
JAVA
ArrayLists

ARRAYLISTS

Στην Java, τα arrayLists είναι μια κλάση στην Java που υλοποιεί τη διεπαφή (interface) List και χρησιμοποιείται για να καταχωρήσουμε **αντικείμενα**. Αντιπροσωπεύει ένα δυναμικού μεγέθους πίνακα, δηλαδή έναν πίνακα που μπορεί να αυξηθεί ή να μειωθεί κατά τη διάρκεια της εκτέλεσης του προγράμματος. Το ArrayList παρέχει πολλές χρήσιμες μεθόδους για την προσθήκη, ανάγνωση, ενημέρωση και διαγραφή στοιχείων από τη λίστα.

Για να δημιουργήσετε ένα arrayList το κάνετε ως εξής:

```
ArrayList<Τύπος> myList = new ArrayList<>();
```

Αρχικά δηλώνεται η κλάση ArrayList συνέχεια δυό <> όπου εντός των παρενθέσεων δηλώνεται ο τύπος του στοιχείου που θα αποθηκεύει (πχ String), στη συνέχεια το όνομα της λίστας και στο τέλος = new ArrayList<>() ώστε να δημιουργηθεί ένα αντικείμενο της κλάσης ArrayList

Περιορισμοί

Στην ArrayList μπορείτε να αποθηκεύσετε σχεδόν οποιοδήποτε αντικείμενο. Ωστόσο, υπάρχουν μερικοί **περιορισμοί** λόγω του τρόπου λειτουργίας της γλώσσας:

Δεν μπορείτε να αποθηκεύσετε πρωτότυπα δεδομένα (primitive types) όπως int, char, boolean, κ.λπ., απευθείας σε ένα ArrayList. Αντ' αυτού, πρέπει να χρησιμοποιήσετε τις αντίστοιχες wrapper κλάσεις* όπως Integer, Character, Boolean, κ.λπ.

Εάν θέλετε να αποθηκεύσετε αντικείμενα διαφορετικών κλάσεων σε ένα ArrayList, τότε αυτά τα αντικείμενα πρέπει να κληρονομούν από μια κοινή υπερκλάση* ή να υλοποιούν ένα κοινό interface*.

Ορισμένες από τις βασικές μεθόδους που παρέχονται από την κλάση ArrayList είναι:

add(E element): Προσθέτει ένα στοιχείο στο τέλος της λίστας.

add(int index, E element): Προσθέτει ένα στοιχείο σε συγκεκριμένη θέση στη λίστα.

get(int index): Επιστρέφει το στοιχείο που βρίσκεται σε συγκεκριμένη θέση στη λίστα.

set(int index, E element): Ενημερώνει το στοιχείο που βρίσκεται σε συγκεκριμένη θέση στη λίστα με ένα νέο στοιχείο.

remove(int index): Αφαιρεί το στοιχείο που βρίσκεται σε συγκεκριμένη θέση στη λίστα.

size(): Επιστρέφει τον αριθμό των στοιχείων στη λίστα.

contains(Object element): Ελέγχει εάν η λίστα περιέχει ένα συγκεκριμένο στοιχείο.

isEmpty(): Ελέγχει εάν η λίστα είναι άδεια.

clear(): Αδειάζει τη λίστα, διαγράφοντας όλα τα στοιχεία.

indexOf(Object element): Επιστρέφει την πρώτη εμφάνιση ενός στοιχείου στη λίστα, εάν υπάρχει, αλλιώς επιστρέφει -1.

lastIndexOf(Object element): Επιστρέφει την τελευταία εμφάνιση ενός στοιχείου στη λίστα, εάν υπάρχει, αλλιώς επιστρέφει -1.

```java
public class Arraylists {
    public static void main(String[] args) {
        List<String> names = new ArrayList<>();

        // Προσθέτει στοιχεία στη λίστα
        names.add("Peter");
        names.add("John");
        names.add("Maria");
        names.add("Sonia");

        // Προσθέτει στοιχείο στη λίστα στη θέση 4
        names.add( index: 4,  element: "Tony");

        // Επιστρέφει το στοιχείο απο τη λίστα στη θέση 1
        String secondElement = names.get(1);

        // Διαγράφει το στοιχείο απο τη λίστα στη θέση 3
        names.remove( index: 3);

        // Επιστρέφει το μέγεθος της λίστας
        names.size();

        // Ελέγχει αν υπάρχει το στοιχείο στη λίστα και επιστρέφει true / false
        boolean contains = names.contains("Tony");

        // Αδειάζει τη λίστα
        names.clear();

        // Επιστρέφει τη θέση ενός στοιχείου, αν δεν υπάρχει επιστρέφει -1,
        // Αν υπάρχουν πολλαπλά ίδια στοιχειά επιστέφει το πρώτο απο αυτά
        names.indexOf("Peter");

        // Επιστρέφει τη θέση ενός στοιχείου, αν δεν υπάρχει επιστρέφει -1,
        // Αν υπάρχουν πολλαπλά ίδια στοιχειά επιστέφει το τελευταίο απο αυτά
        names.lastIndexOf( o: "Peter");
    }
}
```

WRAPPER CLASSES

Οι wrapper classes είναι κλάσεις που χρησιμοποιούνται για να μετατρέψουν τους πρωτογενείς τύπους δεδομένων της Java (όπως int, char, boolean, κλπ) ως αντικείμενα. Αυτές οι κλάσεις βρίσκονται στο πακέτο java.lang και παρέχουν μεθόδους και λειτουργίες που διευκολύνουν τη χρήση των πρωτογενών τύπων ως αντικείμενα στην Java.

