

Core Java

آموزش ساده و آسان جاوا

به نام خدا

تقدیرم به هموطنان عزیزم

جاوا را با لذت یاد بگیرید!

Core Java

آموزش ساده و آسان جاوا

آموزش زبان برنامه نویسی جاوا

در جاوا Collections

موضوع: کلاس LinkedList

جلسه: سوم

نویسنده: رحمان زارعی

جاوا را ساده، آسان و شیرین بنوشید!!!



این جلسه آموزشی رایگان است، فروش و ویرایش آن ممنوع و حرام می باشد. اما این کتاب را می توانید همین جور که هست در سایت و شبکه اجتماعی خود به اشتراک بگذارید.

Core Java

آموزش ساده و آسان جاوا

سلام. امیدوارم خوب و شاد باشی ، امروز قصد داریم کلاس LinkedList یکی از کلاس های موجود در فریم ورک Collection در جاوا را بررسی کنیم. در فارسی به LinkedList ، لیست پیوندی می گویند. در درس ساختمان داده ها شما با نحوه پیاده سازی الگوریتم LinkedList (لیست پیوندی) آشنا می شوید. اما ما اصلا کار به این الگوریتم ها نداریم! تنها میخوایم کاربرد LinkedList (لیست پیوندی) مثل ArrayList در جاوا رو بررسی کنیم.

کلاس LinkedList در جاوا

کلاس LinkedList از یک لیست پیوندی دو طرفه (doubly linked list) برای ذخیره عناصر استفاده می کند. کلاس LinkedList کلاس AbstractList را به ارث برده و دو اینترفیس List و Deque را implements کرده است، بنابراین به ویژگی ها و رفتارهای آنها دسترسی دارد.

یک LinkedList (لیست پیوندی) می تواند عناصر تکراری داشته باشد.

دستکاری یک LinkedList (لیست پیوندی) به سرعت انجام می شود.

یک LinkedList می تواند از list ، stack یا queue استفاده کند.

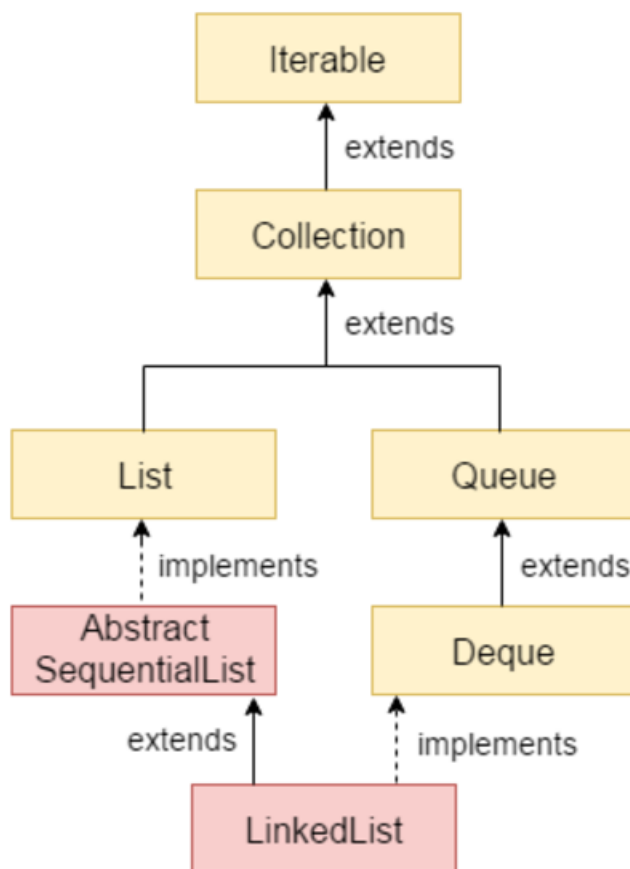
- در فارسی به queue صف ، به list لیست و به stack هم پشته می گویند. 😊

Core Java

آموزش ساده و آسان جاوا

سلسله مراتب کلاس LinkedList در جاوا

در نمودار تصویر (۱) نشان داده ایم که کلاس LinkedList کلاس AbstractList را extends کرده و دو اینترفیس List و Deque را implements کرده است.



