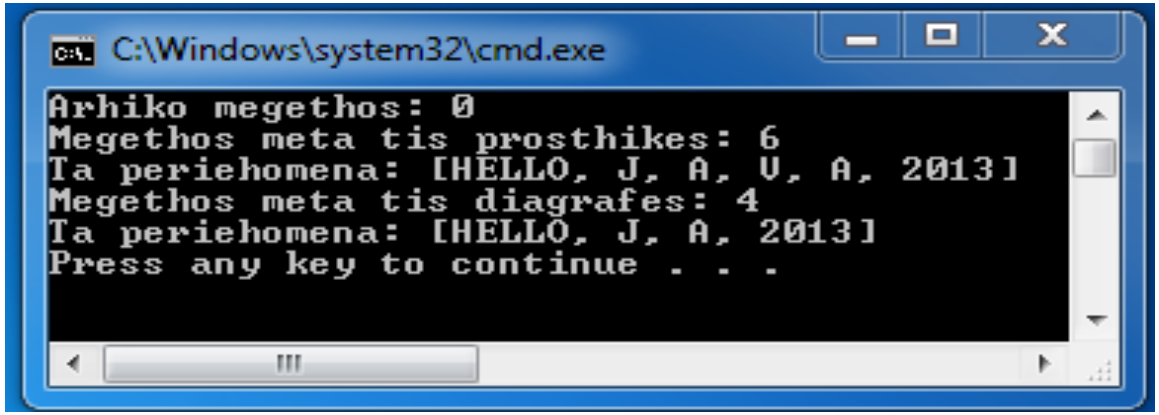
Παράδειγμα απλού ArrayList

Δοκιμάζουμε τις μεθόδους size(), add() και remove(). Επειδή δεν χρησιμοποιούμε generic - ArrayList θα λάβουμε προειδοποιητικό μήνυμα κατά την 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");
```

Παραδείγματα *(2/3)*

```
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);
  }
}
```



Παραδείγματα *(3/3)*

Παράδειγμα generic - ArrayList

Μια πιο ασφαλής παραλλαγή της **ArrayList (generics).**

```
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);
   }
  }
```