Core Java

آموزش ساده و آسان جاوا

به نام خدا

تقدیم به هموطنان عزیزم

جاوا را با لذت یاد بگیرید!

Core Java

آموزش ساده و آسان جاوا

آموزش زبان برنامه نویسی جاوا

Collections در جاوا

موضوع: کلاس ArrayList

جلسه: دوم

نویسنده : رحمان زارعی

جاوا را ساده،آسان و شیرین بنوشید!!!

Core Java

آموزش ساده و آسان جاوا

سلام. دوست من، امروز قصد داریم به کلاس ArrayList یکی از کلاس های موجود در فریم ورک Collection در جاوا بپردازیم. ما پیش تر در آموزش مفاهیم جاوا، جسله سی و هشتم به کلاس ArrayList پرداخته ایم، به دلیل این که قصد داریم در یک بسته آموزشی مجزا به فریم ورک Collection در جاوا بپردازیم و کلاس ArrayList نیز یکی از کلاس های موجود در فریم ورک Collection ها است، دوباره همون جلسه سی و هشتم رو با ویرایش در قالب جلسه دوم آموزش Collection ها در جاوا آورده ایم. اصلا گیج نشید! کلا اگه قبلا ArrayList رو خوندید و اشنایی دارید چه اشکالی داره یکبار دیگه یک مروری داشته باشید ☺ ☺ ☺

کلاس ArrayList

کلاس ArrayList از یک **آرایه پویا**(dynamic) برای ذخیره سازی عناصر استفاده می کند. کلاس ArrayList ، کلاس AbstractList را به ارث برده و اینترفیس List را implements کرده است.

در کل ArrayList را یک آرایه تصور کنید که پویا می باشد.منظور از پویا بودن چیست؟! یعنی این که هر وقت خواستید خانه های ArrayList را کم و زیاد کنید!!! به عبارتی می توانید خانه های آن را حذف یا اضاف کنید کاری که در آرایه ها نمی توانستیم انجام دهیم.

آرایه ها در جاوا دارای طول ثابتی هستند، پس از ایجاد آرایه ها ، نمی توانیم طول آنها را کم یا زیاد کنیم. برای تعریف یک ارایه باید از قبل طول ارایه و تعداد عناصری که قراره در آرایه قرار بگیرند را بدانیم اما برای تعریف ArrayList نیاز به دانستن طول و تعداد عناصر نداریم و هر موقع خواستیم می توانیم تعداد عناصر درون ArrayList را به دلخواه کم یا زیاد کنیم.پس نتیجه میگیریم ArrayList محدودیت های طول و اندازه آرایه را ندارد.

نکته

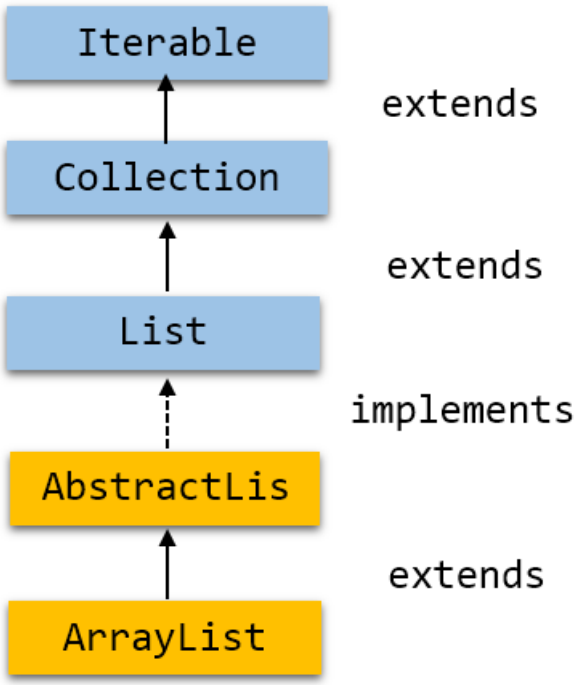
ArrayList *می تواند عناصر تکراری داشته باشد*

Core Java

آموزش ساده و آسان جاوا

سلسله مراتب کلاس ArrayList در جاوا:

نمودار موجود در تصویر(۱) نشان می دهد که کلاس ArrayList ، کلاس AbstractList ای که اینترفیس List را implements کرده را به ارث برده است. همچنین اینترفیس `List` اینترفیس `Collection` را به ارث برده است و اینترفیس `Collection` اینترفیس `Iterable` را به ارث برده است.



تصویر(۱) - نمودار سلسله مراتب `ArrayList` در جاوا

تعریف کلاس ArrayList در برنامه:

برای تعریف کلاس `ArrayList` در برنامه خود باید پکیج زیر را بالای کلاس خود `import` کنید:

```
import java.util.ArrayList;
```

Core Java

آموزش ساده و آسان جاوا

سازنده های کلاس ArrayList :

سازنده	توضیح
`ArrayList()`	برای ایجاد یک `arraylist` خالی استفاده می شود
`ArrayList(Collection c)`	برای ایجاد یک `arraylist` که با عناصر موجود در `collection` (مجموعه) `c` مقدار دهی اولیه می شود. به زبان ساده برای ایجاد یک `arraylist` که با مقادیر موجود در مجموعه `c` مقداردهی اولیه می شود. طبق نمودار موجود در تصویر(۱) `ArrayList` یک `collection` حساب می شود.
`ArrayList(int capacity)`	برای ایجاد یک `arraylist` با ظرفیت اولیه مشخص استفاده می شود. منظور از ظرفیت یا `capacity` اندازه خانه های `arraylist` است که برای ذخیره عناصر موجود در `arraylist` استفاده می شود. با هر بار اضاف شدن یک عنصر به `arraylist` به طور خودکار ظرفیت `arraylist` افزایش می یابد. به همین دلیل است که `arraylist` یک آرایه پویا است و می توان اندازه آن را کم و زیاد کرد.

نکته

ما در جلسه اول ، مفاهیم `Collection` در جاوا را بررسی کرده ایم.

