```java
import java.util.StringTokenizer;

import BasicIO.ASCIIDataFile;

/*This program searches and compare the ReservedWords
 * with the NestedSquares.java, it does a cross reference
 * if the word is a reserved word then it doens't show it
 * @author Long Nguyen
 * Student Number 5427059
 * @version 1.0 (Feb 10, 2014)
 * */

//This is the Link Class
public class Link {

 public String word;
 public int lineNum;

 public Link next; //reference the next link

   //List constructor
 public Link(String word, int lineNum){
  this.word = word;
  this.lineNum = lineNum;
 }

 //return the word of the Link
 public String getWord(){
  return word;
 }

 //display the information of the Link method
 public void display(){
   //System.out.println("The word is "+word + " " + lineNum);
         System.out.print(word + "    " + lineNum);

         //printing out the line number of the words if it's the same word
   while(this.next.word != null && this.word.equalsIgnoreCase(this.next.word)){
   System.out.print(" " + this.next.lineNum);
    //System.out.println();
       this.next = this.next.next;
  }
  System.out.println();
 }

 public static void main(String[] args) {

  ASCIIDataFile inReservedWordsFile = new ASCIIDataFile("JavaReservedWords.txt");
//file to be read
  ASCIIDataFile inNestedSquaresjava = new ASCIIDataFile("NestedSquares.java");
//file to be read


  LinkList inReservedWordsLinkedList = new LinkList();//creating the linkList
object
  LinkList crossReferenceLinkedList = new LinkList();//creating the linkList object

  //passing in the file "JavaReservedWords.txt" to be read
  String breakingUpReservedWordsFile= inReservedWordsLinkedList.readWord
(inReservedWordsFile);

  //passing in the file "NestedSquares.java" to be read
  //String breakingUpCrossReferenceFile= inReservedWordsLinkedList.readWord
(inNestedSquaresjava);

  StringTokenizer breakingUpWords = new StringTokenizer
(breakingUpReservedWordsFile);

  //StringTokenizer breakingUpbreakingUpCrossReferenceFile = new StringTokenizer
(breakingUpCrossReferenceFile);
```

```

  //breaking up the JavaReservedWords and creating a linkedLists
  while (breakingUpWords.hasMoreElements()) {

   inReservedWordsLinkedList.insertInOrder((String) breakingUpWords.nextElement(), (int)0);

  }

  ////////////////////////////////
  //This part reads in the code and check against the the reserved word list
  ///////////////////////////////
  int lineNumber = 1; //Increment for the line counter
  while(!inNestedSquaresjava.isEOF() && lineNumber < 40){//while it's the Java file is not end of file

  //passing in the file to be read and return the line as a string
  String breakingUpbreakingUpCrossReferenceFile = crossReferenceLinkedList.readLine(inNestedSquaresjava);

  //This part uses the String Tokenizer to  parse the string
  StringTokenizer CrossReferenceFile = new StringTokenizer(breakingUpbreakingUpCrossReferenceFile);
  //breaking up the words one at a time by white space

  LinkList FirstPointer =  new LinkList();
  //Points to the begining of the Link List of the Reserved Word Link List
   FirstPointer.firstLink = inReservedWordsLinkedList.firstLink;

  while (CrossReferenceFile.hasMoreElements()) {
   //casting the StringTokenize to string and setting it to worToMatch
    String wordToMatch = (String) CrossReferenceFile.nextElement();
    boolean wordNeverFound = false; //setting the boolean flag to false
   //while((inReservedWordsLinkedList.firstLink.word != wordToMatch)){

    while((!inReservedWordsLinkedList.firstLink.word.equalsIgnoreCase(wordToMatch))){

     //System.out.println(" word look up " + inReservedWordsLinkedList.firstLink.word);

    if(inReservedWordsLinkedList.firstLink.next == null){//checking the reserved while until the end of the link list
     wordNeverFound = true; // set word to never found
           break;//break out of the loop
    }else{//point to the next list list in the Reserved Word
     inReservedWordsLinkedList.firstLink = inReservedWordsLinkedList.firstLink.next;
    }//end else
    //If the word don't match the ReservedWord, then create a link list object of that word
   }if(wordNeverFound == true){
    crossReferenceLinkedList.insertInOrderCode((String) wordToMatch , lineNumber);
    //System.out.println(" Didn't find :) ");
   }//end if
  }//end while loop
  //System.out.println(" New Loop ");

  //pointer back to the begining of the linkListed Reserved Word
  inReservedWordsLinkedList.firstLink = FirstPointer.firstLink;
  lineNumber++;
  }//end while loop
  //inReservedWordsLinkedList.display();
  crossReferenceLinkedList.display();
 }

}//end class

//The pointer to the Link
class LinkList{
```

```java

/////////////////////////////////////////////
public String reservedWords;
public String result = "";
//////////////////////////////////////////


 public Link firstLink; // A reference to the first Link in the list or the last link that was added to the list

 LinkList(){