تصویر (۱)

Core Java

آموزش ساده و آسان جاوا

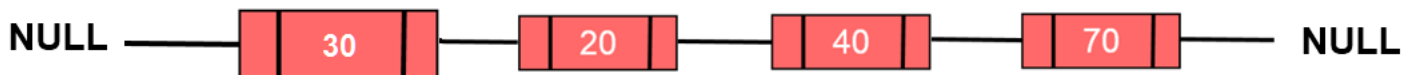
لیست پیوندی دو طرفه (Doubly Linked List)

در نمونه لیست پیوندی دو طرفه (doubly linked list)، ما می توانیم عناصر خود را از هر طرف لیست اضافه یا حذف کنیم. در تصویر (۲) نمونه ای از یک لیست پیوندی دو طرفه را مشاهده می کنید:



تصویر (۲) - لیست پیوندی دو طرفه

- منظور از این که می توانیم از هر طرف لیست، یک عنصر را اضافه یا حذف کنیم این است که مثلاً می توانیم عنصر خود را از سمت راست یا چپ لیست حذف یا به آن اضافه کنیم.
- مثلاً قصد داریم عنصر ۳۰ را به ابتدای لیست پیوندی دو طرفه موجود در تصویر (۲) اضافه کنیم، نتیجه در تصویر (۳) مشاهده کنید:



تصویر (۳)

- حالا طریقه اضافه یا حذف کردن یک عنصر به لیست پیوندی دو طرفه رو جلوتر یاد خواهیم گرفت.
- برای استفاده و تعریف کلاس LinkedList در برنامه خود باید پکیج زیر را بالای سر کلاس خود import کنید:

Core Java

آموزش ساده و آسان جاوا

```
import java.util.LinkedList;
```

سازنده های کلاس LinkedList

توضیح	Constructor (سازنده)	شماره
برای ساختن یک لیست پیوندی (LinkedList) خالی استفاده می شود.	LinkedList()	۱
برای ایجاد یک لیست پیوندی (LinkedList) که عناصر درون لیست از عناصر مجموعه C تشکیل می شود. همان طور که می دانید ما در فارسی به Collection مجموعه یا کلکسیون می گوئیم. مجموعه C می تواند شامل اشیای انواع کلاس های موجود در فریم ورک Collection مثل ArrayList و... شود.	LinkedList(Collection c)	۲

متدهای کلاس LinkedList (لیست پیوندی)

Core Java

آموزش ساده و آسان جاوا

توضیح	متد	شماره
برای درج (اضاف کردن) یک عنصر مشخص در خانه index ام لیست پیوندی کاربرد دارد.	<code>void add(int index, Object element)</code>	۱
عنصر 0 را ابتدای لیست پیوندی درج می کند. منظور از ابتدای لیست همان خانه اول لیست می باشد.	<code>void addFirst(Object o)</code>	۲
عنصر 0 را انتهای لیست پیوندی درج می کند.	<code>void addLast(Object o)</code>	۳
تعداد عناصر موجود در لیست پیوندی را برمی گرداند.	<code>int size()</code>	۴
عنصر 0 را به انتهای لیست پیوندی وصل می کند.	<code>boolean add(Object o)</code>	۵
اگر عنصر 0 در لیست پیوندی ما وجود داشت مقدار <code>true</code> برمی گرداند در غیر این صورت مقدار <code>false</code> برمی گرداند. معمولا برای پیدا کردن یک عنصر در لیست پیوندی استفاده می شود.	<code>boolean contains(Object o)</code>	۶
برای حذف اولین عنصر 0 موجود در لیست پیوندی استفاده می شود. برای مثال فرض کنید یک لیست پیوندی به صورت زیر داشته باشیم: [e, a, b, c, a, d] در لیست پیوندی بالا همان طور که می بینید، دو عنصر a وجود دارد، حال اگر بخواهیم از طریق متد <code>remove</code> عنصر a را حذف کنیم، اولین عنصر a موجود در لیست حذف می شود، نتیجه بعد از حذف عنصر a از طریق این متد به صورت زیر است: [e, b, c, a, d] پس متد <code>remove(Object o)</code> همیشه اولین عنصر مشخص 0 را در یک لیست پیوندی حذف می کند.	<code>boolean remove(Object o)</code>	۷
اولین عنصر لیست پیوندی را برمی گرداند.	<code>Object getFirst()</code>	۸