Core Java

آموزش ساده و آسان جاوا

متد های کلاس ArrayList :

سازنده	توضیح
`void add(int index, Object element)`	این متد برای درج کردن (اضاف کردن) یک عنصر مشخص در موقعیت ایندکس خاص از `ArrayList` استفاده می شود. به زبان ساده برای اضاف کردن یک عنصر به خانه شماره **index** ام `ArrayList` مورد استفاده قرار می گیرد.
`boolean addAll(Collection c)`	این متد برای افزودن تمام عناصر یک مجموعه (`Collection`) مشخص `c` را به انتهای لیست `ArrayList` است. اگر مجموعه ای (`Collection`) که قراره به `ArrayList` اضاف شود مقدارش `null` بود ، برنامه دچار استثنای `NullPointerException` می شود.
`void clear()`	برای حذف همه عناصر موجود در لیست `ArrayList` استفاده می شود.
`int lastIndexOf(Object o)`	این متد یک عنصر را به عنوان پارامتر می گیرد و در صورت وجود این عنصر در `ArrayList` ، آخرین ایندکس آن را برمی گرداند. منظور از آخرین ایندکس این است که فرض کنید عنصری که قراره ایندکس اش را پیدا کنیم در `ArrayList` تکراری باشد یا به عبارتی چند مورد از آن وجود داشته باشد، این متد آخرین ایندکس این عنصر را پیدا کرده و برمی گرداند. به شماره خانه عنصر موجود در `ArrayList` ایندکس یا اندیس می گویند.

Core Java

آموزش ساده و آسان جاوا

اگر شی داده شده در ArrayList موجود باشد شماره ایندکس آن را برمی گرداند در صورت عدم وجود شی در ArrayList مقدار -۱ را برمی گرداند.	
تمام عناصر موجود در یک ArrayList را در قالب یک آرایه از نوع کلاس Object بر میگرداند. کلاس Object پدر همه کلاس ها در جاوا می باشد. به زبان ساده برای تبدیل یک ArrayList به یک آرایه از نوع Object از این متد استفاده می شود. در این روش تبدیل ArrayList به آرایه ، به دلیل این که نوع برگردانده شده از نوع Object هست باید از تبدیل نوع یا casting استفاده کنید ، به عبارتی هر عنصر از آرایه را از نوع Object به نوع دلخواه باید cast یا تبدیل کنید.	`Object[] toArray()`
برای تبدیل ArrayList به آرایه ای از نوع مشخص از این متد استفاده می کنیم. مزیتی که این متد دارد دیگر نیازی به casting کردن عناصر آرایه نداریم و مستقیم نوع آرایه از نوع عناصر ArrayList می شود.	`T[] toArray(T[] a)`
برای اضاف کردن یک عنصر مشخص به انتهای لیست موجود در ArrayList استفاده می شود.	`boolean add(Object o)`
این متد برای افزودن تمام عناصر مجموعه (Collection) مشخص c از خانه index ام به بعد ArrayList استفاده می شود.	`boolean addAll(int index, Collection c)`
کپی از یک ArrayList را برای ما برمیگرداند. به زبان ساده یک کپی از شی ArrayList که این متد را صدا زده است را برمی گرداند.	`Object clone()`

بر عکس متد lastIndexOf این متد اولین ایندکس عنصری که به عنوان پارامتر گرفته در صورت وجود در ArrayList برمی گرداند. اگر شی داده شده در ArrayList موجود باشد شماره ایندکس آن را برمی گرداند در صورت عدم وجود شی در ArrayList مقدار -۱ را برمی گرداند.	`int indexOf(Object o)`

مقایسه (Collection)مجموعه Generic و غیر Generic در جاوا :

فریم ورک collection قبل از JDK 1.5 غیر Generic بوده و از JDK 1.5 به بعد، Generic می باشد.
(collection)مجموعه Generic جاوا امکانات جدیدی به ما می دهد، امکاناتی که به ما اجازه می دهد تا تنها یک نوع شی در مجموعه(collection) داشته باشید. یعنی نوع عناصری که قراره در مجموعه(collection) ما قرار بگیرند را می توانیم مشخص کنیم. در حال حاضر (Collection)مجموعه Generic ایمن تر می باشد و در زمان اجرای برنامه نیاز به تبدیل نوع یا typecasting نداریم.

روش قدیمی ایجاد یک (Collection)مجموعه غیر Generic در جاوا:

مثالی از ایجاد یک (Collection)مجموعه غیر Generic در جاوا:
بازم میگم ArrayList خود یک مجموعه حساب می شود مثل سایر مجموعه های دیگر.

```
ArrayList al=new ArrayList();
```

Core Java

آموزش ساده و آسان جاوا

روش جدید ایجاد یک (Collection)مجموعه Generic در جاوا:

مثالی از ایجاد یک (Collection)مجموعه Generic در جاوا:

```
ArrayList<String> al=new ArrayList<String>();
```

- ما در یک مجموعه Generic نوع عناصر موجود در ArrayList را مشخص می کنیم.پس ArrayList مجبور است که فقط نوع مشخصی از اشیا را در خود داشته باشد.اگر شما سعی کنید نوع دیگری از شی را به یک ArrayList اضاف کنید، خطای زمان کامپایل رخ می دهد.
- در مثال بالا ArrayList ما تنها اشیا یا عناصری که از نوع String باشد را می پذیرد.

- در کل اگر برای یک Collection نوعش را مشخص کریم می شود یک مجموعه Generic و اگر نوع آن را مشخص نکردیم میشه یک مجموعه غیر جنریک می شود.

- Generic خود یکی از مباحث جاوا می باشد که در یک بسته آموزشی رایگان به طور مفصل به آن می پردازیم.

- *توضیحات رو بیخیال میشیم!! میریم سراغ مثال های ArrayList در جاوا:*

نکته: اگه توضیحات رو متوجه نشدید به مثال ها دقت کنید برای خودتون تغییرشون بدید یواش یواش یاد می گیرید.

Example1:

```
package javalikeArrayList;

import java.util.*;

class TestCollection1 {
     public static void main(String args[]) {
          ArrayList<String> list = new ArrayList<String>();// Creating arraylist
          list.add("Ravi");// Adding object in arraylist
          list.add("Vijay");
          list.add("Ravi");
```

```
            list.add("Ajay");
            // Traversing list through Iterator
            Iterator itr = list.iterator();
            while (itr.hasNext()) {
                  System.out.println(itr.next());
            }
      }
}
```

خروجی(output)::

```
Ravi
Vijay
Ravi
Ajay
```

توضیحات:

```
import java.util.*;
```

- قبل از هرچیز پکیج بالا را در برنامه خود برای استفاده از کلاس ArrayList ، import می کنیم.