  firstLink = null; //first link always start as a null value
 }
 //checking if the link is empty
 public boolean isEmpty(){
  //if it's null then there's no data in the link
  return(firstLink == null);
 }

 //method to creating a new Link object
 public void insertFirstLink(String word, int lineNum){

  Link newLink = new Link(word, lineNum);// creating a new Link object

  newLink.next = firstLink; //point to the previous link, of the new object link that was created
  firstLink = newLink; //first Link points to the newly created link object, added the link into the link list

 }
 //to remove a link object of the linkList
 public Link removeFirst(){
  Link linkReference = firstLink;

  //checking if the link is empty before removing
  if(!isEmpty()){
   firstLink = firstLink.next;
  }else{
   System.out.println("The link list is empty ");
  }
  return linkReference; //return the deleted link
 }//end removeFirst Link object method

 //displaying the link list
 public void display(){
  System.out.print("Word Match     Line Number \n");
  Link theLink = firstLink;  // pointing the the beginning of the link

  while(theLink != null && theLink.next !=null){// while the link is not empty
   theLink.display();//calling the display method in the Link Class
   //System.out.println("Next Link: " + theLink.next );//print out the link data
   theLink = theLink.next; //pointing to the next link data
   System.out.println( );//print out a new line
  }
 }//end display method


 //This method insert the data in order
 public void insertInOrder(String word, int lineNum){
  Link newLink = new Link(word, lineNum);// creating a new Link object
  Link perviousLink = null; //perviousLink is set to null because the begining of the list won't have a pervious pointer
  Link currentLink = firstLink; //starting at the begining of the linkedList

  //while the link is not empty and the first character of the word in ASSIIC is greater then the linklisted word first Character
  while((currentLink != null) && ((int)word.charAt(0) > (int)currentLink.word.charAt(0))){
       perviousLink = currentLink;  //assign the perviousLink to the current link
       currentLink = currentLink.next;// currentLink points to next
  }//end while loop
  if(perviousLink == null){//if there is not pervious Link, means it on the first link
    firstLink = newLink; //point the firstLink pointer to the newly created newLink
  }//end if
  else{
   perviousLink.next =newLink;
  }
  newLink.next = currentLink;

 }//end insert in Order method



 //This method insert the data from the Java file
 public void insertInOrderCode(String word, int lineNum){
  Link newLink = new Link(word, lineNum);// creating a new Link object
  Link perviousLink = null; //perviousLink is set to null because the begining of the list won't have a pervious pointer
  Link currentLink = firstLink; //starting at the begining of the linkedList


  //while the link is not empty and the first character of the word in ASSIIC to lower case is greater then the linklisted word first Character
  while((currentLink != null) && ((int)word.toLowerCase().charAt(0) >= (int) currentLink.word.toLowerCase().charAt(0))){

   perviousLink = currentLink;  //assign the perviousLink to the current link
    currentLink = currentLink.next;// currentLink points to next
   //System.out.print(currentLink.word + " " + currentLink.lineNum + " ");

       //perviousLink = currentLink;  //assign the perviousLink to the current link
       //currentLink = currentLink.next;// currentLink points to next
                //}
  }//end while loop
  //System.out.println();


  if(perviousLink == null){//if there is not pervious Link, means it on the first link
    firstLink = newLink; //point the firstLink pointer to the newly created newLink
  }//end if
  else{
   perviousLink.next =newLink;
  }

  newLink.next = currentLink;



 }//end insert in Order method


 public String readWord(ASCIIDataFile file) {//reading the file as one long string

  ASCIIDataFile fileToRead = file; //passing in the file to read


   while(!fileToRead.isEOF()){//while until the end of file
    //hasNewLIne = in.readLine();

                  //line = fileToRead.readLine();
                  //System.out.println(line);
      result += fileToRead.readString().toString().replaceAll("[^a-z^A-Z]"," "); /*reading in as one long string and using
      //regex to match only  lower and uppercase letters in A-Z*/
     //System.out.println(result);
         result += " ";//adding a space when starting a new line
             //     }//end if
    }

    //System.out.println(result);
  return result;//return the string
}//end method readWord

 //This method read in the line one at a time
 public String readLine(ASCIIDataFile file) {
  String line = "";
  ASCIIDataFile fileToRead = file; //passing in the file to read

  //if(!fileToRead.isEOF()){
  line = fileToRead.readLine().replaceAll("[^a-z^A-Z]"," ");
  //}
  return line;
 }//end readLine Method

}
```