CHAPTER V

Implementation and Testing

5.1. Implementation

This class is used to make a node for tree.

```
1 class Nodedua
2 {
3         String isi;
4         Nodedua kiri,kanan;
5
6         public Nodedua(String isi)
7         {
8                 this.isi=isi;
9                 kiri=kanan=null;
10        }
11
12        public void setIsi(String a)
13        {
14                isi=a;
15        }
16
17        public void setKiri(Nodedua a)
18        {
19                kiri=a;
20        }
21
22        public void setKanan(Nodedua a)
23        {
24                kanan=a;
25        }
26
27        public String getIsi()
28        {
29                return isi;
30        }
31
32        public Nodedua getKiri()
33        {
34                return kiri;
35        }
36
37        public Nodedua getKanan()
38        {
39                return kanan;
40        }
41 }
```

Link class is to make link that used in the list of linked list for hash table

```
class Link
{
        public String data;
        public Link next;

        public Link(String i)
        {
                data=i;
                next=null;
        }

        public void displayLink()
        {
                if(next == null)
                        System.out.print(" (" + data + "," +next+ ") ");
                else
                        System.out.print(" (" + data + ") ");
        }

        public Link getNext()
        {
                return next;
        }

        public void setNext (Link link)
        {
                next = link;
        }
}
```

Make hash table for the data structures

```
        public HashTable(int size)
        {
                hashArray = new LinkList[size];

                for(int i=0; i<hashArray.length; i++)
                {
                        hashArray[i] = new LinkList();
                }
        }
```

Fill hash table with concordance database

```
try
{
        FileInputStream fstream = new FileInputStream("../konkordansisuratpaulus.txt");

        DataInputStream in = new DataInputStream(fstream);

        BufferedReader br = new BufferedReader(new InputStreamReader(in));

        String strLine;

        while ((strLine = br.readLine()) != null)
        {

                String arrydata[]=strLine.split(";");

                String ayat=arrydata[2]+":"+arrydata[3]+":"+arrydata[4];

                abc.cek(arrydata[1],ayat);
        }
        in.close();
}
catch(Exception e)
{
        System.err.println("Error: " + e.getMessage());
}
```

This is method to check if there is collision in the hashtable

```
public void cek(String key,String ayat)
{
        int hashVal=hashFunction(key);

        hashArray[hashVal].findLink(key,ayat);
}
```

Fill the linked list with the list of nodes tree

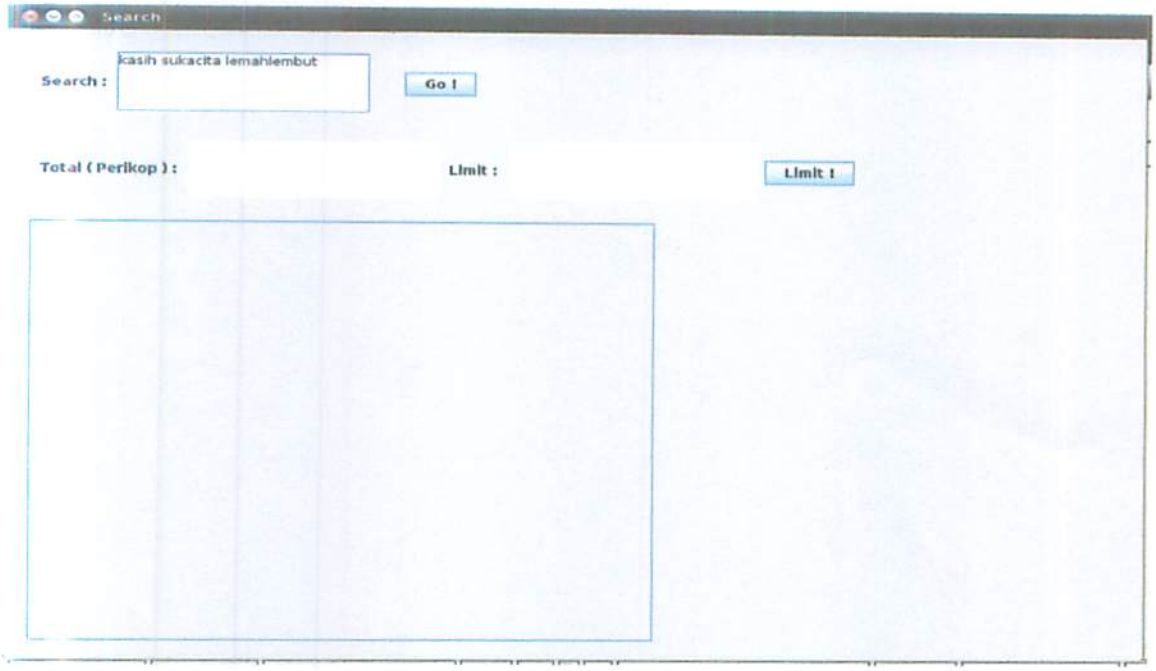
```
90          public void findLink(String cek,String ayat)
91          {
92                  Link nw = new Link(cek);
93
94                  if(isEmpty())
95                  {
96                          listArray = new Treedua[150];
97
98                          for(int i=0; i<listArray.length; i++)
99                          {
100                                 listArray[i] = new Treedua();
101                         }
102
103                         listArray[i].add(ayat);
104
105                         head=nw;
106                         tail=nw;
107                 }
108
109                 else
110                 {
111                         Link curr=head;
112
113                         while(!curr.data.equals(cek))
114                         {
115                                 if(curr.next == null)
116                                 {
117                                         i++;
118                                         listArray[i].add(ayat);
119                                         curr.next=nw;
120                                 }
121
122                                 else
123                                 {
124                                         curr=curr.next;
125                                 }
126                         }
127
128
129                         if(curr.data.equals(cek))
130                         {
131                                 listArray[i].add(ayat);
132                         }
133                 }
134         }
135
```