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

- یک ArrayList تعریف کرده و نوع آن را String گذاشته ایم.
- برای تعریف یک ArrayList در جاوا بصورت زیر عمل می کنیم:

۱. نام کلاس ArrayList را تایپ می کنیم:

```
ArrayList
```

۲. نوع عناصر یا اشیایی که قراره در ArrayList ما قرار بگیرند را بین دو علامت < > قرار می دهیم.

```
ArrayList<>
```

در این مثال نوع اشیای ArrayList را از نوع String انتخاب کرده ایم.پس:

```
ArrayList<String>
```

۳. حال نامی را برای شی ایجاد شده از نوع کلاس ArrayList انتخاب می کنیم:

Core Java

آموزش ساده و آسان جاوا

```
ArrayList<String> list
```

۴. بعد از نام ، علامت = گذاشته و سمت راست علامت مساوی از کلمه کلیدی new استفاده می کنیم.

```
ArrayList<String> list = new
```

۵. دوباره نام کلاس ArrayList به همراه نوع عناصری که قراره در ArrayList قرار بگیرند را بعد از کلمه کلیدی new تکرار می کنیم:

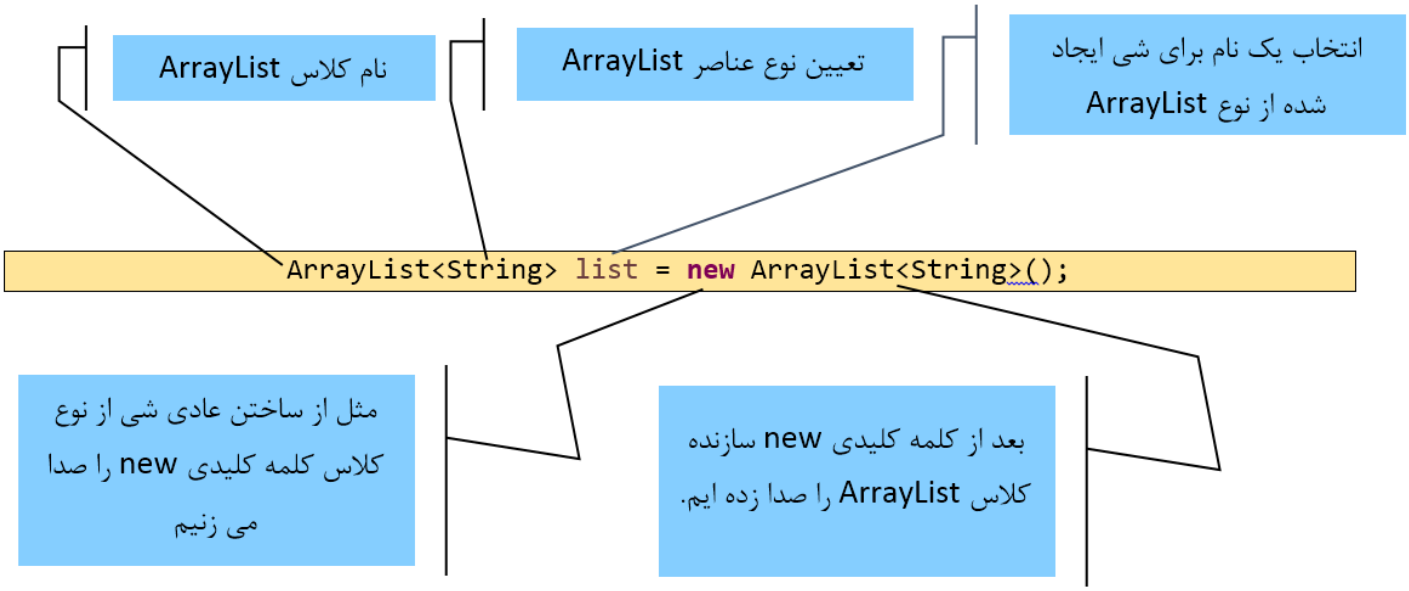
```
ArrayList<String> list = new ArrayList<String>
```

۶. حال یک پرانتز باز و بسته () و یک ; آخر خط دستور قرار می دهیم!

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

- به همین راحتی یک شی از نوع کلاس ArrayList که نوع عناصر درون ان از نوع String می باشد ایجاد کردیم!!!!

خلاصه کل ماجرا ساخت شی از کلاس ArrayList را در تصویر(۲) ببنید:



تصویر(۲) - روش ایجاد شی از کلاس ArrayList در جاوا

Core Java

آموزش ساده و آسان جاوا

ادامه توضیحات مثال....

```
list.add("Ravi");// Adding object in arraylist
list.add("Vijay");
list.add("Ravi");
list.add("Ajay");
```

- افزودن اشیا خود به **arraylist**، در اینجای اشیای ما از نوع کلاس **String** می باشد.

Iterator

نکته مهم: همان طور که در جلسه اول **Collections** در جاوا گفتیم برای پیمایش یک **Collection** از امکانات اینترفیس **Iterator** استفاده می کنیم.

- پیمایش یک **Collection** چه کاربردی دارد؟ گاهی نیاز داریم عناصر یک **Collection** را دستکاری یا چاپ کنیم به همین خاطر نیاز به پیمایش عناصر **Collection** پیدا می کنیم.

- همان طور که می دانید **Arraylist** طبق تصویر(۱) یک **Collection** حساب می شود. پس می توانیم عناصر یک **Arraylist** را با امکانات اینترفیس **Iterator** پیمایش کنیم.

- یکی از راه های پیمایش **Collection** ها استفاده از امکانات اینترفیس **Iterator** می باشد ، راه دیگری نیز وجود دارد که جلوتر بررسی خواهیم کرد.

