2.0.4. Iteradoreak

Creative Commons lizentziapean publikatua

Iteradoreak?

- Zein da arazoa?
  - Bilduma bateko elementuen korritzea egiteko mekanismoa
- Nola egin?
  - Bildumako elementuei erreferentzia egingo dien objektua
  - Java-z Iterator<T> interfazea dago
  - Iterator<T> inplementatzen duen klase bakoitza objektuak era batean korritzeko diseinatuta dago

Iterator Interfazea

- Java-ko klase estandarren liburutegian (java.util) diseinatuta dago: bilduma batean objektu batetik bestera mugitzeko mekanismoa eskaintzen du

  – // prozesatuko den hurrengo objektua bueltatzen du
  public Object **next()**
  /* Iteradorea implementatzen duen klasearen diseinatzaileak erabaki beharko du nola eskainiko den hurrengo objektua. Hau da, bilduma korritzeko modua erabaki behar du */

  – // true bueltatuko du prozesatzeko elementu gehiago baleude
  public boolean **hasNext()**

  – // iteradoreak bueltatuko duen azken elementua kenduko du
  // bildumatik (eragiketa hau aukerazkoa da)
  public void **remove()**

Nola erabiltzen da iteradorea?

```
// demagun nireSet ArraySet<Person> objektu bat dela
// (pertsonen multzoa)

    Iterator<Person> erakusleNireSet = nireSet.iterator();
    Person p;
    while (erakusleNireSet.hasNext()){
          p = erakusleNireSet.next();
          System.out.println(p.toString());
    }
```

Nola erabiltzen da iteradorea?

```
public int bikoteKopurua (LinkedSet<Integer> nireMultzoa)
{
   Iterator<Integer> iteradore = nireMultzoa.iterator();
   int bikKop = 0;
   while (iteradore.hasNext() ) {
        if (iteradore.next() % 2 == 0 )
                bikKop++;
   }
   return bikKop;
}
```

iterator() metodoa inplementatu behar da

Non agertuko da iteradore metodo bat?

- Normala da iterator() metodo bat agertzea bere elementu guztiak aztertu behar den klase bakoitzean

- Iterator() metodoaren inplementazioak objektu iteradorea bueltatuko du, DMAko objektuen euskarria den egitura korritu ahal izateko

  - Array-ekin inplementatutako bildumetan:

```
public Iterator<T> iterator( ) {
        return new ArrayIterator<T> (taula, luzera);
}
```

  - LinearNode<T> adabegiekin inplementatutako bildumetan:

```
public Iterator<T> iterator( ) {
        return new LinkedIterator<T> (contents, count);
}
```

- ArrayIterator<T> eta LinkedIterator<T> klaseak inplementatu behar dira

Array-en iteradorea

```
import java.util.*;

public class ArrayIterator<T> implements Iterator<T> {

        private int count;   // bildumako elementu-kopurua
        private int current; // korritzearen uneko posizioa
        private T[] items;

        // Emandako bilduma korritzeko iteradorea hasieratzen du
        public ArrayIterator (T[] collection, int size) {
            items = collection;
            count = size;
            current = 0;
        }

        // true bueltatuko du bildumak elementu bat gehiago baldin badu
        // (oraindik ez izan bisitatua)
        public boolean hasNext(){
            return (current < count);
        }

        // Bildumako hurrengo elementua bueltatzen du, korritzearen arabera
        public T next(){
            current ++;
            return items[current-1];
        }
}
```

LinearNode<T> nodoetarako iteradorea zerrenda estekatuetarako (Lewis & Chase 2010)

```
public class LinkedIterator<T> implements Iterator<T> {

	//private int count; // liburuan agertzen da baina ez da beharrezkoa
	private LinearNode<T> current;  // uneko posizioa

	 // Emandako bilduma korritzeko iteradorea hasieratzen du
	public LinkedIterator (LinearNode<T> collection, int size){
		current = collection;
		//count = size; Ez da erabiltzen. Liburua jarraitzearren utzi dugu
	}

	// true bueltatuko du bildumak elementu bat gehiago baldin badu, oraindik bisitatugabea
	public boolean hasNext(){
		return ( current != null);
	}

	// Bildumako hurrengo elementua bueltatuko du, korritzearen arabera
	public T next(){
		if (! hasNext())
		    throw new NoSuchElementException();
		T result = current.getElem();
		current = current.getNext();
		return result;
	}

	// Ezabaketarako eragiketa ez dago inplementatuta
	public void remove() throws UnsupportedOperationException {
		throw new  UnsupportedOperationException();
	}
}
```

Matrizeak aztertzeko iteradorearen adibidea

```
import java.util.Iterator;


public class Matrize<T> {

   private T[][] m;
   private static int N = 20;
   private static int M = 50;

   public Matrize() {
      m = (T[][]) new Object[N][M];
   }

   public Iterator<T> NireIteradorea() {
      return new MatrizeIteradore<T>(m, N, M);
   }

}
```

Matrizeak aztertzeko iteradorearen adibidea

```
import java.util.Iterator;
public class MatrizeIteradore<T> implements Iterator<T> {

   T[][] itMat;
   int i, j;
   int lerroMax, zutMax;

   public MatrizeIteradore(T[][] mat, int zutKop, int lerroKop) {
         itMat = mat;
         lerroMax = lerroKop;
         zutMax = zutKop;
         i = 0;
         j = 0;
   }
   public boolean hasNext() {
      if ((i < lerroMax) && (j < zutMax)) return true;
      else return false;
   }
   public T next() {
      T temp = itMat[i][j];
      j++;
      if (j >= zutMax) {
         i++;
         j = 0;
       }
      return temp;
   }
   public void remove(){} // egin gabe!!!!!
}
```

Beste aukera bat: klase pribatu bat iteradorea emateko

```
public class nireArrayList<T> {

   private T[] taula;
   private final int OINARRIZKO_TAMAINA = 100;
   private int lehenLibrea;

   public nireArrayList() { // eraikitzailea...

   public Iterator<T> it(){ return new It(); }

   private class It implements Iterator<T>{
         int i = 0; // Hasieraketa

         public T next(){
                  if (i < lehenLibrea) { i++; return taula[i - 1]; }
                  else return null;
         }

         public boolean hasNext(){
           if (i < lehenLibrea) { return true;} else return false;
         }

         public void remove(){} // inplementatu barik
   }
```

Beste aukera bat: klase pribatu bat iteradorea emateko

```
public class nireList<T> {

private Link<T> first;

public Iterator iterator() { return new ListIterator(); }
```

```
private class ListIterator implements Iterator<T> {

   private Link<T> current = first; // Hasieraketa

   public boolean hasNext() { return current != null; }

   public void remove() { throw new UnsupportedOperationException(); }

   public Tnext() {
      if (!hasNext()) throw new NoSuchElementException();
      T item = current.data;
      current = current.next;
      return item;
    }
} // private class
```