Method to add / insert new nodes in the tree

```
14        public void add(String isi)
15        {
16                baru=new Nodedua(isi);
17
18                if(root==null)
19                {
20                        root=baru;
21                }
22
23                else
24                {
25                        insert(root,baru);
26                }
27        }
28
29        public void insert(Nodedua parent,Nodedua baru)
30        {
31                if(baru.isi.compareTo(parent.isi) < 0)
32                {
33                        if (parent.kiri != null)
34                        {
35                                // Go more depth to left.
36                                insert(parent.kiri, baru);
37                        }
38
39                        else
40                        {
41                                parent.kiri = new Nodedua(baru.isi);
42                        }
43                }
44
45                else if(baru.isi.compareTo(parent.isi) > 0)
46                {
47                        if (parent.kanan != null)
48                        {
49                                // Go more depth to right
50                                insert(parent.kanan, baru);
51                        }
52
53                        else
54                        {
55                                parent.kanan = new Nodedua(baru.isi);
56                        }
57                }
58        }
```

5.2. Testing

The application interface in the first process



Words from input box will be case folded and tokenized

```
6       public String[] caseFolding(String kalimat,String[] check)
7       {
8               System.out.println("\n == Case Folding == \n");
9
10              System.out.println(" Sebelum : " + kalimat);
11
12              kalimat = kalimat.toLowerCase();
13
14              System.out.println(" Sesudah : " + kalimat + "\n");
15
16              tokenisasi(kalimat,check);
17
18              return check;
19      }
```

```java
21        public String[] tokenisasi(String kalimat, String[] check)
22        {
23                System.out.println(" == Tokenisasi == \n");
24
25                System.out.println(" Kalimat : " + kalimat);
26
27                String tokenisasi[]=kalimat.split(" ");
28
29                for(int i=0;i<tokenisasi.length;i++)
30                {
31                        System.out.println("["+ i + "]" + " " + tokenisasi[i]);
32                }
33
34                stopWords(tokenisasi,check);
35
36                return check;
37        }
```

After that, throwing the stopwords from the input phrase

```java
39        public String[] stopWords(String[] tokenisasi, String[] check)
40        {
41                int flag=0, a=0;
42                String[] result;
43                result = new String[10];
44
45                System.out.println("\n == Stopwords == " + "\n");
46
47
48
49                try{
50                        FileInputStream fstream = new FileInputStream("../stopwords.txt");
51
52                        DataInputStream in = new DataInputStream(fstream);
53
54                        BufferedReader br = new BufferedReader(new InputStreamReader(in));
55
56                        String strLine;
57
58                        while ((strLine = br.readLine()) != null)
59                        {
60
61                                String stopwords[]=strLine.split(";");
62
63                                for(int i=0;i<tokenisasi.length;i++)
64                                {
65                                        for(int j=0;j<stopwords.length;j++)
66                                        {
67                                                if(tokenisasi[i].equals(stopwords[j]))
68                                                {
69                                                        flag=1;
70
71                                                }
72                                        }
73                                        if(flag==0)
74                                        {
75                                                result[a]=tokenisasi[i];
76
77                                                a++;
78                                        }flag=0;
79                                }
80
81                                for(a=0;a<result.length;a++)
82                                {
83                                        if(result[a]!=null)
84                                        {
85                                                System.out.println(" "+result[a]);
86                                        }
87                                }
88
89                                System.out.print("\n");
```