روش استفاده از اینترفیس **Iterator** برای پیمایش یک **Arraylist** را در همین مثال قبل بررسی می کنیم:

- ابتدا یک شی از نوع اینترفیس **Iterator** تعریف می کنیم:

```
Iterator itr =
```

Core Java

آموزش ساده و آسان جاوا

اینترفیس Iterator یک متد با نام iterator() دارد که به کمک این متد می توانیم از اینترفیس Iterator شی ایجاد کنیم به عبارتی یک پیمایش گر برای پیمایش Collection مورد نظر که در اینجا یک Arraylist هست بسازیم.

چطور این کارو انجام بدیدم؟ کافیه که از طریق شی Collection مورد نظر متد iterator() را صدا بزنیم. در اینجا شی Collection ما نامش list که از نوع کلاس Arraylist هستش می باشد. پس روش ایجاد پیمایش گر برای پیمایش عناصر Arraylist ما به صورت زیر است: (این توضیحات مربوط به مثال قبل می باشد)

```
Iterator itr = list.iterator();
```

- همان طور که مشاهده می کنید برای پیمایش شی list که از نوع Arraylist است ، یک شی از نوع اینترفیس Iterator تعریف کرده و از طریق شی list متد iterator() را صدا زده ایم.
- برای پیمایش همه Collection از این روش می توانیم استفاده کنیم. در اینجا list از نوع Arraylist بود ، برای پیمایش Collection های دیگر کافیه شی list از نوع Collection های دیگر باشد.

به طور کلی برای پیمودن عناصر درون یک مجموعه که در اینجا یک arraylist با استفاده از Iterator سه گام زیر را بر می داریم:

۱. ابتدا یک شی از نوع Iterator ایجاد کرده و از طریق شی از نوع کلاس ArrayList متد iterator() را صدا می زنیم.

```
Iterator itr = list.iterator();
```

۲. یک حلقه ایجاد کرده و شرط آن را به صورت زیر قرار می دهیم:

```
.hasNext() شی از نوع اینترفیس Iterator
```

```
while (itr.hasNext()) {

}
```

- تا زمانی که شرط حلقه برقرار است یعنی هنوز به انتهای لیست نرسیدیم حلقه تکرار می شود.

- متد hasNext هنگام پیمایش ArrayList چک می کند که آیا عنصر بعدی نیز وجود دارد یا خیر.اگر وجود داشت شرط حلقه true است و به پیمایش ادامه می دهیم در غیر این صورت پیمایش متوقف می شود.

۳. در بدنه حلقه هر عنصر موجود در ArrayList را با صدا زدن متد next() به دست می آوریم:

```
.next() شی از نوع اینترفیس Iterator
```

- متد next() عنصر بعدی را به ما می دهد.

در اینجا با پیمودن ArrayList عناصر آن را چاپ کرده ایم:

```
while (itr.hasNext()) {
                System.out.println(itr.next());
            }
```

- هر عنصری که از طریق متد next() دریافت می کنیم را چاپ کرده ایم.

- پایان توضیحات مثال قبل.

در کل دو راه برای پیمودن یک ArrayList وجود دارد:

۱. با اینترفیس Iterator

۲. با حلقه for-each

- در مثال قبل ما ArrayList را با استفاده از اینترفیس Iterator پیمایش کردیم حال در مثل بعدی قصد داریم یک ArrayList را با حلقه for-each پیمایش کنیم.

Example2:

```
package javalikeArrayList;

import java.util.*;

class TestCollection2 {
     public static void main(String args[]) {
          ArrayList<String> al = new ArrayList<String>();
          al.add("javalike");
          al.add("javapro");
```

```
            al.add("java for iranian");
            al.add("borazjan");
            for (String obj : al)
                  System.out.println(obj);
      }
}
```

خروجی(output)::

```
javalike
javapro
java for iranian
borazjan
```

توضیحات:

```
      for (String obj : al)
                  System.out.println(obj);
      }
```

- در اینجا با حلقه for-each ، ArrayList خود را پیمایش کرده ایم.

احتمالا براتون الان سوال پیش اومده حلقه for-each چیه؟!

حلقه for-each:

حلقه for-each در Java5 معرفی شده است.عمدتا حلقه for-each برای پیمایش آرایه ها و مجموعه ها استفاده می شود. از مزیت های حلقه for-each می توان به حذف رخ دادن باگ های احتمالی و بهتر خوانده شدن کد برنامه می باشد.

مزیت های حلقه for-each:

- کدها را بیشتر قابل خواندن می کند.
- باگ های (خطاهای) احتمالی برنامه نویسی رو حذف می کند.

Core Java

آموزش ساده و آسان جاوا

سینتکس (نحوه نوشتن) حلقه for-each:

```
for(data_type variable : array | collection){
}
```

- حلقه for-each مثل یک حلقه for معمولی هستش که تنها دستورات درون پرانتز آن متفاوت می باشد. برای نوشتن یک حلقه for-each به صورت زیر عمل می کنیم

۱. ابتدا کلمه کلیدی for را تایپ می کنیم:

```
for
```

۲. یک پرانتز باز و بسته جلوی کلمه کلیدی for قرار می دهیم:

```
for()
```

۳. درون پرانتز یک متغیر تعریف کرده و نوع آن را از نوع عناصری از آرایه یا مجموعه ای که قراره پیمایشش کنیم قرار می دهیم:

```
for(data_type variable)
```

۴. بعد از نام متغیر از علامت " : " دو نقطه استفاده می کنیم:

```
for(data_type variable :)
```

۵. بعد از علامت " : " دو نقطه نام شی که از نوع آرایه یا مجموعه یا مثلا ArrayList بود استفاده می کنیم:

```
for(data_type variable : collection)
```

یا

```
for(data_type variable : array)
```

یا

```
for(data_type variable : arraylist)
```

- مثلا در اینجا شی **arraylist** از قبل از نوع کلاس **ArrayList** تعریف کرده ایم.

