Nama : Ahmad Muzakki

Kelas ; 2183D

Nim : 2130803095

Mata Kuliah : Struktur Data Dan Algoritma

Akun Github :

https://github.com/Nuvuss/SDA_2130803095_AhmadMuzakkii

Daftar Tertaut Sederhana (A Simple Linked List)

Contoh program pertama kami, linkList.java, menunjukkan daftar tertaut sederhana. Satu-satunya operasi yang diizinkan dalam versi daftar ini adalah

- Memasukkan item pada awal daftar
- Menghapus item pada awal daftar
- Iterasi melalui daftar untuk menampilkan isinya

Operasi ini cukup mudah dilakukan, jadi kita akan mulai dengan mereka. (Seperti yang akan kita lihat nanti, operasi ini juga semua yang Anda butuhkan untuk menggunakan daftar tertaut sebagai dasar untuk tumpukan.)

Sebelum kita sampai ke linkList lengkap.program java, kita akan melihat beberapa bagian penting dari Link dan Link Daftar Kelas.

The Link Class

Anda sudah melihat bagian data dari kelas Link. Berikut definisi kelas lengkap :

```
      class Link
         {
         public int iData;              // data item
         public double dData;           // data item
         public Link next;              // next link in list

----------//----------------------------------------------------------------------

      -
         public Link(int id, double dd) // constructor
            {
      iData = id;                  // initialize data
            dData = dd;                  // ('next' is automatically
            }                            //  set to null)

----------//----------------------------------------------------------------------
```

```
-
   public void displayLink()      // display ourself
      {
      System.out.print("{" + iData + ", " + dData + "} ");
      }

   }  // end class Link
```

Selain data, ada konstruktor dan metode, displayLink(), yang menampilkan data tautan dalam format {22, 33.9}. Puritan objek mungkin akan keberatan untuk menamai metode ini displayLink (), dengan alasan bahwa itu seharusnya hanya display (). Ini akan menjadi semangat polimorfisme, tetapi itu membuat daftar agak sulit untuk dipahami ketika Anda melihat pernyataan seperti

saat ini.tampilan(); (current.display();)

dan Anda lupa apakah current adalah objek tautan, objek LinkList, atau yang lainnya.

Konstruktor menginisialisasi data. Tidak perlu menginisialisasi bidang berikutnya, karena secara otomatis disetel ke null saat dibuat. (Meskipun dapat diatur ke null secara eksplisit, untuk kejelasan.) Nilai null berarti tidak mengacu pada apa pun, yang merupakan situasi sampai tautan terhubung ke tautan lain.

Kami telah membuat jenis penyimpanan bidang tautan (iData dan sebagainya) menjadi publik. Jika bersifat pribadi, kami perlu menyediakan metode publik untuk mengaksesnya, yang memerlukan kode tambahan, sehingga membuat daftar lebih lama dan lebih sulit dibaca. Idealnya, untuk keamanan kita mungkin ingin membatasi akses Link-object ke metode kelas LinkList. Namun, tanpa hubungan warisan antara kelas-kelas ini, itu sangat tidak nyaman. Kita dapat menggunakan penentu akses default (tanpa kata kunci) untuk memberikan akses Paket data (akses terbatas ke kelas di direktori yang sama) tetapi itu tidak berpengaruh dalam program demo ini, yang hanya menempati satu direktori. Penentu publik setidaknya memperjelas bahwa data ini tidak pribadi.

The LinkList Class

Kelas LinkList hanya berisi satu item data: referensi ke tautan pertama dalam daftar. Referensi ini disebut pertama. Ini adalah satu-satunya informasi permanen yang disimpan dalam daftar tentang lokasi tautan mana pun. Ia menemukan link lain dengan mengikuti rantai referensi dari pertama, menggunakan bidang berikutnya masing-masing link.

```
class LinkList
   {
```

```
            private Link first;             // ref to first link on
            list


----------//--------------------------------------------------------------------

            -
               public void LinkList()          // constructor
                  {
                  first = null;                // no items on list yet
                  }

----------//--------------------------------------------------------------------

            -
               public boolean isEmpty()        // true if list is empty
                  {
                  return (first==null);
                  }

----------//--------------------------------------------------------------------

            -
               ...                             // other methods go here

               }
```

Konstruktor untuk LinkList Menetapkan Pertama Ke null. Ini tidak benar-benar diperlukan karena seperti yang kami catat, referensi disetel ke null secara otomatis saat dibuat. Namun, konstruktor eksplisit memperjelas bahwa ini adalah bagaimana pertama kali dimulai.

Ketika pertama kali memiliki nilai null, kita tahu tidak ada item dalam daftar. Jika ada item, pertama akan berisi referensi ke yang pertama. Metode isEmpty() menggunakan fakta ini untuk menentukan apakah daftar kosong.