Check the words in the array whether there is in the database or not, if it is not available, it comes to Nazief and Adriani Algorithms to find the root words.

```
if(abc.find(result[a])!=null)
{
        abc.cetak(result[a]);
        check[a]=result[a];
}

else
{
        String hasil=infleksiSufix(result[a]);
        System.out.println("Hasil infleksi sufix : "+hasil);

                hasil=derivasiSufix(hasil);
        System.out.println("Hasil derivasi sufix : "+hasil);

                hasil=derivasiPrefix(hasil);
        System.out.println("Hasil derivasi prefix : "+hasil);

        abc.cetak(hasil);

        check[a]=hasil;
}
```

Check for the inflexion suffix, such as -kah, -lah, -tah, -pun, -ku, -mu, -nya

```
199
200     public String infleksiSufix(String cek) // -kah, -lah, -tah, -pun, -ku, -mu, -nya
201     {
202             String[] m = {"kah","lah","tah","pun","nya","ku","mu"};
203             String hasil=cek;
204             int count=0;
205
206             for(int i=0;i<m.length;i++)
207             {
208                     if(cek.matches("(.*)"+m[i]))
209                     {
210                             count=m[i].length();
211                             int p = cek.length();
212                             cek = cek.substring(0, (p - count));
213                             System.out.println("infleksi sufix");
214                             System.out.println(cek);break;
215                     }
216
217             }
218
219             return cek;
220     }
```

Check for the derivation suffix, such as -i, -kan, -an

```
222         public String derivasiSufix(String cek) // -i, -kan, -an
223         {
224                 String res = cek;
225                 String[] m = {"i","kan","an"};
226                 String hasil="",value="true";
227                 int count=0;
228
229                 for(int i=0;i<m.length;i++)
230                 {
231                         if(res.matches("(.*)"+m[i]))
232                         {
233                                 count=m[i].length();
234                                 int length = res.length();
235                                 hasil = res.substring(0, (length - count));
236
237                                 System.out.println("derivasi sufix");
238                                 System.out.println(hasil);
239
240                                 if(abc.find(hasil)!=null)
241                                 {       cek =hasil;
242                                         System.out.println(cek);
243
244                                 }
245
246                                 else
247                                 {
248                                         if(disallowedPrefixSufix(res)==true)
249                                         {
250                                                 cek=cek;System.out.println(cek);
251                                         }
252
253                                         else {cek=hasil;}
254                                 }
255
256                                 break;
257                         }
258
259                 }
260
261
262
263                 return cek;
264         }
265
```

Check for the derivation prefix, such as di- , ke- , se- , be- , me- , pe- , te-

```
294     public String derivasiPrefix(String cek) // di- , ke- , se- , be- , me- , pe- , te-
295     {
296             String[] m = {"di","ke","se","be","me","pe","te"};
297             String hasil=cek,value="true";
298             int count=0;
299
300             for(int i=0;i<m.length;i++)
301             {
302                     if(cek.matches(m[i]+"(.*)"))
303                     {
304                             count=m[i].length();
305                             int length = cek.length();
306                             hasil = cek.substring(length-(length - count));
307
308                             System.out.println("derivasi prefix");
309                             System.out.println(hasil);
310                             System.out.println(cek);
311
312                             if(abc.find(hasil)!=null)
313                             {       cek =hasil;
314                                     System.out.println(cek);
315
316                             }
317                             else
318                             {
319                                     cek=cek;
320                             }
321
322                             break;
323                     }
324
325             }
```

After finding the root words using Nazief and Adriani algorithms, find the root words in the data structures of concordance and then find the verses in the data structures of bible.