۶. اگر دستوراتی که میخوایم در بدنه حلقه اجرا شود بیش از یک عدد باشد درون دو "{}" ، آکولاد باز و بسته قرار می دهیم.

```
for(data_type variable : arraylist){

}
```

قبلا با حلقه **for** با داشتن شمارنده و از طریق **index** های ارایه به عناصر ارایه دسرسی پیدا می کردیم و در صورت عدم تطابق مقدار شمارنده با محدوده **index** های آرایه برنامه باگ یا خطا میداد اما در حلقه **for-each** کافیه یک متغیر از نوع عنصر آرایه و شی ارایه رو بهش بدید براتون گاز پلمپ! آرایه رو بدون هیچ خطایی پیمایش می کند ☺

یک مثالم ازش بزنیم برای درک بهتر.....

Example3:

```
package javalikeArrayList;

class ForEachExample1 {
	public static void main(String args[]) {
		int arr[] = { 12, 13, 14, 44 };

		for (int i : arr) {
			System.out.println(i);
		}

	}
}
```

خروجی(output)::

```
12
13
14
44
```

Core Java

آموزش ساده و آسان جاوا

توضیحات:

```
for (int i : arr) {
                System.out.println(i);
        }
```

- در این برنامه آرایه arr از نوع int می باشد و قصد داریم با حلقه for-each این آرایه را پیمایش کنیم.کافیه یک حلقه for تعریف کنیم و درون پرانتز حلقه یک متغیر از نوع int که هم نوع با نوع آرایه می باشد تعریف کنیم و بعد از تعریف متغیر از علامت دو نقطه " : " استفاده می کنیم و بعد از علامت دو نقطه " : " شی آرایه را به کار می بریم .

بریم سراغ ادامه توضیحات مثال۲ که در مورد ArrayList بود!!!!

- در کل هر Collection را می توان از طریق دو روش اینترفیس Iterator و حلقه for-each پیمایش کرد.روش های دیگری هم هست این دو متداول ترین روش است.

ادامه توضیح Example2:

```
    for (String obj : al)
                System.out.println(obj);
    }
```

- در این مثال نیز یک متغیر که نوعش برابر با **نوع عناصر** ArrayList است ، تعریف می کنیم و بعد علامت دو نقطه و بعد از علامت دو نقطه شی که از نوع کلاس ArrayList است بکار می بریم. به همین راحتی ArrayList خود را پیمایش می کنیم!

اشیای کلاس تعریف شده توسط کاربر در ArrayList :

ما می توانیم نوع عناصر یا اشیای ArrayList خود را از نوع یک کلاس تعریف کنیم.مثال زیر نمونه ای از ذخیره سازی اشیای یک کلاس درون ArrayList می باشد.

Core Java

آموزش ساده و آسان جاوا

Exampl4:

نکته: برنامه زیر دارای دو کلاس یا دو فایل به نام های زیر می باشد:

Student.java TestCollection3.java

- این دو کلاس را در کنار هم در یک پکیج قرار دهید و متد main برنامه درون کلاس TestCollection3 می باشد.پس برای اجرای برنامه کلاس TestCollection3 را run کنید.

Student.java

```
package javalikeArrayList;

class Student {
	int rollno;
	String name;
	int age;

	Student(int rollno, String name, int age) {
		this.rollno = rollno;
		this.name = name;
		this.age = age;
	}
}
```

TestCollection3.java

```
package javalikeArrayList;

import java.util.*;

public class TestCollection3 {
	public static void main(String args[]) {
		// Creating user-defined class objects
		Student s1 = new Student(101, "ali", 23);
```

```
            Student s2 = new Student(102, "hasan", 21);
            Student s3 = new Student(103, "sara", 25);
            // creating arraylist
            ArrayList<Student> al = new ArrayList<Student>();
            al.add(s1);// adding Student class object
            al.add(s2);
            al.add(s3);
            // Getting Iterator
            Iterator itr = al.iterator();
            // traversing elements of ArrayList object
            while (itr.hasNext()) {
                  Student st = (Student) itr.next();
                  System.out.println(st.rollno + " " + st.name + " " + st.age);
            }
      }
}
```

خروجی(output)::

```
101 ali 23
102 hasan 21
103 sara 25
```

توضیحات:

```
Student s1 = new Student(101, "ali", 23);
Student s2 = new Student(102, "hasan", 21);
Student s3 = new Student(103, "sara", 25);
```

- از کلاس خود سه شی ایجاد کرده و پارامتر سازنده های آنها را مقداردهی کرده ایم.

```
ArrayList<Student> al = new ArrayList<Student>();
```

- برای ذخیره اشیای یک کلاس درون ArrayList فرقی با سایر داده های دیگر نمیکند! کافیست که نوع عناصر ArrayList را از نوع کلاس مربوطه قرار دهید.

```
al.add(s1);// adding Student class object
al.add(s2);
al.add(s3);
```

Core Java

آموزش ساده و آسان جاوا

- با متد add اشیای کلاس را به ArrayList اضاف می کنیم.

```
Iterator itr = al.iterator();
          // traversing elements of ArrayList object
          while (itr.hasNext()) {
                Student st = (Student) itr.next();
                System.out.println(st.rollno + " " + st.name + " " + st.age);
          }
```

- اینجا دیگه مربوط به پیمایش و نمایش عناصر درون ArrayList می باشد.

```
Student st = (Student) itr.next();
```

- در اینجا که یکی بعد از دیگری عناصر ArrayList را پیمایش می کنیم عمل casting یا تبدیل نوع را انجام داده ایم. چرا؟ عناصری که پیمایش میشه همگی از نوع کلاس Object هستش و برای برنامه قابل تشخیص نیست که این عناصر از نوع کلاس Student می باشد پس مجبوریم عمل casting را انجام بدیم.

مثالی دیگر.....