Core Java

آموزش ساده و آسان جاوا

<p>آخرین عنصر لیست پیوندی را برمی گرداند.</p>	<p>Object getLast()</p>	<p>۹</p>
<p>ایندکس یا شماره خانه اولین عنصر مشخص 0 در لیست پیوندی را برمی گرداند. در صورت عدم وجود عنصر مورد نظر در لیست پیوندی مقدار -1 را برمی گرداند.</p> <p>منظور اولین عنصر این است که اگر چند عنصر 0 تکراری در لیست پیوندی داشته باشیم، اولین 0 را انتخاب کرده و شماره خانه آن را برمی گرداند.</p> <p>برای مثال فرض کنید یک لیست پیوندی به صورت زیر داشته باشیم:</p> <p>[e, a, b, c, a, d]</p> <p>در این لیست پیوندی قصد داریم ایندکس یا شماره خانه اولین عنصر a را به دست آوریم، همان طور که مشاهده می کنید در این لیست پیوندی ما دو عنصر مشابه a وجود دارد. اگر عنصر a را به عنوان پارامتر به متد indexOf بدهیم، شماره خانه اولین عنصر a موجود در لیست پیوندی را به ما تحویل می دهد، که برابر 1 می باشد.</p>	<p>int indexOf(Object o)</p>	<p>10</p>
<p>این متد برعکس متد indexOf عمل می کند.</p> <p>این متد ایندکس یا شماره خانه آخرین عنصر 0 موجود در لیست پیوندی را برمی گرداند. در صورت عدم وجود عنصر مورد نظر در لیست پیوندی مقدار -1 را برمی گرداند.</p> <p>برای مثال فرض کنید یک لیست پیوندی به صورت زیر داشته باشیم:</p> <p>[e, a, b, c, a, d]</p> <p>در این لیست پیوندی قصد داریم ایندکس یا شماره خانه آخرین عنصر a را به دست آوریم، همان طور که مشاهده می کنید در این لیست پیوندی ما دو عنصر مشابه a وجود دارد. اگر عنصر a را به عنوان پارامتر به متد lastIndexOf بدهیم، شماره خانه</p>	<p>int lastIndexOf(Object o)</p>	<p>11</p>

Core Java

آموزش ساده و آسان جاوا

آخرین عنصر a موجود در لیست پیوندی را به ما تحویل می دهد،
که برابر ۴ می باشد.

- اگر توضیحات روشن نبود نگران نباشید چرا که حل مثال و آزمون و خطا کردن شما رو در یادگیری کمک خواهد کرد.

Example:

```
package linkedList;

/*
 *
 * https://t.me/javaliike
 *
 */
import java.util.*;

public class TestCollection7 {
    public static void main(String args[]) {

        LinkedList<String> al = new LinkedList<String>();
        al.add("borazjan");
        al.add("bushehr");
        al.add("Tehran");
        al.add("shiraz");
        al.addFirst("kermanshah");

        Iterator<String> itr = al.iterator();
        while (itr.hasNext()) {
            System.out.println(itr.next());
        }
    }
}
```

