Implementasi List Menggunakan Java

Yohanes Nugroho

Overview

- Materi ini membahas mengenai implementasi List di Java
- Peserta kuliah dianggap sudah mengerti konsep List di C

Konsep Reference

- Java tidak memiliki pointer namun memiliki reference
- Reference adalah penunjuk ke objek (seperti pointer)
  - Penunjuk/reference tipe dasar tidak ada

Contoh

- Contoh penggunaan reference:

```
Point a = new Point();
Point b = a; /*b menunjuk ke a*/
```

- Jika b diubah, a juga berubah

```
b.setX(5); /*a berubah*/
```

Elemen List

- Di C, elemen list didefinisikan dengan struct, di Java tidak ada struct, semua adalah kelas, jadi elemen List juga adalah kelas
- Suatu kelas boleh merefer dirinya sendiri (tidak perlu ada typedef seperti di C)

```
class ListElement {
   String info;
   ListElement next;
}
```

Deklarasi ElemenList (lengkap)

- Gunakan accessor set/get

```
class ListElement {
    private String info; private ListElement next;
    public ListElement(){ this.info=null;this.next=null;
      }

    public ListElement(String info, ListElement
     next){ this.info=info;this.next=next; }
    public void setInfo(String info){this.info = info; }
    public String getInfo(){ return info;}
    public void setNext(ListElement next){ this.next =
      next; }
    public ListElement getNext(){ return next;}
}
```

NULL

- Reference yang tidak menunjuk ke manapun dinyatakan sebagai null (huruf kecil, bukan NULL seperti di C)
- Reference tidak bisa diperlakukan sebagai boolean (tidak seperti C)

```
if (head) { /*aksi*/} //salah
if (head!=null) {/*aksi*/} //benar
```

Implementasi List

- List diimplementasikan sebagai kelas:
  - Property yang dimiliki adalah head dan atau tail

```
class SingleLinkedList {
   private ListElement head;
}
```

Konstruktor

- Konstruktor membuat list kosong
  - Biasanya mengisi head/tail dengan null
- Contoh:

```
public SingleLinkedList(){
   head=null;
}
```

Method lain

- Algoritma untuk method lain sama dengan C
- Contoh:

```
public void addFirst(String s){
   if (head==null){
      head = new ListElement();
      head.setInfo(s); head.setNext(null);
   } else {
      ListElement e = new ListElement();
      e.setInfo(s); e.setNext(head);
      head = e;
   }
}
```

`addFirst()` menggunakan konstruktor

- AddFirst dapat diimplementasikan menggunakan konstruktor

```
public void addFirst(String s){
      head = new ListElement(s, head);
   }
```

Dealokasi

- Dealokasi elemen tidak perlu dilakukan (tidak ada free)
- Contoh:

```
/*list pasti tidak kosong*/
public String delFirst(String s){
   String s = head.getInfo();
   head = head.getNext();
   return s;
}
```

Inner Class

- Kelas ListElement boleh ada di dalam kelas List, seperti ini:

```
class List {
   class ListElement {
      /*isi kelas list element*/
   }
  /*isi kelas List*/
}
```

Static Inner Class

- Dengan sintaks sebelumnya, kelas ListElement tidak bisa diakses tanpa instance kelas List

```
class List {
    static class ListElement { /*isi kelas list
      element*/ }
    /*isi kelas List*/
}
```

- Dengan keyword static, kelas ListElement memiliki sifat *static* bisa diakses dari luar kelas tanpa perlu ada instance dari kelas yang melingkupi:

```
SingleLinkedList.ListElement = new
  SingleLinkedList.ListElement();
```

List Rekursif

- List bisa diimplementasikan seperti pada bahasa fungsional
- List memiliki info (car) dan sisanya adalah list tail (cdr)

```
class List {
   private String car;
   private List cdr;
   /*accessor, java mengijinkan nama
     method sama dengan nama property*/
   public String car() { return car;}
   public List cdr() { return cdr;}
}
```

Konstruktor List Rekursif

```
class List {
/*…*/
/*properti dan method lain */
/*…*/
  public List(String car, List cdr)
  {
  this.car = car;
  this.cdr = cdr;
 }
}
```

Definisi List Kosong

- List kosong adalah list yang menunjuk ke null (tidak ada operasi yang bisa dilakukan):

```
List l = null; /*list kosong*/
l.print(); /*illegal*/
```

Algoritma Rekursif

- Menghitung jumlah elemen:

```
int countElement(){
   return 1 +
     ((cdr()==null)?0:cdr().countElement()
     );
}
```

- Print (traversal):

```
void print(){
   System.out.println(car());
   if (cdr()!=null) cdr().print();
}
```

Contoh Pemakaian List Rekursif

```
public static void main(String
  argv[]){
    List l = new List("Hello", new
  List("World", null));
    System.out.println("Count="+
     l.countElement());
    l.print();
}
```

Catatan Penutup

- Kelas internal sebaiknya digunakan dibanding kelas terpisah jika hanya ada satu jenis list (*coupling* lebih terlihat)
- Jika ada banyak jenis list dengan satu jenis elemen (misal single linked list dengan head, dan single linked list dengan head dan tail) maka kelas ListElement boleh dipisah