Exampl5:

```
package javalikeArrayList;
import java.util.*;
class Book {
int id;
String name,author,publisher;
int quantity;
public Book(int id, String name, String author, String publisher, int quantity) {
    this.id = id;
    this.name = name;
    this.author = author;
    this.publisher = publisher;
    this.quantity = quantity;
}
}
public class ArrayListExample {
public static void main(String[] args) {
    //Creating list of Books
    List<Book> list=new ArrayList<Book>();
```

```
    //Creating Books
    Book b1=new Book(101,"Let us C","Yashwant Kanetkar","BPB",8);
    Book b2=new Book(102,"Data Communications & Networking","Forouzan","Mc Graw
Hill",4);
    Book b3=new Book(103,"Operating System","Galvin","Wiley",6);
    //Adding Books to list
    list.add(b1);
    list.add(b2);
    list.add(b3);
    //Traversing list
    for(Book b:list){
        System.out.println(b.id+" "+b.name+" "+b.author+" "+b.publisher+"
"+b.quantity);
    }
}
}
```

خروجی(output)::

```
101 Let us C Yashwant Kanetkar BPB 8
102 Data Communications & Networking Forouzan Mc Graw Hill 4
103 Operating System Galvin Wiley 6
```

مثال از متد addAll(Collection c) در ArrayList :

این متد عناصر یک مجموعه رو به انتهای لیست مجموعه ای دیگر اضاف می کند. منظور من از استفاده از عبارت مجموعه اشاره کلی هستش یک مجموعه میتونه شامل ArrayList و... شود.

Exampl6:

```
package javalikeArrayList;
import java.util.*;
class TestCollection4 {
     public static void main(String args[]) {
          ArrayList<String> al = new ArrayList<String>();
          al.add("jafar");
          al.add("morad");
```

```
            al.add("sara");
            ArrayList<String> al2 = new ArrayList<String>();
            al2.add("maryam");
            al2.add("zahra");
            al.addAll(al2);// adding second list in first list
            Iterator itr = al.iterator();
            while (itr.hasNext()) {
                  System.out.println(itr.next());
            }
      }
}
```

خروجی(output)::

```
jafar
morad
sara
maryam
zahra
```

توضیحات:

- در این برنامه دو **ArrayList** به نام های **al** و **al2** داریم.حال با دستور زیر عناصر **al2** را به انتهای **al** اضاف کرده ایم:

```
al.addAll(al2);
```

مثال از متد **removeAll()** *در* **ArrayList** :

Exampl7:

```
package javalikeArrayList;

import java.util.*;

class TestCollection5 {
```

```
	public static void main(String args[]) {
		ArrayList<String> al = new ArrayList<String>();
		al.add("jafar");
		al.add("morad");
		al.add("sara");
		al.add("hasan");
		al.add("maryam");

		ArrayList<String> al2 = new ArrayList<String>();
		al2.add("maryam");
		al2.add("sara");
		al2.add("jafar");
		al.removeAll(al2);
		System.out
				.println("iterating the elements after removing the
elements of al2...");
		Iterator itr = al.iterator();
		while (itr.hasNext()) {
			System.out.println(itr.next());
		}

	}
}
```

خروجی(output):

```
iterating the elements after removing the elements of al2...
morad
hasan
```

توضیحات:

```
public boolean removeAll(Collection<?> c)
```

- متد removeAll() برای حذف عناصر تکراری میان دو ArrayList استفاده می شود. فرض کنید دو ArrayList با نام های b1 و b2 داشته باشیم. و تعدادی از عناصر b1 و b2 مثل هم باشند. حالا اگر از دستور زیر استفاده کنیم عناصری از b1 که در b2 وجود دارد را حذف می کند:

Core Java

آموزش ساده و آسان جاوا

```
b1.removeAll(b2);
```

- پارامتر متد removeAll() از نوع اینترفیس Collection می باشد. به عبارت دیگر یک شی از Collection مثل ArrayList و.. را می توانیم به آن دهیم.

حالا مثال بالا را بررسی می کنیم:

```
al.removeAll(al2);
```

- در این مثال قصد داریم عناصری از al که در al2 وجود دارد را حذف کنیم.پس تنها عناصری از al حذف می شود که شبیه عناصر موجود al2 باشد.
- پارامتر این متد را شی از نوع کلاس ArrayList قرار داده ایم.

برای درک بهتر مثال دیگری در این زمینه می زنیم.....

Exampl8:

```
package javalikeArrayList;

import java.util.*;

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

		// create an empty array list
		ArrayList<String> color_list = new ArrayList<String>();

		// use add() method to add values in the list
		color_list.add("White");
		color_list.add("Black");
		color_list.add("Red");

		// create an empty array sample with an initial capacity
		ArrayList<String> sample = new ArrayList<String>();

		// use add() method to add values in the list
		sample.add("Green");
		sample.add("Red");
		sample.add("White");
```

```
        // remove all elements from second list if it exists in first list
        sample.removeAll(color_list);

        System.out.println("First List :" + color_list);
        System.out.println("Second List :" + sample);

    }
}
```

خروجی(output):

```
First List :[White, Black, Red]
Second List :[Green]
```

- همان طور که مشاهده می کنید عناصری از sample که در color_list وجود دارد حذف شده است.

مثال از متد retainAll() در ArrayList :

متد retainAll() عناصری از یک لیست که در لیست دیگری وجود ندارد را حذف می کند!

فرض کنید دو ArrayList با نام های b1 و b2 داشته باشیم. و عناصر b1 به صورت زیر باشد:

"ali","mohammad","jafar"

و عناصر b2 نیز بصورت زیر باشد:

"ali","sara","amir"

حال اگر دستور زیر استفاده کنیم:

```
b1.retainAll(b2);
```

عناصر موجود در b1 به شکل زیر تغییر می کنند:

Core Java

آموزش ساده و آسان جاوا

"ali"

در اینجا عناصری از b1 که در b2 وجود نداشت حذف شده است. حالا مثالی در این باره می زنیم:

Exampl9:

```
package javalikeArrayList;

import java.util.*;

class TestCollection6 {
      public static void main(String args[]) {
            ArrayList<String> al = new ArrayList<String>();
            al.add("Ravi");
            al.add("Vijay");
            al.add("Ajay");
            ArrayList<String> al2 = new ArrayList<String>();
            al2.add("Ravi");
            al2.add("Hanumat");
            al.retainAll(al2);
            System.out
                        .println("iterating the elements after retaining the
elements of al2...");
            Iterator itr = al.iterator();
            while (itr.hasNext()) {
                  System.out.println(itr.next());
            }
      }
}
```

خروجی(output):

```
iterating the elements after retaining the elements of al2...
Ravi
```

```
al.retainAll(al2);
```

Core Java

آموزش ساده و آسان جاوا

- تنها عنصری از al که در al2 وجود دارد Ravi است.

مثال دیگر.....

