**Cursor(usage) and there types in java with differences.**

**Cursor:**

cursor is used to iterate over collection items. means if we want to get objects one by one from the collection then we should go for cursor.it will retrieve object one by one.Let's have a look of the types of cursors.

There are 3 types of cursors available in java. They are:

* **Enumeration**
* **Iterator**
* **ListIterator**

**Enumeration:**

1. By using Enumeration cursor we can access object one by one ,but the limitation with the Enumeration is that it works only for legacy collection like Vector.
2. we can get an **Enumeration object**, we can use **elements()** method of any **legacy collection** class.

Vector v=**new** Vector();

**for**(**int** i=0;i<=10;i++)

{

          v.addElement(i);

}

System.out.println(v);//[0, 1, 2, 3, 4, 5, 6, 7,8, 9, 10]

Enumeration e=v.elements();

**while**(e.hasMoreElements())

{

          Integer i=(Integer)e.nextElement();

**if**(i%2!=0)

                   System.out.println(i);//1 3 5 7 9

}

**Iterator:**

1. By using **Iterator** we can access object one by one from any collection object.we can say that it is universal cursor.
2. we can iterate through collection items only in **Forward**direction.

ArrayList a=**new** ArrayList();

**for**(**int** i=0;i<=10;i++)

{

a.add(i);

}

System.out.println(a);//[0, 1, 2, 3, 4, 5, 6, 7,8, 9, 10]

Iterator itr=a.iterator();

**while**(itr.hasNext())

{

          Integer i=(Integer)itr.next();

          System.out.println(i+", ");//0, 1, 2, 3, 4, 5, 6, 7,8, 9, 10

}

**ListIterator:**

1. **ListIterator** is the child interface of Iterator.
2. By using listIterator() we can move either to the **forward direction** (or) to the **backward direction** that is it is a **bi-directional cursor**.
3. While iterating by **listIterator()** we can perform replacement and addition of new objects in addition to read and remove operations.

LinkedList l=**new** LinkedList();

l.add("CodeHungry");

l.add("code");

l.add("Hungry");

ListIterator itr=l.listIterator();

**while**(itr.hasNext())

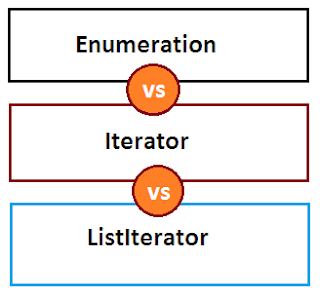
{

String s=(String)itr.next();

System.out.println(s +" "); //CodeHungry code Hungry

}

Now we will the differences among them. here i have given differences in form of tabular data.

[](http://4.bp.blogspot.com/-NcZi9ElkI6M/Wj5eSZcSTWI/AAAAAAAAYoo/-lLVRCZ29AIduQKK9DE-E12XrlqkFnU6QCK4BGAYYCw/s1600/enum.jpg)

|  |  |  |  |
| --- | --- | --- | --- |
| Property | Enumeration | Iterator | ListIterator |
| **Applicable For** | Works with Only Legacy Classes | Works with Any Collection Objects | Works with Only List Objects |
| **Movement** | Single Direction (Only  Forward). | Single Direction (Only Forward). | Its movement is Bi-Direction. |
| **How To Get** | By using elements() | By using iterator() | By using listIterator() of List(I) |
| **Accessibility** | Using Enumeration interface, we can only enumerate to read or get element/object from legacy collection. | Using Iterator interface, we can read as well as removecollection items, while iterating | Addition or replacement of new objects is possible alongside read and remove operation in ListIterator interface |
| **Methods** | 1. public boolean hasMoreElements(); 2. public Object nextElement(); | 1. public boolean hasNext(); 2. public object next();   3.public void remove(); | 1. public boolean hasNext(); 2. public Object next(); 3. public int nextIndex(); 4. public boolean hasPrevious(); 5. public Object previous(); 6. public int previousIndex(); 7. public void remove(); 8. public void set(Object new);   9.public void add(Object new); |
| **Is it legacy?** | Yes (1.0 Version) | No (1.2 Version) | No, (1.2 Version) |