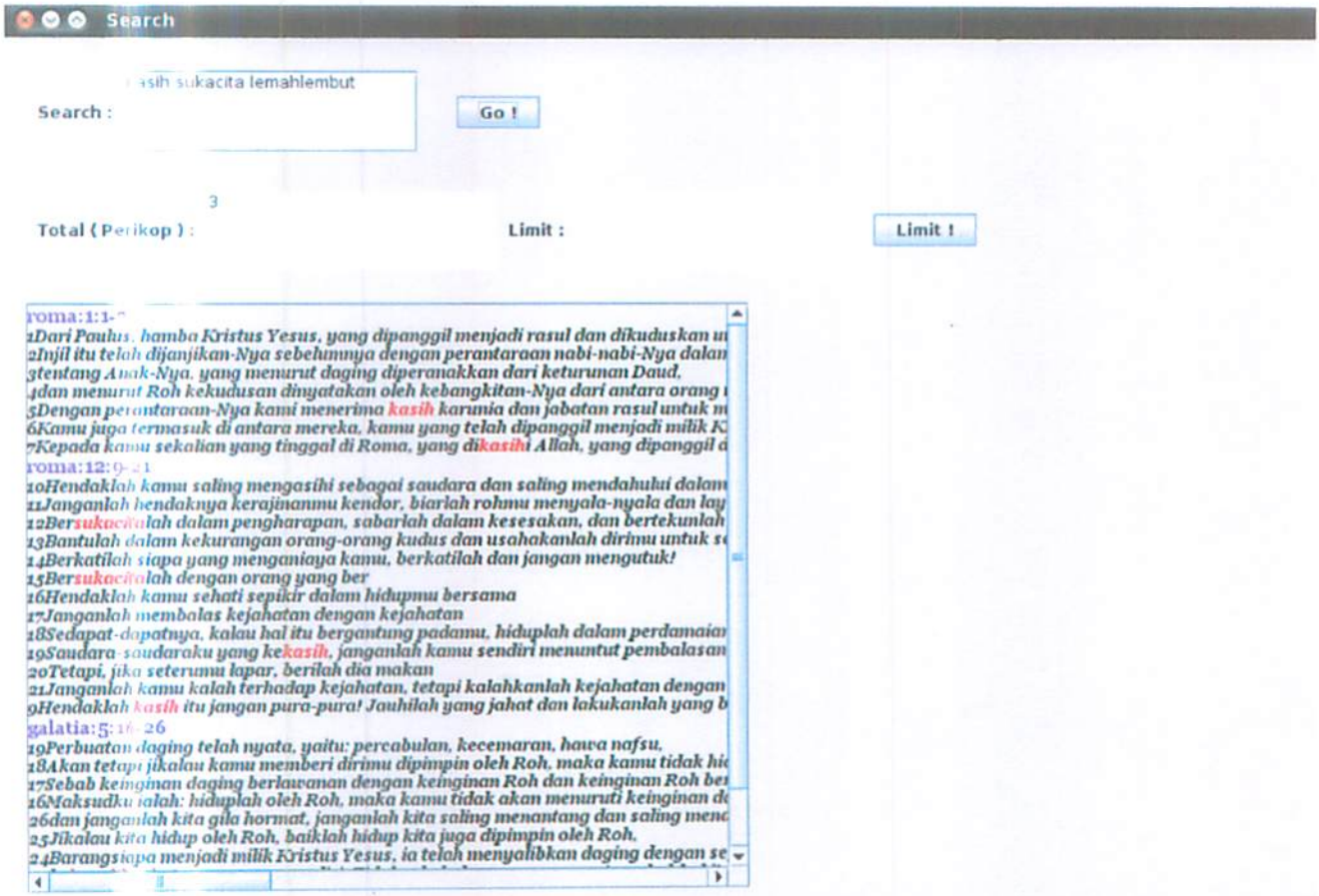
```
        //System.out.println("hoi");System.out.println(arrayisi[0]);
        alkitab.cetakalkitab(arrayisi[0],arrayisi[1],ayat,ayatakhir);
    public  void cetakalkitab(String injil,String pasal,int ayat, int ayatakhir)
    {
            //System.out.println("bisa");
                    int hashVal=hashFunction(injil);

            hashArray[hashVal].cariinjil(injil,pasal,ayat,ayatakhir);

    }

    public Linkdua cariinjil(String injil,String pasal,int ayat, int ayatakhir)
    {
            Linkdua curr=head, res=null;
            int num=0;
            String result="";
            System.out.print("Searching: " + injil + " ... ");

            while(!curr.data.equals(injil))
            {
                    num++;
                    if(curr.next == null)
                    {
                            result="Can't found data";
                            break;
                    }
                    else
                    {
                            curr=curr.next;
                    }
            }

            if(curr.data.equals(injil))
            {
                    result="Found at # " + num;
                    res=curr;
            }
System.out.println(" "+result+" "+injil);

            listArray[num].caritreealkitab(injil,pasal,ayat,ayatakhir);


            return res;
    }
```