Exampl10:

```
package javalikeArrayList;
import java.util.*;

public class test2 {
   public static void main(String[] args) {

    // create an empty array list
    ArrayList<String> color_list = new ArrayList<String>();

    // use add() method to add values in the list
    color_list.add("White");
    color_list.add("Black");
    color_list.add("Red");

   // create an empty array sample with an initial capacity
    ArrayList<String> sample = new ArrayList<String>();

   // use add() method to add values in the list
    sample.add("Green");
    sample.add("Red");
    sample.add("White");

    System.out.println("First List :"+ color_list);
    System.out.println("Second List :"+ sample);

    sample.retainAll(color_list);

    System.out.println("After applying the method, First List :"+ color_list);
    System.out.println("After applying the method, Second List :"+ sample);

  }
}
```

خروجی(output):

Core Java

آموزش ساده و آسان جاوا

```
First List :[White, Black, Red]
Second List :[Green, Red, White]
After applying the method, First List :[White, Black, Red]
After applying the method, Second List :[Red, White]
```

مرتب سازی عناصر String درون یک ArrayList :

Exampl11:

```
package javalikeArrayList;
import java.util.*;
public class Details  {

      public static void main(String args[]){
         ArrayList<String> listofcountries = new ArrayList<String>();
         listofcountries.add("India");
         listofcountries.add("US");
         listofcountries.add("China");
         listofcountries.add("Denmark");

         /*Unsorted List*/
         System.out.println("Before Sorting:");
         for(String counter: listofcountries){
                 System.out.println(counter);
           }

         /* Sort statement*/
         Collections.sort(listofcountries);

         /* Sorted List*/
         System.out.println("After Sorting:");
         for(String counter: listofcountries){
                 System.out.println(counter);
           }
      }
}
```

Core Java

آموزش ساده و آسان جاوا

خروجی(output):

```
Before Sorting:
India
US
China
Denmark
After Sorting:
China
Denmark
India
US
```

- تنها با دستور ساده زیر می توانید عناصر درون ArrayList خود را مرتب سازی کنید.

```
Collections.sort(listofcountries);
```

- متد sort را با کلاس Collections صدا زده و شی از نوع کلاس ArrayList به عنوان پارامتر به این متد می دهیم.
- مرتب سازی عناصر Integer درون یک ArrayList :

Exampl12:

```
package javalikeArrayList;

import java.util.*;

public class ArrayListOfInteger {

	public static void main(String args[]) {
		ArrayList<Integer> arraylist = new ArrayList<Integer>();
		arraylist.add(11);
		arraylist.add(2);
		arraylist.add(7);
```

Core Java

آموزش ساده و آسان جاوا

```
            arraylist.add(3);
            /* ArrayList before the sorting */
            System.out.println("Before Sorting:");
            for (int counter : arraylist) {
                  System.out.println(counter);
            }

            /* Sorting of arraylist using Collections.sort */
            Collections.sort(arraylist);

            /* ArrayList after sorting */
            System.out.println("After Sorting:");
            for (int counter : arraylist) {
                  System.out.println(counter);
            }
      }
}
```

خروجی(output):

```
Before Sorting:
11
2
7
3
After Sorting:
2
3
7
11
```

مثالی دیگر.....

Exampl13:

```
package javalikeArrayList;

import java.util.ArrayList;
import java.util.Collections;
```

Core Java

آموزش ساده و آسان جاوا

```
import java.util.List;

public class SortArrayList {

	public static void main(String args[]) {

		List<String> unsortList = new ArrayList<String>();

		unsortList.add("CCC");
		unsortList.add("111");
		unsortList.add("AAA");
		unsortList.add("BBB");
		unsortList.add("ccc");
		unsortList.add("bbb");
		unsortList.add("aaa");
		unsortList.add("333");
		unsortList.add("222");

		// before sort
		System.out.println("ArrayList is unsort");
		for (String temp : unsortList) {
			System.out.println(temp);
		}

		// sort the list
		Collections.sort(unsortList);

		// after sorted
		System.out.println("ArrayList is sorted");
		for (String temp : unsortList) {
			System.out.println(temp);
		}
	}

}
```

خروجی(output):

```
ArrayList is unsort
CCC
111
```

```
AAA
BBB
ccc
bbb
aaa
333
222
ArrayList is sorted
111
222
333
AAA
BBB
CCC
aaa
bbb
ccc
```

- **Search کردن یک عنصر در میان عناصر یک ArrayList :**

متدهای از ArrayList که از آنها برای Search کردن عنصری خاص استفاده می کنیم شامل زیر می شوند:

- contains(Object elem)
- indexOf(Object elem)

مثال:

Exampl14:

```
package javalikeArrayList;

import java.util.*;

public class SearchArrayList1 {

	public static void main(String[] args)

	{
```

```
        String searchString = "";
        ArrayList<String> arrayList = new ArrayList<String>();

        arrayList.add("Delhi"); // Adding elements to Arraylist
        arrayList.add("Kolkata");
        arrayList.add("Bangalore");
        arrayList.add("Mumbai");
        arrayList.add("Chennai");
        arrayList.add("Hyderabad");
        arrayList.add("Pune");

        searchString = "Mumbai";
        search(searchString, arrayList);

        searchString = "Bhopal";
        search(searchString, arrayList);

    }

    public static void search(String searchStr, ArrayList<String> aList) {

        boolean found = false;
        Iterator<String> iter = aList.iterator();
        String curItem = "";
        int pos = 0;

        while (iter.hasNext() == true) {
            pos = pos + 1;
            curItem = (String) iter.next();
            if (curItem.equals(searchStr)) {
                found = true;
                break;
            }

        }

        if (found == false) {
            pos = 0;
        }

        if (pos != 0) {
```

```
                System.out.println(searchStr + " City Found in position : " +
pos);
            } else {

                System.out.println(searchStr + " City not found");
            }

      }

}
```

خروجی(output):

```
Mumbai City Found in position : 4
Bhopal City not found
```

مثال دیگر.....

