PRAKTIKUM 20
DOUBLE LINKED LIST : CLASS LINKEDLIST DI COLLECTION

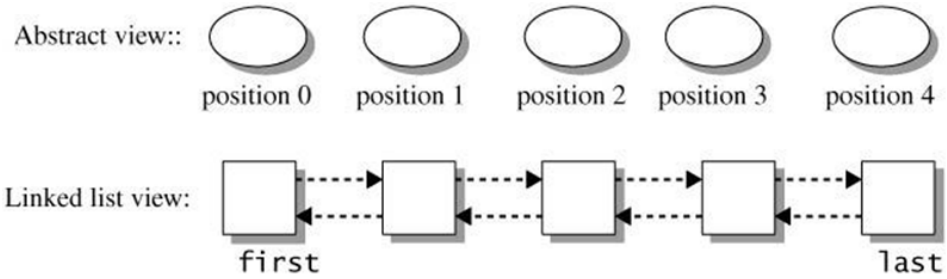
A. TUJUAN PEMBELAJARAN

1. Memahami konsep Class LinkedList di Collection
2. Memahami penggunaan method-method pada Class LinkedList.

B. DASAR TEORI

B Class Linked List pada Collection

Java telah menyediakan class LinkedList yang menerapkan Double Linked List.



Gambar 20.1 Konsep LinkedList

Class LinkedList mempunyai default konstruktor yang membuat linked list kosong. Method toString() mengembalikan string yang merepresentasikan isi list berdasarkan urutan masuk (sequence), dipisahkan dengan tanda koma dan di awali dan diakhiri dengan […].

Contoh penggunaan LinkedList :

1. Menggunakan default konstruktor untuk membuat list kosong

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

2. Misal list berisi "Red", "Blue", "Green". Tampilkan jumlah elemen yang tersimpan di list dan lakukan pengecekan apakah ada elemen ‘White’ pada list.

```
aList.add("Red");
aList.add("Blue");
```

```
aList.add("Green");
System.out.println("Size = " +  aList.size());
System.out.println("List contains the string 'White' is " +
                                    aList.contains("White");
```

Output:

```
[Red, Blue, Green]
Size = 3
List contains the string 'White' is false
```

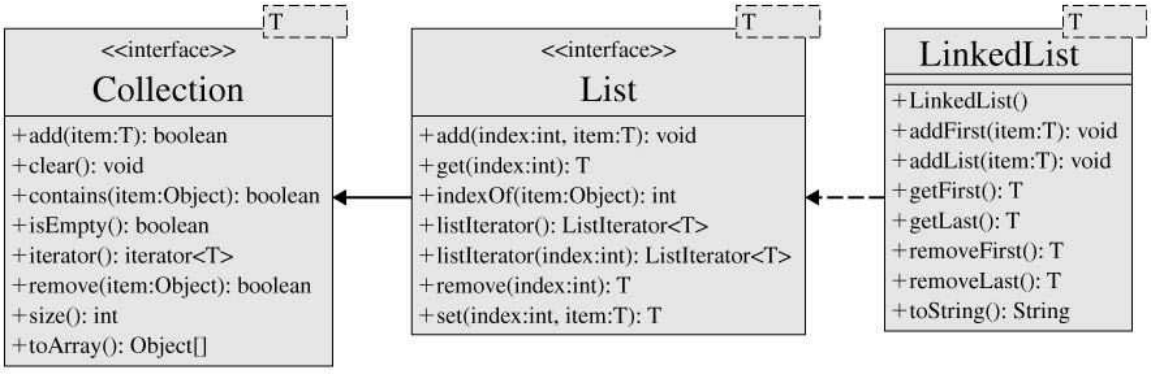
3. Hapuslah elemen ‘Blue’

```
aList.remove(“Biru”);
System.out.println("Size = " +  aList.size());
```

Output :

```
[Red, Green]
Size = 2
```

Class LinkedList adalah class yang mengimplementasikan interface List. Method-method yang terdapat pada Class LinkedList adalah sebagai berikut:



Gambar 20.2 Class LinkedList di Collection

C. TUGAS PENDAHULUAN

Buatlah review mengenai :

- Class LinkedList di Collectioon.
- Berikan 3 contoh method yang terdapat pada LinkedList dan jelaskan kegunaan dari method tersebut ! Berikan contoh penggunaan method tersebut!