خروجی (output):

```
kermanshah
borazjan
bushehr
```

Core Java

آموزش ساده و آسان جاوا

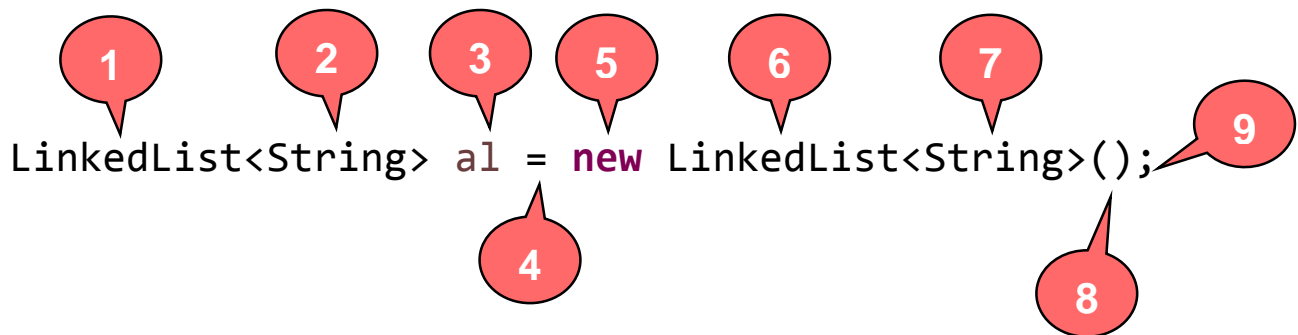
Tehran
shiraz

توضیحات:

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

برای ایجاد یک `LinkedList` (لیست پیوندی) نیز می توانیم شبیه `ArrayList` عمل کنیم.

گام های ایجاد یک `LinkedList` که عناصرش می توانند از نوع `String` باشند به صورت زیر است:



۱. ابتدا نام کلاس `LinkedList` را تایپ می کنیم.

۲. بعد نوع عناصری که لیست پیوندی ما می تواند ذخیره کند را بین دو تگ `<` `>` قرار می دهیم. ما در اینجا نوع عناصر لیست پیوندی را از نوع `String` تعیین کرده ایم.

۳. یک نام برای لیست پیوندی خود تعیین می کنیم.

۴. یک علامت مساوی می گذاریم.

۵. همان طور که می دانید برای شی سازی از هر کلاس باید از کلمه کلیدی `new` استفاده کنیم.

۶. دوباره نام کلاس `LinkedList` را تایپ می کنیم.

۷. دوباره نوع عناصری که لیست پیوندی ما می تواند ذخیره کند را بین دو تگ `<` `>` قرار می دهیم.

۸. پرانتز باز و بسته می کنیم

۹. یک علامت `;` هم انتهای پرانتز می گذاریم.

• مورد ۶ تا ۸ همان سازنده کلاس `LinkedList` می باشد که بعد از کلمه کلیدی `new` صدا زده شده است..

Core Java

آموزش ساده و آسان جاوا

به زبان ساده.....

برای ایجاد یک شی از نوع کلاس `LinkedList` به روش معمول از آن شی می سازیم:

```
LinkedList al = new LinkedList();
```

بعد نوع عناصر `LinkedList` را در بین دو تگ `<` `>` قرار داده و در کنار نام کلاس `LinkedList` در سمت چپ و راست مساوی قرار می دهیم.