After finding the result from the data structures of Bible, it is written in the file named result.txt to be shown in the GUI interface.

```
121     public void searchTreealkitab(Nodeduadua node,String injil,String pasalcari,int ayat, int ayatakhir)
122     {               //System.out.print("searchtreealkitab");
123             if (node != null) {
124                 searchTreealkitab(node.kiri,injil,pasalcari,ayat,ayatakhir);
125                System.out.print("nodekiri sta");
126                             String pasalasli=node.pasal;
127                             String isiasli=node.isi;
128                             String splitisi[];
129
130
131                     if(pasalasli.equals(pasalcari))
132                     {//System.out.print("equals pasalcari");
133                             splitisi=isiasli.split("::");
134
135                             int ayatasli=Integer.parseInt(splitisi[0]);
136
137                             for(int apa=ayat;apa<=ayatakhir;apa++)
138                             {
139                                     System.out.print("for ayat");
140
141
142                                     if(ayatasli==apa)
143                                     {
144                                                     String res=ayatasli+splitisi[1];
145
146
147                                                     String file="result.txt";
148                                                     try
149                                                     {
150                                                             FileWriter fw= new FileWriter(file,true);
151                                                             fw.write(res+"\n");
152                                                             fw.close();
153                                                     }
154                                                     catch (Exception e){
155                                                             System.err.println("Error: " + e.getMessage());
156                                                     }
157                                     }
158                             }
159                     }
```

After getting through such process, and finally written in the file named result.txt, the program will load the result from file result.txt and then show it in the GUI by using Jlabel. The spesific words that inputed from text area will be bold so that the result can clearly be seen.

