**TUGAS PRAKTIKUM**

*(diajukan untuk memenuhi salahsatu tugas mata kuliah AlproSD)*



Disusun oleh :

***Dede Iskandar***

***(1127050031)***

**JURUSAN TEKNIK INFORMATIKA**

**FAKULTAS SAINS DAN TEKNOLOGI**

**UNIVERSITAS ISLAM NEGERI SUNAN GUNUNG DJATI**

**BANDUNG**

**2013**

**Tugas Praktikum Algoritma & Struktur Data**

**Contoh 1**

class Link

{

public long dData;

public Link next;

public Link(long dd)

{

dData=dd;

}

public void dsplayLink()

{

System.out.print(dData+" ");

}

}

class sortedList

{

private Link first;

public sortedList()

{

first = null;

}

public boolean isEmpty()

{

return(first == null);

}

public void insert (long key)

{

Link newlink = new Link (key);

Link previous = null; // start pertama

Link current = first; // kondisi list

while(current != null && key > current.dData)

{

previous=current;

current=current.next;

}

if(previous==null)

{

first=newlink;

}

else

{

previous.next=newlink;

}

newlink.next=current;

}

public Link remove()

{

Link temp = first;

first = first.next;

return temp;

}

public void displayList()

{

System.out.print("List(first->last) : ");

Link current = first;

while(current != null)

{

current.dsplayLink();

current=current.next;

}

System.out.println(" ");

}

}

class SortedListApp

{

public static void main(String[]args)

{

sortedList sortlist=new sortedList();

sortlist.insert (2);

sortlist.insert (4);

sortlist.insert (1);

sortlist.displayList ();

sortlist.insert (7);

sortlist.insert (0);

sortlist.displayList ();

sortlist.remove();

sortlist.displayList ();

}

}

Program diatas terdiri dari tiga class, yaitu : class Link, class sortList, dan class sortedListApp. Cara kerja program ini yaitu pertama lihat class sortedList, “**first == null**” berarti nilai awalnya sama dengan nol. Lalu ” insert (long **key**)” nya bernilai **2**, nilai 2 ini terdapat pada data yang berada pada class sortedListApp. Selanjutnya “**previous bernilai nol**” dan **“current = first “** . Lakukan pengulangan dengan mengecek **current != null**hasilnya salah, karena current bernilai nol. Lalu masuk pada langkah selanjutnya **if(previous == null)** jawaban nya benar, maka **first=newlink** dan first sekarang bernilai **2**.

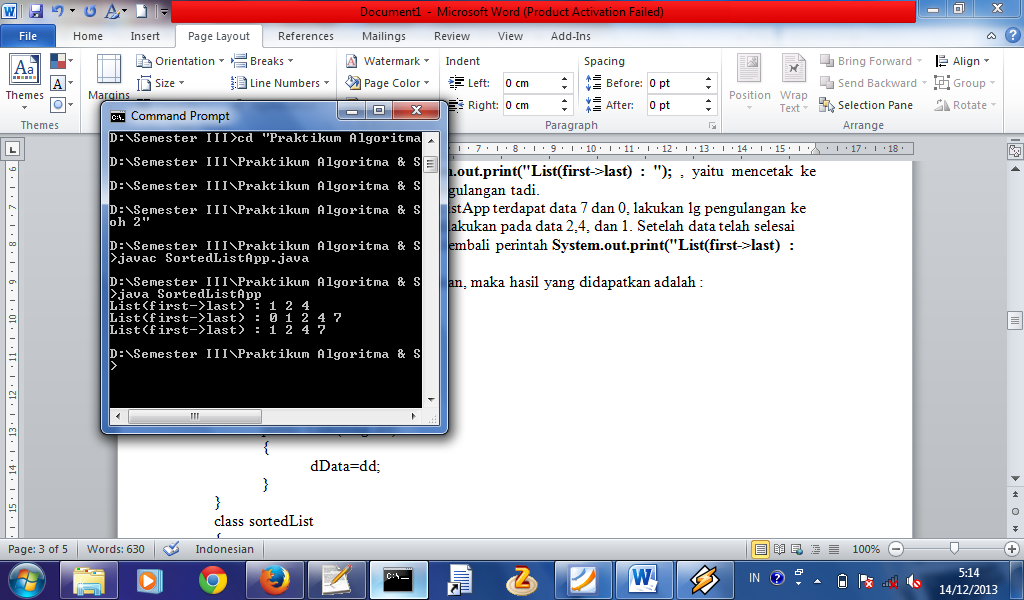
Lakukan lagi pengulangan. **First = newlink**, bernilai 2. Newlink (key) sekarang bernilai 4, nilai 4 tersebut didapat dari data yang ada pada class sortedListApp. Masuk pada pengulangan **current != null** jawabannya benar karena **current = 2**lanjut **key > current** jawabannya benar, karena **key = 4** dan **current = 2**, **4 > 2**. Langkah selanjutnya **previous = current**, previous menjadi 2 dan **current menjadi 4**.