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

- چرا باید نوع اشیا یا عناصری که یک لیست پیوندی می تواند ذخیره کند را هنگام تعریف لیست پیوندی باید مشخص کنیم؟
- این یکی از امکاناتی است که جاوا برای امن تر کردن برنامه در اختیار ما قرار داده است. شما می توانید یک لیست پیوندی را بدون تعیین نوع آن تعریف کنید. اما تصور کنید همه نوع داده ای می تواند بگیرد! مثلا هم `String` هم `int` و... حالا میشه شبیه آش رشته!!! که هر چیزی دستمون میاد میریزیم داخلش!! 😊 حالا هنگام دستکاری و کار کردن با لیست پیوندی سردرگم میشیم که با کدام نوع داده کار داریم همچنین ایجاد خطا هنگام اجرا برنامه بالا میره خیلی هم زرتنگ باشیم هر کدام از داده هایی که درون لیست پیوندی هست را باید `casting` کنیم. پس بهتره همون اول نوع عناصر لیست پیوندی خود را هنگام شی سازی تعیین کنیم که هنگام کار با لیست پیوندی اشیا یی جز از نوعی که برای لیست پیوندی تعریف کردیم نگیرد.

```
al.add("borazjan");
al.add("bushehr");
al.add("Tehran");
al.add("shiraz");
```

- با متد `add` هر عنصر را به انتهای لیست پیوندی خود اضافه می کنیم.

```
al.addFirst("kermanshah");
```

Core Java

آموزش ساده و آسان جاوا

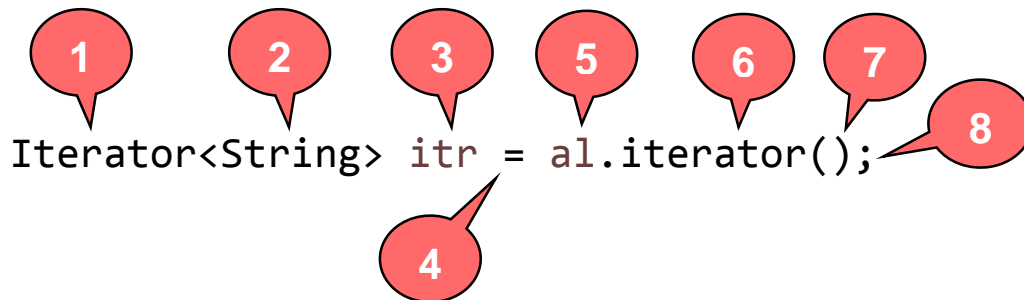
- با متد `addFirst` عنصر "kermanshah" را به ابتدای لیست پیوندی خود اضافه می کنیم.

```
Iterator<String> itr = al.iterator();
while (itr.hasNext()) {
    System.out.println(itr.next());
}
```

- کل این دستور برای پیمایش یک لیست پیوندی کاربرد دارد. در کل برای پیمایش هر `Collection` می توانیم به این شیوه عمل کنیم.

```
Iterator<String> itr = al.iterator();
```

- همان طور که می دانید برای پیمایش یک `Collection` که در اینجا یک لیست پیوندی است باید از اینترفیس `Iterator` شی بسازیم.



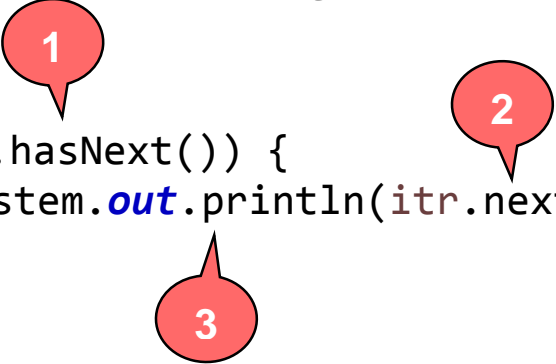
۱. ابتدا نام اینترفیس `Iterator` را تایپ می کنیم.
 ۲. نوع عناصری از `Collection` ما که در اینجا لیست پیوندی است را بین دو تگ `<>` قرار می دهیم. چون نوع عناصر لیست پیوندی ما در این مثال از نوع `String` است ، نوع شی ای که قراره از `Iterator` بسازیم از نوع `String` تعیین کرده ایم.
 ۳. یک نام به دلخواه انتخاب می کنیم.
 ۴. بعد علامت مساوی قرار می دهیم.
 ۵. شی که از نوع `Collection` مورد نظر ما می باشد را صدا می زنیم. در اینجا شی `al` از نوع کلاس `LinkedList` می باشد. به عبارتی دیگر `Collection` مورد نظر ما `LinkedList` می باشد.
 ۶. حالا با شی که از نوع کلاس `LinkedList` هست متد `iterator()` را صدا می زنیم.
- تا اینجا پیمایشگر خود خودمون رو برای پیمایش لیست پیوندی (`LinkedList`) ایجاد کرده ایم.