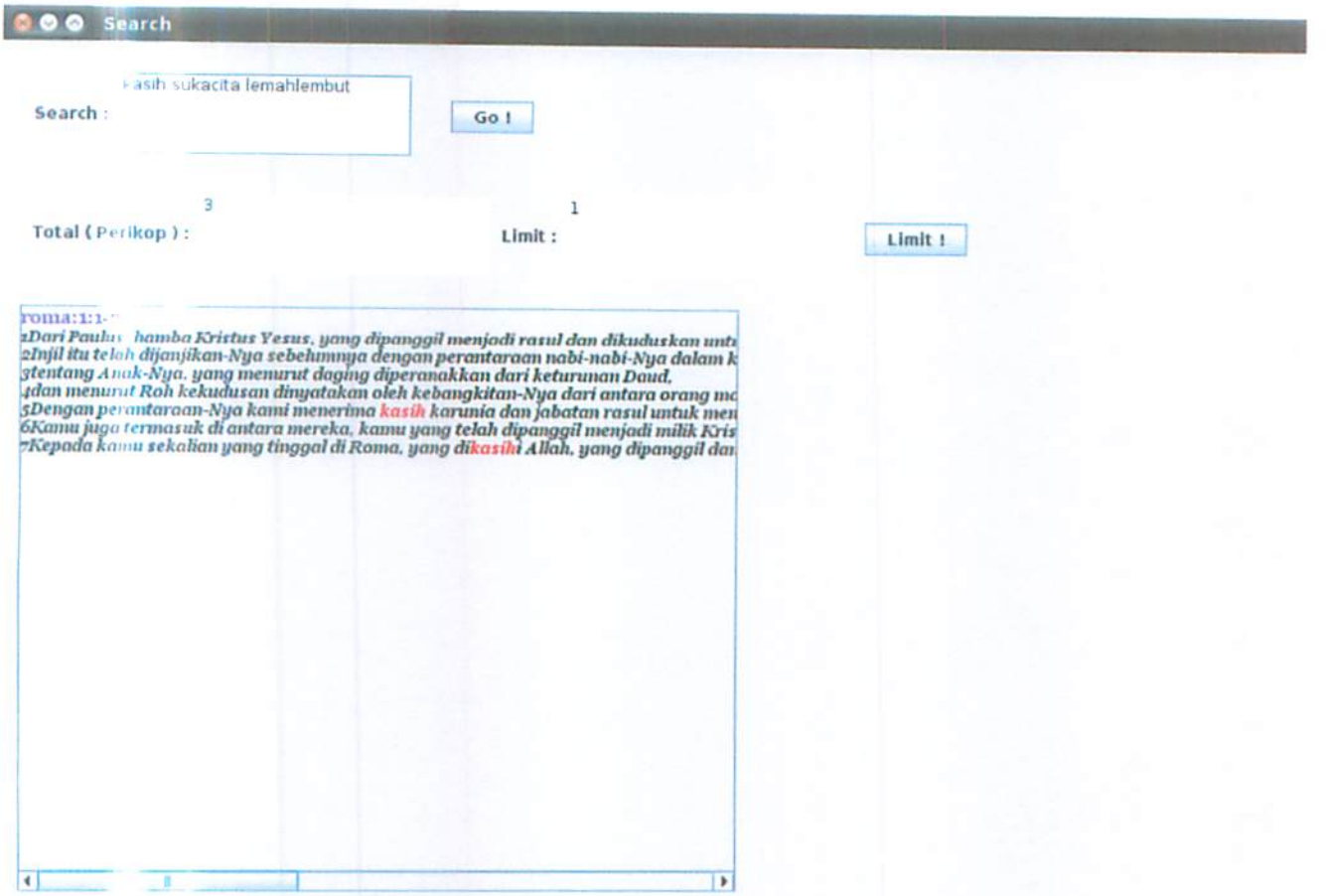
```
193
194                                 try{
195                                         FileInputStream fstream = new FileInputStream("result.txt");
196
197                                         DataInputStream in = new DataInputStream(fstream);
198
199                                         BufferedReader br = new BufferedReader(new InputStreamReader(in));
200
201                                         String strLine, h, cek="";
202
234
235                                             String ayat[]=strLine.split(check[j]);
236                                             panel3.add(new JLabel("<html><i><font color='black' size='3' face='Georgia,
    Arial, Garamond'>"+ayat[0]+"</font><font color='red' size='3' face='Georgia, Arial, Garamond'>"+check[j]+"</font><font color='black'
    size='3' face='Georgia, Arial, Garamond'>"+ayat[1]+"</font></i></html>"));
237                                             System.out.println("Warna");j=check.length;cek="red";
238
```



The result can be limited to be shown in the GUI, enter the limit on the areabox and then press limit button. The input from limit area box will be the limitation for looping the result showed in the GUI so user can automatically limit the result showed in the GUI using this feature.

```
396
397                     int j=0,i=0,counter=0;
398
399                     while ((strLine = br.readLine()) != null && counter !=bts+1)
400                     {
401                             i=0;
402
403                             int stop=0;
404
405                             String[] m = {"roma","1 korintus","2 korintus","galatia","efesus","filipi","kolose","1
    tesalonika","2 tesalonika","1 timotius","2 timotius","titus","filemon"};
406                             for(i=0;i<m.length;i++)
407                             {
408                                     if(strLine.matches(m[i]+"(.*)"))
409                                     {
410
411                                             h=strLine;
412                                             cek="perikop";i=m.length-1;counter++;
413
414                                     }
415
416
417                                     else
418                                     {
419                                             if(stop==0)
420                                             {
421                                                     for( j=0;j<check.length;j++)
422                                                     {
423
424                                                             if(check[j]!=null)
425                                                             {
426                                                                     if(strLine.matches("(.*)"+check[j]+"(.*)"))
427                                                                     {
428
429                                     String ayat[]=strLine.split(check[j]);
430                                     panel3.add(new JLabel("<html><i><font color='black' size='3' face='Georgia, Arial,
    Garamond'>"+ayat[0]+"</font><font color='red' size='3' face='Georgia, Arial, Garamond'>"+check[j]+"</font><font color='black' size='3'
    face='Georgia, Arial, Garamond'>"+ayat[1]+"</font></i></html>"));
431                                     System.out.println("Warna");j=check.length;cek="red";
432
433                                                                     }
434                                                                     else
435                                                                     {
436                                                                             cek="ayat";
437                                                                     }
438
439                                                             }
440                                                             stop=1;
441
442                                                     }
```