Exampl15:

```
package javalikeArrayList;

import java.util.ArrayList;

public class SearchArrayList2 {
      public static void main(String[] args) {
            ArrayList<String> arrayList = new ArrayList<String>();

            try {
                  arrayList.add("India"); // Adding elements to Arraylist
                  arrayList.add("US");
                  arrayList.add("England");
                  arrayList.add("Australia");
                  arrayList.add("China");
                  arrayList.add("France");
                  arrayList.add("Singapore");
                  arrayList.add("US");
            } catch (Exception e) {
            }
```

```
		boolean Found = arrayList.contains("China");
		System.out.println("ArrayList contain China? : " + Found);

		int index = arrayList.indexOf("China");
		if (index == -1)
			System.out.println("ArrayList does not contain China");
		else
			System.out.println("ArrayList contains China at index :" +
index);

		index = arrayList.indexOf("Malasia");
		if (index == -1)
			System.out.println("ArrayList does not contain Malasia");
		else
			System.out.println("ArrayList contains Malasia at index :" +
index);

		int lastIndex = arrayList.lastIndexOf("US");
		if (lastIndex == -1)
			System.out.println("ArrayList does not contain US");
		else
			System.out
					.println("Last occurrence of US in ArrayList is at
index :"
							+ lastIndex);
	}

}
```

خروجی(output):

```
ArrayList contain China? : true
ArrayList contains China at index :4
ArrayList does not contain Malasia
Last occurrence of US in ArrayList is at index :7
```

Core Java

آموزش ساده و آسان جاوا

مثال از متد clear() در ArrayList :

Exampl16:

```
package javalike;
import java.util.*;

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

    // create an empty array list with an initial capacity
    ArrayList<String> color_list = new ArrayList<String>(7);

    // use add() method to add values in the list
    color_list.add("White");
    color_list.add("Black");
    color_list.add("Red");
    color_list.add("White");
    color_list.add("Yellow");
    color_list.add("Red");
    color_list.add("White");

    // Print out the colors in the ArrayList
    System.out.println("****Color list****");
    for (int i = 0; i < 7; i++)
      {
         System.out.println(color_list.get(i).toString());
      }

    // Remove all the elements using clear()
    color_list.clear();

    //Now size of the list
    System.out.println("After using clear() method, the size of the list is: " +
color_list.size());

      }
}
```

Core Java

آموزش ساده و آسان جاوا

خروجی(output):

```
****Color list****
White
Black
Red
White
Yellow
Red
White
After using clear() method, the size of the list is: 0
```

مثال از متد toArray() در ArrayList :

Exampl17:

```
package javalike;

import java.util.*;

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

          // create an empty array list with an initial capacity
          ArrayList<String> color_list = new ArrayList<String>(5);

          // use add() method to add values in the list
          color_list.add("White");
          color_list.add("Black");
          color_list.add("Red");
          color_list.add("White");
          color_list.add("Yellow");

          System.out.println("Size of list: " + color_list.size());

          // Print the colors in the list
```

```
            for (String value : color_list) {
                  System.out.println("Color = " + value);
            }
            // Create an array from the ArrayList
            Object[] obj = color_list.toArray();

            // Display the contents of the array
            System.out.println("Printing elements from first to last:");
            for (Object value : obj) {
                  System.out.println("Color = " + value);
            }
      }
}
```

خروجی(output):

```
Size of list: 5
Color = White
Color = Black
Color = Red
Color = White
Color = Yellow
Printing elements from first to last:
Color = White
Color = Black
Color = Red
Color = White
Color = Yellow
```

مثال از متد toArray(T[] a) در ArrayList :

Exampl18:

```
package javalike;

import java.util.*;
```

Core Java

آموزش ساده و آسان جاوا

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

		// create an empty array list with an initial capacity
		ArrayList<String> color_list = new ArrayList<String>(5);

		// use add() method to add values in the list
		color_list.add("White");
		color_list.add("Black");
		color_list.add("Red");
		color_list.add("White");
		color_list.add("Yellow");

		System.out.println("Size of the color_list: " + color_list.size());

		// Print the colors in the list
		for (String value : color_list) {
			System.out.println("Color = " + value);
		}
		// Create an array from the ArrayList
		String color_list2[] = new String[color_list.size()];
		color_list2 = color_list.toArray(color_list2);

		// Display the contents of the array
		System.out.println("Printing elements of color_list2:");
		for (String color : color_list2) {
			System.out.println("Color = " + color);
		}
	}
}
```

خروجی(output):

```
Size of the color_list: 5
Color = White
Color = Black
Color = Red
Color = White
Color = Yellow
Printing elements of color_list2:
Color = White
```

```
Color = Black
Color = Red
Color = White
Color = Yellow
```

مثال از متد **clone()** در **ArrayList** :

Exampl19:

```
package javalike;

import java.util.*;

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

            // ArrayList with Capacity 4
            ArrayList<String> StudentList = new ArrayList<String>(4);
            Object cloneList;
            // Added 4 elements
            StudentList.add("David");
            StudentList.add("Tom");
            StudentList.add("Rohit");
            StudentList.add("Paul");

            System.out.println("Elements in StudentList are: ");
            System.out.println(StudentList);

            cloneList = StudentList.clone();
            System.out.println("Elements in cloneList are:");
            System.out.println(cloneList);
      }
}
```

خروجی(output):

Core Java

آموزش ساده و آسان جاوا

```
Elements in StudentList are:
[David, Tom, Rohit, Paul]
Elements in cloneList are:
[David, Tom, Rohit, Paul]
```

امیدوارم با بیانی ساده توانسته باشم `ArrayList` را آموزش بدم. این کتاب یکی از کامل ترین آموزش فارسی در زمینه `ArrayList` می باشد.

پیروز و موفق باشید

Core Java

آموزش ساده و آسان جاوا

سایت آموزش زبان جاوا به زبان ساده،آسان و شیرین!!!

www.JAVAPro.ir

آموزش جاوا SE را با تجربه شخصی و به زبان خودمونی یاد بگیرید!!!!!

بازدید از کانال

بازدید از سایت

هر روز مفاهیم و مثال های جدید به سایت اضافه می شود برای اطلاع از مطالب جدید روی سایت عضو کانال شوید.

دخل و تصرف ، ویرایش و کپی زدن تمامی آموزش های جاوالایک به دور از اخلاق حرفه ای ست و حرام می باشد.