Metode insertFirst (). (The insertFirst() Method)

Metode insert First () dari LinkList menyisipkan tautan baru di awal daftar. Ini adalah tempat termudah untuk menyisipkan tautan, karena pertama sudah menunjuk ke tautan pertama. Untuk menyisipkan tautan baru, kita hanya perlu mengatur bidang berikutnya di tautan yang baru dibuat untuk menunjuk ke tautan pertama yang lama, lalu ubah terlebih dahulu sehingga mengarah ke tautan yang baru dibuat. Ini ditunjukkan pada gambar 5.5.

Di insert First () kita mulai dengan membuat tautan baru menggunakan data yang diteruskan sebagai argumen. Kemudian kami mengubah referensi tautan seperti yang baru saja kami catat.

```
                                // insert
at start of listpublic void
insertFirst(int id, double dd)
   {                            // make new link
   Link newLink = new Link(id, dd);
   newLink.next = first;        //
   newLink --> old firstfirst =
   newLink;                     // first
   --> newLink

   }
```

Panah - > di komentar dalam dua pernyataan terakhir berarti bahwa tautan (atau bidang pertama) terhubung ke tautan (hilir) berikutnya. (Dalam daftar tertaut ganda kita akan melihat koneksi upstream juga, dilambangkan dengan < -- arrows.) Bandingkan kedua pernyataan ini dengan gambar 5.5. Pastikan Anda memahami bagaimana pernyataan menyebabkan tautan diubah, seperti yang ditunjukkan pada gambar. Manipulasi referensi semacam ini adalah jantung dari algoritma daftar tertaut.

Metode deleteFirst ()

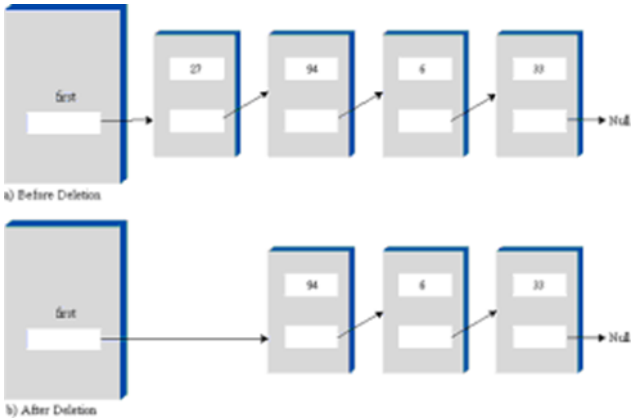
Metode delete First () adalah kebalikan dari insert First (). Ini memutus tautan pertama dengan mengubah rute terlebih dahulu untuk menunjuk ke tautan kedua. Tautan kedua ini ditemukan dengan melihat bidang berikutnya di tautan pertama.

```
public Link deleteFirst()       // delete first item
   {                            // (assumes
   list not empty) Link temp = first;      //
   save reference to link first = first.next;
                                // delete it:
   first-->old nextreturn temp;// return
   deleted link

   }
```

Gambar 5.5: memasukkan tautan baru

Pernyataan kedua adalah semua yang Anda butuhkan untuk menghapus tautan pertama dari daftar. Kami memilih untuk juga mengembalikan tautan, untuk kenyamanan pengguna daftar tertaut, jadi kami menyimpannya di temp sebelum menghapusnya, dan mengembalikan nilai temp. Gambar 5.6 menunjukkan bagaimana pertama dialihkan untuk menghapus objek.

Dalam C++ dan bahasa serupa, Anda perlu khawatir tentang menghapus tautan itu sendiri setelah terputus dari daftar. Itu ada di memori di suatu tempat, tetapi sekarang tidak ada yang merujuk padanya. Apa yang akan terjadi? Di Java, proses pengumpulan sampah akan menghancurkannya di beberapa titik di masa depan; itu bukan tanggung jawab Anda.



Gambar 5.6: menghapus tautan

Perhatikan bahwa metode delete First () mengasumsikan daftar tidak kosong. Sebelum memanggilnya, program Anda harus memverifikasi ini dengan metode is Empty ().