This is the result on the GUI after using limit feature



5.3. Application Interface

Application Interface in the first process

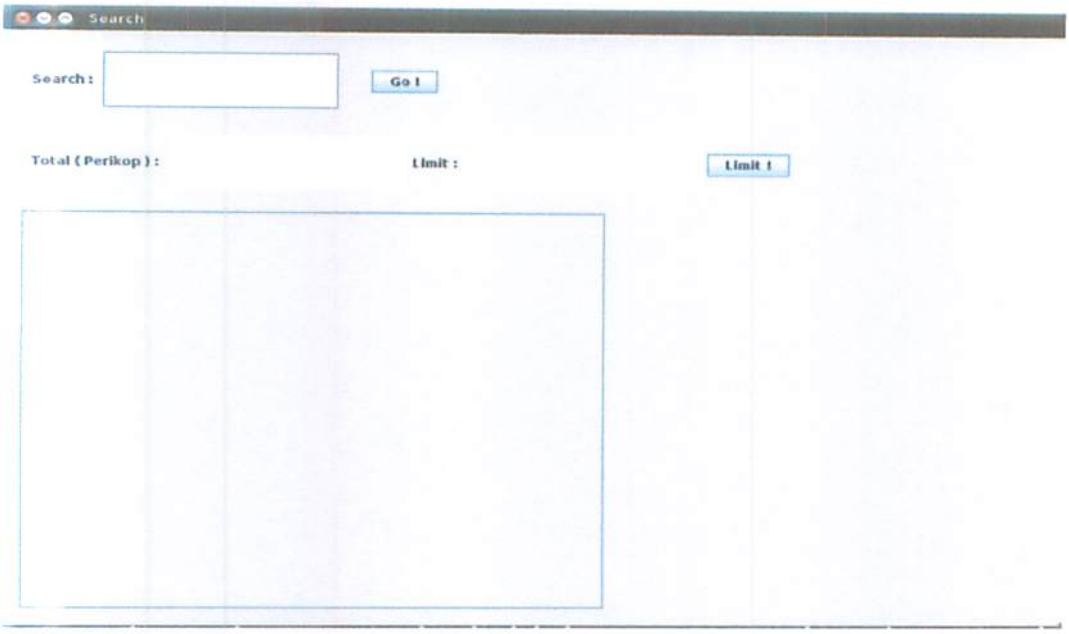


Figure 5.1 Main Interface

The result of process searching