Core Java

آموزش ساده و آسان جاوا

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

- کل این حلقه برای پیمایش عناصر درون لیست پیوندی ما می باشد.



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

۱. شرط حلقه می‌گه تا هنگامی که عنصری وجود دارد دستورات درون بدنه حلقه رو اجرا کن. متد `hasNext` چک می کند که آیا عنصری بعد از عنصر فعلی وجود دارد یا خیر، یا به عبارت دیگر به انتهای لیست رسیدیم یا خیر.
۲. با هر بار صدا زده شدن متد `next` عنصر بعد از عنصر فعلی را به ما می دهد. مثلا بار اول که صدا زده شود عنصر اول لیست پیوندی را به ما می دهد، بار دوم که صدا زده شود عنصر بعد از عنصر اول یعنی عنصر دوم لیست پیوندی را به ما می دهد و بار سوم که صدا زده شود عنصر بعد از عنصر دوم یعنی عنصر سوم لیست پیوندی را به ما می دهد و.....
۳. و هر عنصر بعد از دریافت چاپ می شود.

Example:

```
package linkedList;
/*
 *
 https://t.me/javalikey
 */
import java.util.*;

class Book {
    int id;
    String name, author, publisher;
```

Core Java

آموزش ساده و آسان جاوا

```
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 LinkedListExample {
    public static void main(String[] args) {
        // Creating list of Books
        LinkedList<Book> list = new LinkedList<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
```

توضیحات:

Core Java

آموزش ساده و آسان جاوا

- ما در این مثال یک کلاس با نام Book ایجاد کرده و چند شی از آن ایجاد کرده و اشیای ایجاد شده از آن را در یک لیست پیوندی ذخیره کرده ایم.
- ما می توانیم نوع عناصر یک لیست پیوندی را از هر نوعی که دوست داریم و مورد نیازمون هست از جمله از نوع یک کلاس تعریف کنیم.

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

چون قرار در برنامه مون از لیست پیوندی استفاده کنیم این پکیج را import کرده ایم.

```
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;
    }
}
```

- یک کلاس با نام Book تعریف کرده دارای سری ویژگی می باشد.
- ویژگی های کلاس Book را از طریق سازنده آن مقدار دهی می کنیم.

```
public class LinkedListExample {
    public static void main(String[] args) {
        // Creating list of Books
        LinkedList<Book> list = new LinkedList<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);
    }
}
```

Core Java

آموزش ساده و آسان جاوا

```

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);
}
}
}

```

- حال در کلاس LinkedListExample از کلاس Book سه شی ایجاد کرده و آن اشیاء را به لیست پیوندی اضافه کرده ایم.

```
LinkedList<Book> list = new LinkedList<Book>();
```

- یک لیست پیوندی با نام list ایجاد کرده که نوع عناصرش از نوع کلاس Book می باشد.

```

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);

```

- سه شی از کلاس Book ایجاد کرده و پارامترهای سازنده آن را مقداردهی اولیه کرده ایم.

```

list.add(b1);
list.add(b2);
list.add(b3);

```

- اشیاء کلاس Book را به لیست پیوندی خود اضافه می کنیم.

```

for (Book b : list) {
    System.out.println(b.id + " " + b.name + " " + b.author + " "
        + b.publisher + " " + b.quantity);
}

```

- از طریق حلقه for-each به عناصر درون لیست پیوندی خود دسترسی پیدا کرده ایم.
- ما برخی از مفاهیمی که توضیح نمی دهیم را در جلسات قبل بررسی کردیم پس بهتره که از جلسه اول شروع به یادگیری کنید.

