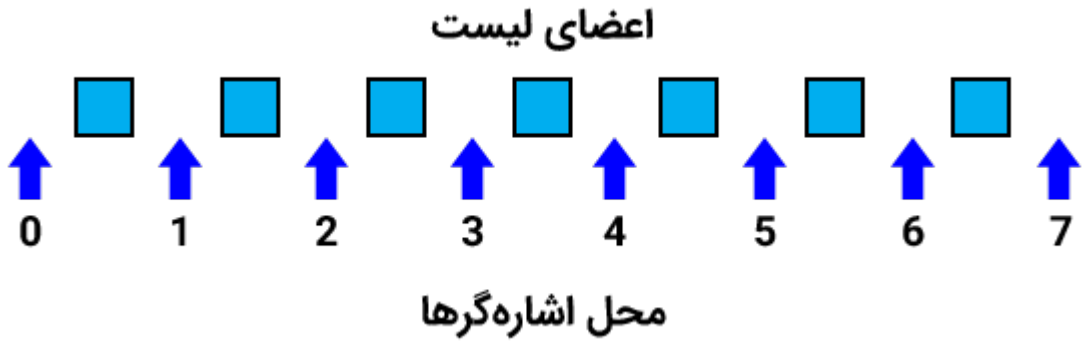
پیمایشگر

در این سوال، باید یک پیمایشگر برای container سوال قبل طراحی کنید بنابراین پیشنهاد می‌شود ابتدا سوال اول را حل کنید.

ابتدا فایل این برنامه را از source دانلود کنید. می‌توانید در پیاده‌سازی قسمت‌هایی از آن، از کدهای سوال قبل استفاده کنید.

کلاس‌هایی که قابلیت پیمایش توسط foreach را دارند در واقع واسط java.lang.Iterable را پیاده‌سازی می‌کنند. برای لیست‌ها، شی قابل پیمایش بازگردانده شده توسط متد iterator، نمونه‌ای از یک کلاس دیگر مثل GenericListIterator است که واسط java.util.ListIterator را پیاده‌سازی می‌کند، قابلیت هم‌زمان عقب و جلو رفتن را دارد، و در عین حال اطلاعاتی از لیست اصلی را برای انجام عملیات، درون خود دارد.

در صورت ابهام در پیاده‌سازی هر یک از متدهای زیر پیشنهاد می‌شود مستندات کلاس java.util.ListIterator را مطالعه کنید.



با توجه به نکات زیر پیمایشگر مشخص شده را پیاده‌سازی کنید:

- پیمایشگر در واقع یک اشاره‌گر مطابق شکل بالا است.

- هربار که متد iterator صدا زده می‌شود، یک پیمایشگر برای آن لیست ایجاد و بازگردانده می‌شود.
- ابتدا پیمایشگر در مکان صفر قرار دارد، فراخوانی متد next آن را یک عنصر جلو و فراخوانی متد previous آن را یک عنصر عقب می‌برد اما هیچ‌گاه نباید از محدوده لیست خارج شود. در شکل، مکان پیمایشگر فقط بین صفر تا هفت تغییر می‌کند.
- متدهای next و previous به هر ترتیبی می‌توانند صدا زده شوند.

```
1 | public boolean hasNext()
```

اگر امکان افزایش شماره‌ی پیمایشگر وجود داشته باشد، مقدار true را برمی‌گرداند.

```
1 | public T next()
```

مقدار پیمایشگر را یکی افزایش می‌دهد و با جابه‌جایی اشاره‌گر آن، داده‌ی عنصر را بر می‌گرداند برای مثال اگر مقدار پیمایشگر 3 باشد و این متد صدا زده‌شود، طبق شکل مقدار چهارمین عضو لیست برگردانده می‌شود.
اگر امکان افزایش وجود نداشته باشد، خطای NoSuchElementException پرتاب می‌شود.

```
1 | public boolean hasPrevious()
```

اگر امکان کاهش شماره‌ی پیمایشگر وجود داشته باشد، مقدار true بر می‌گرداند.

```
public T previous()
```

