```
//////////////////////////////////////////////////////
//www.karamian.com
//Program LinkedList
//////////////////////////////////////////////////////

import java.io.*;
import java.util.*;

//*************************************************************************
//   Class LinkedList
//*************************************************************************

public class LinkedList
{
   int length; // holds length of LinkedList
   ListNode firstNode; // holds first position of LinkedList
   ListNode pos; // holds position of LinkedList
   ListNode prev; // holds previous position of LinkedList
   ListNode current; // holds current position of LinkedList
   //=========================================================================
   //  The LinkedList constructor.
   //=========================================================================

   public LinkedList()
   {
      firstNode=new ListNode();
      pos=new ListNode();
      prev=new ListNode();
      current=new ListNode();
      firstNode=null;
      length=0;
   }// end LinkedList constructor

   //=========================================================================
   //  The appropriatePosition method.
   //  Finds the appropriate position for inserting a new value into LinkedList and
returns the
   //  position to insert.
   //=========================================================================

   public ListNode appropriatePosition(Distance n)
   {
      prev=null;
      pos=firstNode;
      while(pos!=null)
      {
         if(pos.distance.compareTo(n)>=0)
            return pos;
         prev=pos;
         pos=pos.link;
      }
      return pos;
   }// end appropriatePosition method

   //=========================================================================
   //  The insert method.
   //  Takes a new value and calls appropriatePosition, takes value returned by
   //  appropriatePosition and determines the correct location to put the new value,
while
   //  keeping the LinkedList sorted in ascending order.
   //=========================================================================

   public void insert()throws IOException
   {
```

```
      BufferedReader stdin = new BufferedReader(
                                              new InputStreamReader(System.in));
      ListNode newNode=new ListNode();
      System.out.print("To add enter feet: ");
      int f=Integer.parseInt(stdin.readLine());
      System.out.print("Enter inches: ");
      int i=Integer.parseInt(stdin.readLine());
      Distance d1=new Distance(f,i);
      appropriatePosition(d1);
      ListNode P=firstNode;
      newNode.distance=d1;
      if(length==0)
      {
         firstNode=newNode;
         firstNode.link=null;
      }
      else if(P==pos&&length>0)
      {
         ListNode T=firstNode;
         firstNode=newNode;
         firstNode.link=T;
      }
      else
      {
         while(P.link!=pos)
         {
            P=P.link;
         }
            if(P.link==null)
            {
               newNode.link=null;
               P.link=newNode;
            }
            else
               newNode.link=P.link;
               P.link=newNode;
      }
      length++;

   }// end insert method

   //=========================================================================
   //  The empty method.
   //  Checks to determine if LinkedList is empty, if empty returns -1 if not empty
returns 1.
   //=========================================================================

   public int empty(int length)
   {
      if(length==0)
         return -1;
      else
         return 1;

   }// end empty method.

   //=========================================================================
   //  The find method.
   //  Determines whether the value be looked for exists, if not found returns a
null value, else
   //  returns the value at the correct location.
   //=========================================================================

   public ListNode find(Distance d)
```

```
   {
      pos=firstNode;
      while(pos!=null)
      {
         if(pos.distance.compareTo(d)==0)
         {
            return pos;
         }
         else
         {
            pos=pos.link;
         }
      }
      return pos;

   }// end find method

   //=====================================================================
   //  The delete method.
   //  Takes a value in and calls the find method to determine if value exists in
LinkedList,
   //  if not found recieves a null value and prints appropriate method, otherwise
it deletes
   //  node and decrements length by 1 and keeps LinkedList sorted.
   //=====================================================================

   public void delete(Distance d)
   {
      find(d);
      if(pos==null)
      {
         System.out.println("Data not found.");
      }
      else
      {
         prev=firstNode;
         current=firstNode;
         if(current==pos)
         {
            firstNode=pos.link;
         }
         else
         {
            while(current!=pos)
            {
               prev=current;
               current=current.link;
            }
               if(current.link==null)
               {
                  prev.link=null;
               }
               else
                  prev.link=current.link;
         }
         length--;
      }


    }// end delete method

   //=====================================================================
   //  The clear method.
   //  When called sets firstNode to null value and sets length of LinkedList to 0,
```

```
creating an
   //  empty LinkedList.
   //=========================================================================

   public void clear()
   {
      firstNode=null;
      length=0;

   }// end clear method

   //=========================================================================
   //  The print method.
   //  Prints out the LinkedList list in ascending order starting the print at value
1 and
   //  incrementing till final value is printed.
   //=========================================================================

   public void print()
   {
      Distance c;
      int i=1;
      ListNode printList=firstNode;
      while(printList!=null)
      {
         c=printList.distance;
         System.out.print("Position "+ i+ ": ");
         c.printDistance();
         printList=printList.link;
         i++;
      }

   }// end print method

} //end of Class LinkedList
```