Metode displayList ()

Untuk menampilkan daftar, Anda mulai pada awalnya dan mengikuti rantai referensi dari link ke link. Arus variabel menunjuk ke (atau secara teknis mengacu pada) setiap tautan secara bergantian. Ini dimulai menunjuk ke pertama, yang memegang referensi ke link pertama. Pernyataan

```
current = current.next;
```

perubahan saat ini untuk menunjuk ke link berikutnya, karena itulah apa yang ada di bidang berikutnya di setiap link. Berikut adalah seluruh metode displayList() :

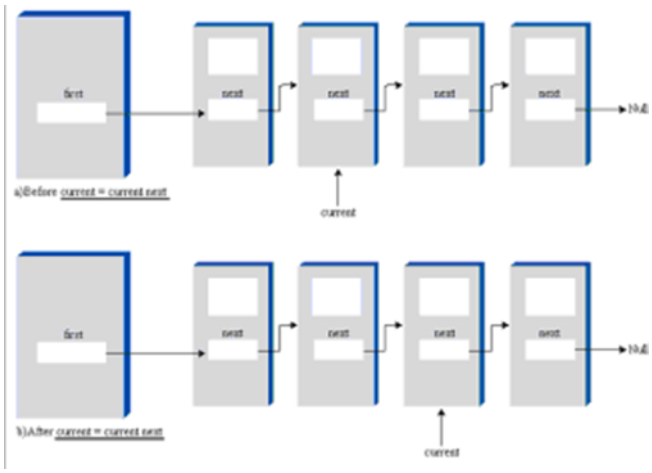
```
public void displayList()
   {
   System.out.print("List (first-->last): ");
   Link current = first;       // start at
   beginning of listwhile(current != null)
                               // until end
   of list,
      {
      current.displayLink();   //
      print data current =
      current.next;  // move to
      next link
      }
   System.out.println("");

   }
```

Akhir daftar ditunjukkan oleh bidang berikutnya di tautan terakhir yang menunjuk ke null daripada tautan lain. Bagaimana bidang ini bisa menjadi nol? Ini dimulai seperti itu ketika tautan dibuat dan tidak pernah diberi nilai lain karena selalu ada di akhir daftar. Sementara loop menggunakan kondisi ini untuk mengakhiri sendiri ketika mencapai akhir daftar. Gambar 5.7 menunjukkan bagaimana langkah-langkah saat ini di sepanjang Daftar.

Pada setiap link, metode displayList() memanggil metode displayLink () untuk menampilkan data di link.

LinkList.program java

Daftar 5.1 menunjukkan linkList lengkap.program java. Anda sudah melihat semua komponen kecuali main () rutin



Gambar 5.7: melangkah sepanjang Daftar

```
// linkList.java
// demonstrates linked list
// to run this program: C>java LinkListApp
////////////////////////////////////////////////
///////////////class Link
   {
   public int iData;              // data item (key)
   public double dData;           // data item
   public Link next;              // next link in list

----------//----------------------------------------------------------------------

-
   public Link(int id, double dd) // constructor
      {
iData = id;                  // initialize data
      dData = dd;                  // ('next' is automatically
      }                            //  set to null)

----------//----------------------------------------------------------------------

-
   public void displayLink()      // display ourself
      {
      System.out.print("{" + iData + ", " + dData + "} ");
      }
   }  // end class Link
```

```
        ////////////////////////////////////////////////

        ///////////////class LinkList
           {
           private Link first;            // ref to first link on
           list


----------//-------------------------------------------------------------------

           -
           public LinkList()              // constructor
              {
              first = null;               // no items on list yet
              }

----------//-------------------------------------------------------------------

           -
           public boolean isEmpty()       // true if list is empty
              {
              return (first==null);
              }

----------//-------------------------------------------------------------------

           -
                                          // insert
           at start of listpublic void
           insertFirst(int id, double dd)
              {                           // make new link
              Link newLink = new Link(id, dd);
              newLink.next = first;       //
              newLink --> old firstfirst =
              newLink;                    // first
              --> newLink
              }

----------//-------------------------------------------------------------------

           -


           public Link deleteFirst()      // delete first item
              {                           // (assumes
              list not empty)Link temp = first;    //
              save reference to link first =
              first.next;                 // delete
```

```
            it: first-->old
         next
              return temp;                  // return deleted link
               }

         //
----------------------------------------------------------------------------

         -
            public void displayList()
               {
               System.out.print("List (first-->last): ");
               Link current = first;       // start at
               beginning of listwhile(current != null)
                                           // until end
               of list,
                  {
                  current.displayLink();   //
                  print data current =
                  current.next;  // move to
                  next link
                  }
               System.out.println("");
               }

----------//----------------------------------------------------------------------

         -
            }  // end class LinkList

         ////////////////////////////////////////////////

         ///////////////class LinkListApp
            {
            public static void main(String[] args)
               {
               LinkList theList = new LinkList();  // make new list

               theList.insertFirst(22, 2.99);      //
               insert four itemstheList.insertFirst(44,
               4.99);
               theList.insertFirst(66, 6.99);
               theList.insertFirst(88, 8.99);

               theList.displayList();              // display list

               while( !theList.isEmpty() )         // until it's
               empty,
                  {
```

```
            Link aLink = theList.deleteFirst();    // delete link
            System.out.print("Deleted ");          // display it
            aLink.displayLink();
            System.out.println("");
            }
         theList.displayList();              // display list
         }  // end main()

}  // end class LinkList
```