D. PERCOBAAN

Percobaan 1 : Membuat objek LinkedList dan menambahkan elemen pada LinkedList

```
public class SimpleJavaLinkedListExample {

  public static void main(String[] args) {

    //create LinkedList object
    LinkedList lList = new LinkedList();

    //add elements to LinkedList
    lList.add("1");
    lList.add("2");
    lList.add("3");
    lList.add("4");
    lList.add("5");

    System.out.println("LinkedList contains : " + lList);
  }
}
```

Percobaan 2 : Mencari elemen pada LinkedList

```
import java.util.LinkedList;

public class SearchElementLinkedListExample {

  public static void main(String[] args) {
    //create LinkedList object
    LinkedList lList = new LinkedList();

    //add elements to LinkedList
    lList.add("1");
    lList.add("2");
    lList.add("3");
    lList.add("4");
    lList.add("5");
    lList.add("2");

     int index = lList.indexOf("2");
     if(index != -1)
     {
       System.out.println("First occurrence of 2 in LinkedList is at index :
" + index);
     }
     else
     {
      System.out.println("LinkedList does not contain 2");
     }

     index = lList.lastIndexOf("2");
```

```
     if(index != -1)
     {
      System.out.println("Last occurrence of 2 in LinkedList is at index : "
+ index);
     }
     else
     {
      System.out.println("LinkedList does not contain 2");
     }
  }
}
```

Percobaan 3 : Mengubah elemen yang lama dengan elemen yang baru.

```
import java.util.LinkedList;

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

    //create LinkedList object
    LinkedList lList = new LinkedList();

    //add elements to LinkedList
    lList.add("1");
    lList.add("2");
    lList.add("3");
    lList.add("4");
    lList.add("5");

    System.out.println("LinkedList contains : " + lList);

    lList.set(3, "Replaced");
    System.out.println("After replacing 4, LinkedList contains : " + lList);
  }
}
```

Percobaan 4 : Meghapus elemen dari LinkedList

```
import java.util.LinkedList;

public class RemoveElementLinkedListExample {

  public static void main(String[] args) {

    //create LinkedList object
    LinkedList lList = new LinkedList();

    //add elements to LinkedList
    lList.add("1");
    lList.add("2");
    lList.add("3");
```

```
    lList.add("4");
    lList.add("5");

    System.out.println("LinkedList contains : " + lList);

     boolean isRemoved = lList.remove("2");
     System.out.println("Is 2 removed from LinkedList ? :" + isRemoved);
     System.out.println("LinkedList now contains : " + lList);

      Object obj = lList.remove(2);
      System.out.println(obj + " has been removed from LinkedList");
      System.out.println("LinkedList now contains : " + lList);
  }
}
```

Percobaan 5 : Menghapus elemen dengan range tertentu.

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

public class RemoveRangeElementsLinkedListExample {

  public static void main(String[] args) {

    //create LinkedList object
    LinkedList lList = new LinkedList();

    //add elements to LinkedList
    lList.add("1");
    lList.add("2");
    lList.add("3");
    lList.add("4");
    lList.add("5");

    System.out.println("LinkedList contains : " + lList);

    //remove elements from index 2(inclusive) to 5(exclusive)
    lList.subList(2,5).clear();

    System.out.println("Range of elements removed from LinkedList");
    System.out.println("LinkedList now contains : " + lList);
  }
}
```

Percobaan 6 : Menghapus elemen pertama dan terakhir pada LinkedList.

```
public class RemoveFirstLastElementsLinkedListExample {

  public static void main(String[] args) {

    //create LinkedList object
```

```
    LinkedList lList = new LinkedList();

    //add elements to LinkedList
    lList.add("1");
    lList.add("2");
    lList.add("3");
    lList.add("4");
    lList.add("5");

    System.out.println("LinkedList contains : " + lList);

     Object object = lList.removeFirst();
     System.out.println(object + " has been removed from the first index
                                                          of LinkedList");
     System.out.println("LinkedList now contains : " + lList);

      object = lList.removeLast();
      System.out.println(object + " has been removed from the last index
                                                          of LinkedList");
      System.out.println("LinkedList now contains : " + lList);
  }
}
```

Percobaan 7 : Menghapus semua elemen pada LinkedList

```
public class RemoveAllElementsLinkedListExample {

  public static void main(String[] args) {

    //create LinkedList object
    LinkedList lList = new LinkedList();