Karena jawaban diatas benar, maka lakukan lagi pengulangan dengan **first=current=4, previous = 4,** dan **key = 1**. Nilai 1 tersebut merupakan data dari class sortedListApp. Kemudian**current != null** jawabannya benar karena current = 4 dan 4!=0, selanjutnya **key > current.dData, key = 1** dan  **current = 4** maka 1 > 4, jawabannya salah maka kerjakan **if(previous==null)** jawabannya salah maka kerjakan **elseprevious.next=newlink**, maka **previous = 1**.

Setelah itu kembali lihat class sortedListApp **sortlist.displayList ();**, pada bagian ini program akan mengerjakan **System.out.print("List(first->last) : ");**, yaitu mencetak ke layar data yang telah melakukan pengulangan tadi.

Lihat lagi pada class sortedListApp terdapat data 7 dan 0, lakukan lg pengulangan ke class sortedList seperti yang telah dilakukan pada data 2,4, dan 1. Setelah data telah selesai melakukan pengulanagan kerjakan kembali perintah **System.out.print("List(first->last) : ");**.

Program pun selesai dikerjakan, maka hasil yang didapatkan adalah :



**Contoh 2**

class Link

{

public long dData;

public Link next;

public Link(long dd)

{

dData=dd;

}

}

class sortedList

{

private Link first;

public sortedList()

{

first=null;

}

public sortedList(Link[]linkArr)

{

first=null;

for(int j=0; j<linkArr.length; j++)

{

insert(linkArr[j]);

}

}

public void insert (Link k)

{

Link previous = null; // start pertama

Link current = first; // kondisi list

while(current != null && k.dData>current.dData)

{

previous=current;

current=current.next;

}

if(previous==null)

first=k;

else

previous.next=k;

k.next=current;

}

public Link remove()

{

Link temp = first;

first=first.next;

return temp;

}

}

class ListInsertionSort

{

public static void main (String[]args)

{

int size=5;

Link[]linkArray= new Link[size];

for(int j=0;j<size;j++)

{

int n=(int)(java.lang.Math.random()\*99);

Link newlink = new Link(n);

linkArray[j]=newlink;

}

System.out.print("insorted Array :");

for (int j=0; j<size;j++)

{

System.out.print(linkArray[j].dData+" ");

}

System.out.println("");

sortedList sortlist=new sortedList(linkArray);

for(int j=0; j<size ;j++)

{

linkArray[j]=sortlist.remove();

}

System.out.print("Sorted Array :");

for (int j=0; j<size ;j++)

{

System.out.print(linkArray[j].dData+" ");

}

System.out.println("");

}

}

Program diatas terdiri dari tiga class, yaitu : class Link, class sortedList, dan class ListInserttionSort. Ini terlihat sedikit rumit untuk menjadi Insertion Sort. Program Link List terlalu banyak mengecek untuk class Link null, sebelum mengerjakan program yang harus ditemukan adalah **Link previous=null; Link current = first;**( jika Anda menerapkan doubly-linked ) .

Pada dasarnya , program ini menghapus setiap elemen dari daftar mulai dari yang pertama , samapai ke akhir. Kemudian , untuk setiap penghapusan , mulai dari awal melalui posisi program ini menarik dan menemukan di mana elemen akan cocok untuk si masukkan ( berarti elemen berikutnya adalah kurang dari elemen dihapus ) dari awal sampai elemen sebelum posisinya di penghapusan , dan menambahkannya di di sana.

Jadi misalnya, diberikan [ 65 , 88 , 76 , 75 , 65 ] , Anda mulai dengan 65. Elemen setelah 65 adalah 88, dan sampai elemen terahir. Sekarang ambil 65awal , dan iterate dari awal ke posisi sebelum 76. Anda harus memasukkan antara 65 awal dan 76 jadi [ 65 , 65 , 76 , 75 , 88 ]. Program ini akan melanjutkan proses ini sampai daftar menjadi [ 65 , 65 , 75 , 76 , 88 ].. Bedanya dengan LinkedLists adalah bahwa Anda hanya menambahkan Node pada indeks yang diinginkan daripada mengubah ukuran array . Jadi jika Anda telah menerapkan add ( indeks , elemen ) dan menghapus ( index ) metode untuk insertion sort.

Hasil yang diperoleh dari program di atas :