مقدار پیمایشگر را یکی کاهش می‌دهد و با جابه‌جایی اشاره‌گر آن، داده‌ی عنصر را بر می‌گرداند برای مثال اگر مقدار پیمایشگر 3 باشد و این متد صدا زده‌شود، طبق شکل مقدار سومین عضو لیست برگردانده می‌شود.
اگر امکان کاهش وجود نداشته باشد، خطای NoSuchElementException پرتاب می‌شود.

```
public int nextIndex()
```

اندیس عضوی را که در صورت فراخوانی next برگردانده می‌شود، برمی‌گرداند. اگر پیمایشگر در آخرین مکان خود باشد، آن‌گاه اندازه‌ی لیست را برمی‌گرداند.

```
public int previousIndex()
```

اندیس عضوی را که در صورت فراخوانی previous برگردانده می‌شود، برمی‌گرداند. اگر پیمایشگر در نخستین مکان خود باشد، مقدار -1 را برمی‌گرداند.

```
public void remove()
```

این متد می‌تواند فقط **یک** بار بعد از فراخوانی یکی از متدهای next یا previous فراخوانی شود. درصورت فراخوانی درست، عنصر برگردانده شده توسط next یا previous را از لیست حذف می‌کند و در غیر این صورت خطای IllegalStateException پرتاب می‌کند. (**راهنمایی: شما در این متد باید از enum Operation استفاده نمایید.**)

```java
public void set(T t) {
    throw new UnsupportedOperationException();
}
public void add(T t) {
    throw new UnsupportedOperationException();
}
```

لازم نیست که این دو متد را پیاده‌سازی کنید. 😊

در صورت رخ‌دادن هر یک از موارد زیر، **تمام** iterator های قبلی که برای آن لیست ساخته شده‌اند، نامعتبر می‌شوند. فراخوانی هر متد دلخواه روی یک شی ListIterator نامعتبر، سبب پرتاب خطای ConcurrentModificationException می‌شود.

۱. اضافه‌شدن عنصر به لیست

۲. حذف‌شدن عضوی از لیست: توجه کنید که صدا شدن تابع remove کافی نیست. در صورتی پیمایشگر نامعتبر می‌شود که حتما **تغییری در لیست** رخ دهد. برای مثال اگر در فراخوانی، خطایی پرتاب شود و عضوی حذف نشود، لیست تغییر نمی‌کند. **موارد دیگری هم وجود دارد!**

۳. فراخوانی متد clear

مثال

```java
GenericLinkedList<Integer> list = new GenericLinkedList<>();
for (int i = 0; i < 3; i++)

```

```
    list.add(i);
ListIterator<Integer> iterator = list.listIterator();
while (iterator.hasNext()) {
    System.out.print(iterator.next() + " ");
}
System.out.println(); // 0 1 2

for (Integer value : list)
    System.out.print(value + " ");
System.out.println(); // 0 1 2

ListIterator<Integer> iterator1 = list.listIterator();
ListIterator<Integer> iterator2 = list.listIterator();
list.remove(4); // doesn't corrupt iterator because list doesn't change
System.out.println(iterator1.next()); // 0
list.remove(0);

try {
    iterator1.next();
    System.out.println("Unable to catch ConcurrentModificationException");
} catch (ConcurrentModificationException e) {
    System.out.println("Catch ConcurrentModificationException");
}
try {
    iterator2.next();
    System.out.println("Unable to catch ConcurrentModificationException");
} catch (ConcurrentModificationException e) {
```

```
    System.out.println("Catch ConcurrentModificationException");
}
list.clear();
for (int i = 0; i < 10; i++) {
    list.add(i + 1);
}
list.set(1, 12); // doesn't corrupt list because pointers valid after set
iterator = list.listIterator();
while (iterator.hasNext()) {
    Integer data = iterator.next();
    if (data % 2 == 0)
        iterator.remove();
}

for (Integer value : list)
    System.out.print(value + " ");
System.out.println(); // 1 3 5 7 9
```

آن چه که باید آپلود کنید

آن‌چه که باید آپلود کنید یک فایل zip است که وقتی آن را باز می‌کنیم، در آن **فقط** فایل GenericLinkedList.java باشد.