    //add elements to LinkedList
    lList.add("1");
    lList.add("2");
    lList.add("3");
    lList.add("4");
    lList.add("5");

    System.out.println("LinkedList contains : " + lList);

     lList.clear();
     System.out.println("LinkedList now contains : " + lList);
  }
}
```

Percobaan 8 : Melakukan iterasi pada LinkedList menggunakan ListIterator.

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

```
public class IterateLinkedListUsingListIteratorExample {

  public static void main(String[] args) {

    //create LinkedList object
    LinkedList lList = new LinkedList();

    //add elements to LinkedList
    lList.add("1");
    lList.add("2");
    lList.add("3");
    lList.add("4");
    lList.add("5");

    ListIterator itr = lList.listIterator();

    System.out.println("Iterating through elements of Java LinkedList using
    ListIterator in forward direction...");
    while(itr.hasNext())
    {
      System.out.println(itr.next());
    }

    System.out.println("Iterating through elements of Java LinkedList using
    ListIterator in reverse direction...");
    while(itr.hasPrevious())
      System.out.println(itr.previous());

  }
}
```

Percobaan 9 : Melakukan iterasi pada LinkedList menggunakan Iterator

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


public class IterateThroughLinkedListUsingIteratorExample {

  public static void main(String[] args) {

    //create LinkedList object
    LinkedList lList = new LinkedList();

    //add elements to LinkedList
    lList.add("1");
    lList.add("2");
    lList.add("3");
    lList.add("4");
    lList.add("5");

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

```
    System.out.println("Iterating through elements of Java LinkedList...");
    System.out.println("LinkedList contains : ");
    while(itr.hasNext())
    {
      System.out.println(itr.next());
    }

  }
}
```

Percobaan 10 : Menyisipkan elemen pada index tertentu

```
import java.util.LinkedList;

public class LinkedListInsertElementExample {

  public static void main(String[] args) {

    //create LinkedList object
    LinkedList lList = new LinkedList();

    //add elements to LinkedList
    lList.add("1");
    lList.add("2");
    lList.add("3");
    lList.add("4");
    lList.add("5");

    System.out.println("LinkedList contains : " + lList);

     lList.add(2, "2.5");
     System.out.println("After inserting 2.5, LinkedList contains : " +
lList);
  }
}
```

Percobaan 11 : Mengambil elemen pertama dan terakhir pada LinkedList

```
import java.util.LinkedList;

public class GetFirstAndLastElementLinkedListExample {

  public static void main(String[] args) {

    //create LinkedList object
    LinkedList lList = new LinkedList();

    //add elements to LinkedList
    lList.add("1");
```

```
    lList.add("2");
    lList.add("3");
    lList.add("4");
    lList.add("5");

     System.out.println("First element of LinkedList is : " +
lList.getFirst());

     System.out.println("Last element of LinkedList is : " +
lList.getLast());
  }
}
```

Percobaan 12 : Mengambil elemen pada index tertentu.

```
import java.util.LinkedList;

public class GetElementsLinkedListExample {

  public static void main(String[] args) {

    //create LinkedList object
    LinkedList lList = new LinkedList();

    //add elements to LinkedList
    lList.add("1");
    lList.add("2");
    lList.add("3");
    lList.add("4");
    lList.add("5");

    System.out.println("LinkedList contains : ");
    for(int i=0; i< lList.size(); i++)
    {
      System.out.println(lList.get(i));
    }
  }
}
```

Percobaan 13 : Mendapatkan subList pada LinkedList dengan index awal sampai index akhir-1

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

public class GetSubListLinkedListJavaExample {

  public static void main(String[] args) {

    //create LinkedList object
```

```
    LinkedList lList = new LinkedList();

    //add elements to LinkedList
    lList.add("1");
    lList.add("2");
    lList.add("3");
    lList.add("4");
    lList.add("5");

    System.out.println("LinkedList contains : " + lList);

    List lst = lList.subList(1,4);
    System.out.println("Sublist contains : " + lst);

     //remove element from sublist
     lst.remove(2);
     System.out.println("Sublist now contains : " + lst);
     System.out.println("Original LinkedList now contains : " + lList);
  }
}
```