Οι πιο συνηθισμένες Wrapper classes είναι:

Integer: Αναπαριστά έναν ακέραιο τύπου **int.**

Double: Αναπαριστά έναν δεκαδικό τύπου **double.**

Boolean: Αναπαριστά μια λογική τιμή **true ή false.**

Character: Αναπαριστά έναν χαρακτήρα τύπου **char.**

Byte: Αναπαριστά έναν ακέραιο τύπου **byte.**

Short: Αναπαριστά έναν ακέραιο τύπου **short.**

Long: Αναπαριστά έναν ακέραιο τύπου **long.**

Float: Αναπαριστά έναν δεκαδικό τύπου **float.**

INTERFACES

Interface είναι ενας τύπος κλάσης που περιέχει μεταβλητές που ορίζονται ως σταθερές και μεθόδους χωρίς υλοποίηση. Αποτελεί ένα συμβόλαιο (contract) που περιγράφει τις λειτουργίες που πρέπει να υποστηρίζει μια κλάση έναν αυτή υλοποιεί το συγκεκριμένο interface. Ένα interface ορίζεται με τη χρήση της λέξης- **interface** και περιέχει μόνο δηλώσεις μεθόδων, συνήθως χωρίς υλοποίηση.

Ορισμένα βασικά χαρακτηριστικά των interfaces είναι:

- Μια κλάση μπορεί να υλοποιήσει πολλά interfaces. Αυτό επιτρέπει την υιοθέτηση πολλαπλής κληρονομικότητας σε Java.
- Οι μέθοδοι σε ένα interface είναι αυτόματα δημόσιες και αφηρημένες (abstract). Δεν πρέπει να περιέχουν υλοποίηση.
- Ένα interface μπορεί να περιλαμβάνει σταθερές (constants), οι οποίες είναι αυτόματα δημόσιες, σταθερές και τελικές.
- Οι κλάσεις που υλοποιούν ένα interface πρέπει να υλοποιήσουν **όλες** τις μεθόδους που ορίζονται στο interface, διασφαλίζοντας έτσι τη συμμόρφωση προς το συμβόλαιο του interface.

```java
public interface Shape {
    double P = 3.14;  2 usages

    2 implementations
    double getArea();  2 usages

    2 implementations
    double getPerimeter();  2 usages
}
```

```java
public class Circle implements Shape{

    private double radius;  3 usages

    @Override  2 usages
    public double getArea() {
        return Shape.P * Math.pow(radius, 2);
    }

    @Override  2 usages
    public double getPerimeter() {
        return 2 * Shape.P * radius;
    }

    public Circle(double radius) {  1 usage
        this.radius = radius;
    }
}
```

```java
public class Rectangle implements Shape{

    private double length;  3 usages
    private double width;  3 usages

    @Override  2 usages
    public double getArea() {
        return length * width;
    }

    @Override  2 usages
    public double getPerimeter() {
        return 2 *(length + width);
    }

    public Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }
}
```

```java
public class Main {
    public static void main(String[] args) {
        Shape circle = new Circle( radius: 3.0);

        System.out.println("Circle's area: " + circle.getArea());
        System.out.println("Circle's perimeter: " + circle.getPerimeter());

        Shape rectangle = new Rectangle( length: 2.0,  width: 3.0);

        System.out.println("Rectangle's area: " + rectangle.getArea());
        System.out.println("Rectangle's perimeter: " + rectangle.getPerimeter());

    }
}
```

ABSTACT CLASSES

Abstract class είναι μια κλάση που δηλώνεται με την λέξη-κλειδί "abstract" και μπορεί να περιέχει τόσο αφηρημένες μεθόδους (χωρίς υλοποιήση) όσο και κανονικές μεθόδους. Η αφηρημένη κλάση χρησιμοποιείται συνήθως ως βάση για άλλες κλάσεις και **δεν μπορεί να δημιουργηθεί ένα αντικείμενο από αυτή καθαυτή.**

Μια αφηρημένη κλάση μπορεί να περιέχει αφηρημένες μεθόδους, που δεν έχουν υλοποίηση και **πρέπει να οριστούν από τις κλάσεις που την κληρονομούν**. Ενώ με τις υλοποιημένες μεθόδους επιτρέπει την υλοποίηση συγκεκριμένης λειτουργικότητας από κάθε κλάση που κληρονομεί την αφηρημένη κλάση.

```java
abstract class Animal {
    private String name;

    public Animal(String name) {
        this.name = name;
    }

    public abstract void makeSound();

    public void sleep() {
        System.out.println(name + " is sleeping");
    }
}

class Dog extends Animal {
    public Dog(String name) {
        super(name);
    }

    @Override
    public void makeSound() {
        System.out.println("Woof!");
    }
}
```

```java
class Cat extends Animal {
    public Cat(String name) {
        super(name);
    }

    @Override
    public void makeSound() {
        System.out.println("Meow!");
    }
}

public class Main {
    public static void main(String[] args) {
        Animal dog = new Dog("Max");
        dog.makeSound();
        dog.sleep();

        Animal cat = new Cat("Luna");
        cat.makeSound();
        cat.sleep();
    }
}
```