Core Java

آموزش ساده و آسان جاوا

```

1      2      3
for (Book b : list) {
    System.out.println(b.id + " " + b.name + " " +
b.author + " "
                        + b.publisher + " " + b.quantity);
}

```

۱. قصد داریم عناصر درون لیست پیوندی خود را چاپ کنیم. عناصر درون لیست پیوندی ما اشیایی از نوع کلاس Book می باشد. به همین خاطر در این بخش از حلقه یک شی از نوع کلاس Book تعریف کردیم. هر بار که لیست پیوندی پیمایش می شود عنصر دریافت شده درون شی **b** ریخته می شود.
۲. در این بخش از حلقه شی از نوع کلاس LinkedList خود قرار می دهیم.
۳. در این بخش از طریق شی **b** به ویژگی های هر عنصر که از نوع کلاس Book هست دسترسی پیدا می کنیم و آنها را چاپ می کنیم. در کل ویژگی های اشیای که از نوع کلاس Book هستند و در لیست پیوندی ذخیره شدند را چاپ می کنیم.

Example:

```

package linkedList;
import java.util.LinkedList;
public class LinkedListDemo {
    public static void main(String[] args) {
        LinkedList<String> myLinkedList = new LinkedList<String>();
        myLinkedList.addFirst("A");
        myLinkedList.add("B");
        myLinkedList.add("C");
        myLinkedList.add("D");
        myLinkedList.add(2, "X");//This will add C at index 2
        myLinkedList.addLast("Z");
        System.out.println("Original List before deleting elements");
        System.out.println(myLinkedList);
        myLinkedList.remove();
        myLinkedList.removeLast();
        myLinkedList.remove("C");
    }
}

```

Core Java

آموزش ساده و آسان جاوا

```

        System.out.println("Original List After deleting first and last object");
        System.out.println(myLinkedList);
        System.out.println("First object in linked list: "+
myLinkedList.getFirst());
        System.out.println("Last object in linked list: "+
myLinkedList.peekLast());
    }
}

```

خروجی (output):

```

Original List before deleting elements
[A, B, X, C, D, Z]
Original List After deleting first and last object
[B, X, D]
First object in linked list: B
Last object in linked list: D

```

Example:

```

package linkedList;

import java.util.LinkedList;

public class Test3 {

    public static void main(String[] args) {
        LinkedList<String> myLinkedList = new LinkedList<String>();
        myLinkedList.addFirst("A");
        myLinkedList.add("B");
        myLinkedList.add("C");
        myLinkedList.add("D");
        myLinkedList.add("A");
        myLinkedList.add("E");
        myLinkedList.add("F");

        System.out.println("return the first index A element: "
            + myLinkedList.indexOf("A"));
        System.out.println("return the last index A element: "
            + myLinkedList.lastIndexOf("A"));
    }
}

```

Core Java

آموزش ساده و آسان جاوا

```
        System.out.println("return the last index W element: "
            + myLinkedList.indexOf("W"));
    }
}
```

خروجی (output):

```
return the first index A element: 0
return the last index A element: 4
return the last index W element: -1
```

دوست من ، تمام سعی ام رو کردم که به شکل کاملاً ساده این `LinkedList` (لیست پیوندی) را در جاوا آموزش دهم. هنوز می توان مثال های فراوانی در این زمینه طرح کرد که از حوصله این جلسه آموزشی خارج است. پس بهتر در کنار این جلسه آموزشی مثال های فراوانی حل کنید و خودتون به چالش بکشید . من هم سعی می کنم در کنار جلسات آموزشی مثال های متنوعی در زمینه مباحث جاوا طرح کنم.

 پیروز و موفق باشی

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 شفقی 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.

دفل و تصرف ، ویرایش و کپی زدن تمامی آموزش های جاواالایک به دور از افلاق حرفه ای ست و حرام می باشد.