Percobaan 14 : Mengubah LinkedList menjadi array

```
import java.util.LinkedList;

public class ObjectArrayFromElementsOfLinkedListExample {

  public static void main(String[] args) {

    //create LinkedList object
    LinkedList lList = new LinkedList();

    //add elements to LinkedList
    lList.add("1");
    lList.add("2");
    lList.add("3");
    lList.add("4");
    lList.add("5");

     Object[] objArray = lList.toArray();

     System.out.println("Object array created from Java LinkedList.");
     System.out.println("Object array contains : ");

     for(int i=0; i<objArray.length ; i++)
     {
       System.out.println(objArray[i]);
     }
  }
}
```

Percobaan 15 : Mengecek apakah elemen terdapat pada LinkedList

```
import java.util.LinkedList;

public class CheckElementLinkedListExample {

  public static void main(String[] args) {

    //create LinkedList object
    LinkedList lList = new LinkedList();

    //add elements to LinkedList
    lList.add("1");
    lList.add("2");
    lList.add("3");
    lList.add("4");
    lList.add("5");

     boolean blnElement = lList.contains("4");

     if(blnElement)
     {
       System.out.println("LinkedList contains 4");
     }
     else
     {
       System.out.println("LinkedList does not contain 4");
     }
  }
}
```

Percobaan 16 : Menambahkan elemen di awal dan akhir LinkedList

```
import java.util.LinkedList;

public class AddElementsAtStartEndLinkedListExample {

  public static void main(String[] args) {

    //create LinkedList object
    LinkedList lList = new LinkedList();

    //add elements to LinkedList
    lList.add("1");
    lList.add("2");
    lList.add("3");
    lList.add("4");
    lList.add("5");

    System.out.println("LinkedList contains : " + lList);

     lList.addFirst("0");
```

```
     System.out.println("After inserting 0 at beginning, LinkedList contains
:"
     + lList);

     lList.addLast("6");
    System.out.println("After appending 0 at end, LinkedList contains :" +
lList);
  }
}
```

E. LATIHAN

1. Buatlah sebuah object kosong LinkedList dengan tipe String, beri nama dengan **aList** tambahkan elemen Merah, Biru, Hijau.

```
Output   : [Merah, Biru, Hijau]
```

2. Tambahkan elemen lagi Hitam dan Biru, hitung jumlah elemen ”Biru” dan tampilkan posisi dari elemen ”Biru” pada aList

```
Output   : [Merah, Biru, Hijau, Hitam, Biru]
Biru = 2
Posisi di index 1,4.
```

3. Hapus warna Biru yang pertama.

```
Output   : [Merah, Hijau, Hitam, Biru]
```

4. Tampilkan elemen pada index ke 1 dan 3, selanjutnya hapus index ke 1.

```
Index ke-1     : Hijau
Index ke-3     : Biru
Output   : [Merah, Hitam, Biru]
```

5. Ubah pada index ke-0 dari merah menjadi ungu, selanjutnya tambahkan elemen baru pada index ke-1 dengan Coklat.

```
Output   : [Ungu, Hitam, Biru]
Output   : [Ungu, Coklat, Hitam, Biru]
```

6. Gunakan method addFirst() untuk menambahkan elemen di awal aList, elemen yang ditambahkan adalah Kuning, ulangi lagi dengan menambahkan Merah

```
Output   : [Merah, Kuning, Ungu, Coklat, Hitam, Biru]
```

7. Gunakan method addLast() untuk menambahkan elemen di akhir aList, elemen yang ditambahkan Hijau, ulangi lagi dengan menambahkan Oranye.

```
Output   : [Merah, Kuning, Ungu, Coklat, Hitam, Biru, Hijau, Oranye]
```

8. Tampilkan elemen yang pertama dan terakhir.

```
Elemen pertama       : Merah
Elemen terakhir      : Oranye
```

9. Selanjutnya hapus elemen yang pertama, dan elemen yang terakhir.

```
Output : [Kuning, Ungu, Coklat, Hitam, Biru, Hijau, Oranye]
Output : [Kuning, Ungu, Coklat, Hitam, Biru, Hijau]
```

10. Menggunakan iterasi, lakukan pembacaan mundur.

```
Output : [Hijau, Biru, Hitam, Coklat, Ungu, Kuning]
```

11. Urutkan elemen yang terdapat di aList.

12. Hapuslah elemen 1-3 pada LinkedList

```
Output : [Hijau, Ungu, Kuning]
```

13. Acaklah elemen yang terdapat di aList.

14. Ubahlah menjadi array.

F. LAPORAN RESMI

Kerjakan hasil percobaan(D) dan latihan(E) di atas dan tambahkan analisa.