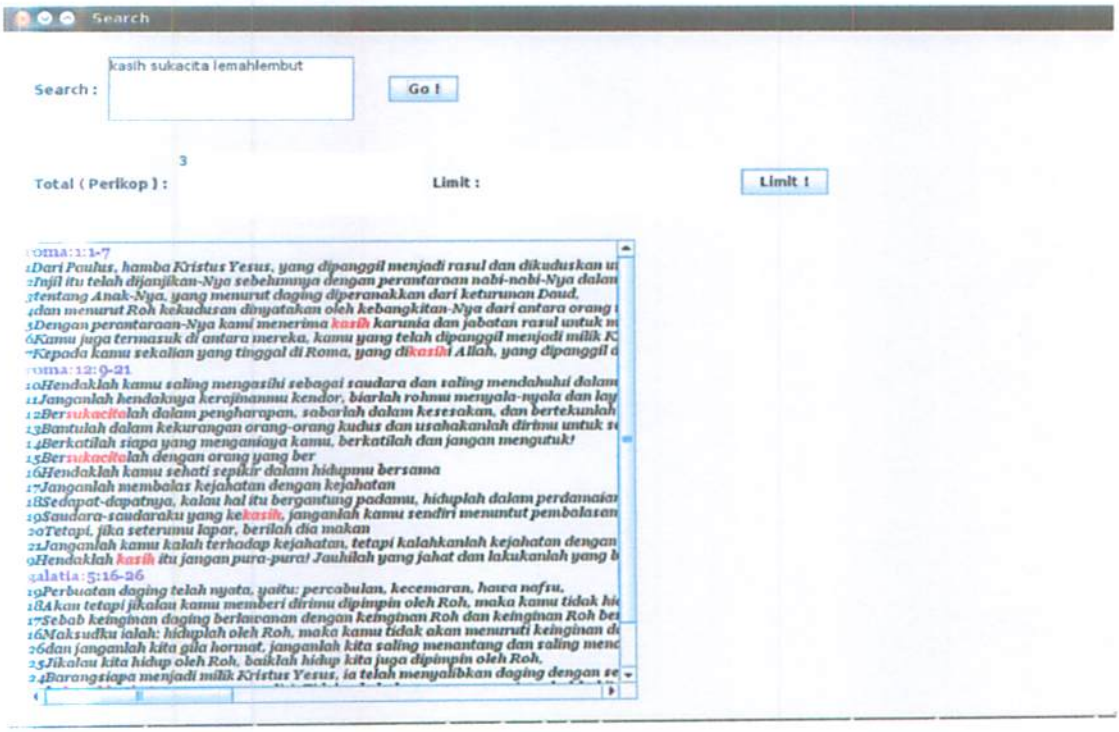


Figure 5.2 Result of Searching

After pressing the limit button to limit the result showed in the GUI interface

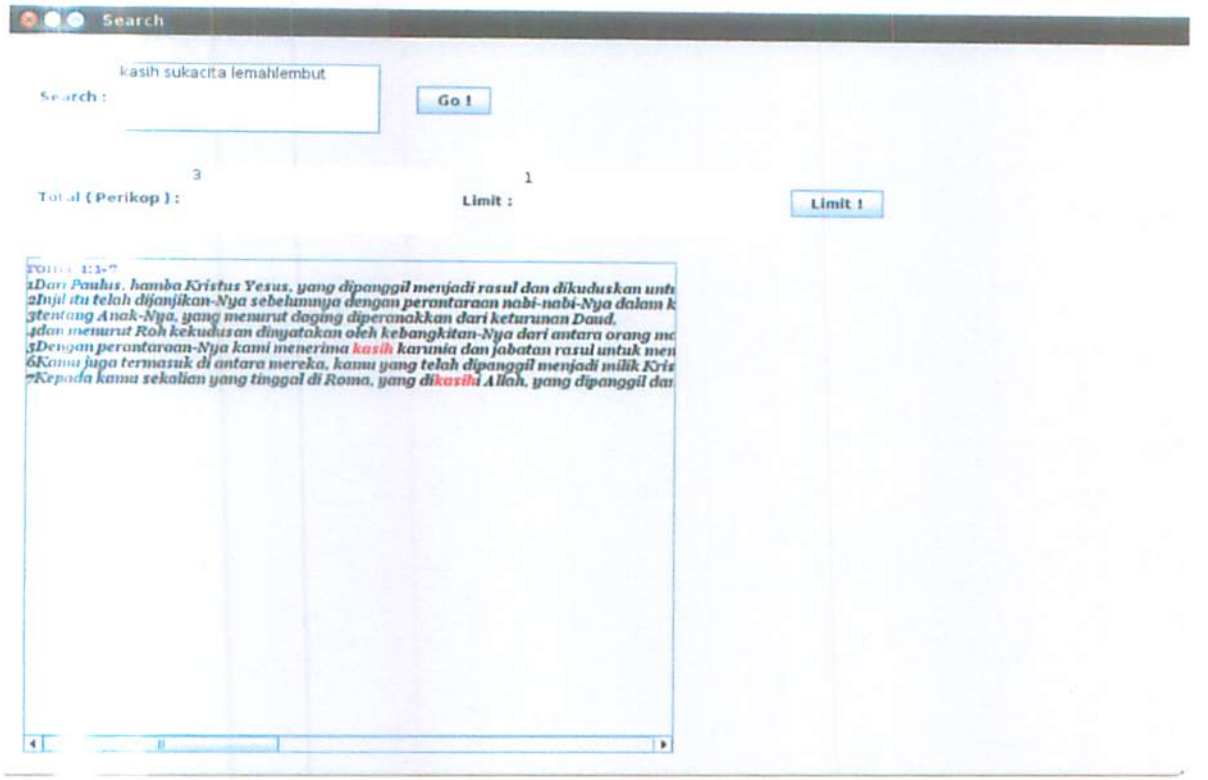


Figure 5.3 Result After Limitation