Appendix D: Sample Code-Java Programs

Code of some Partial Set of Programs developed as part of the Research Work, each one used for various purposes:

FinalMainMenu.java

/// Beginning of Program: FinalMainMenu.java ///

```
/* Code from where the whole application begins.
Class which defines main menu from where other sub-
modular menus are invoked */

import java.io.*;

class FinalMainMenu {

     static int menuchoice;
     static char continuechoice;

     public static void main (String args[]) throws
IOException    {
          FinalMainMenu.doFinalMainMenu();
     }


     public static void doFinalMainMenu() throws
IOException {
          do {
               BufferedReader br_continuechoice = new
BufferedReader(new InputStreamReader(System.in) );
               doMenu();

               if ((char)menuchoice == '1')
doMenuChoice1();
               else if ((char)menuchoice == '2')
doMenuChoice2();
               else if ((char)menuchoice == '3')
doMenuChoice3();
               else if ((char)menuchoice == '4')
doMenuChoice4();
               else System.out.println("Enter the choice
as per Menu");
```

```
                System.out.println("Do you want to
continue (y/n) : ");
                continuechoice = (char)
br_continuechoice.read();
          } while (continuechoice == 'y');
     }

     public static void doMenu() throws IOException {

          BufferedReader br_menuchoice = new
BufferedReader(new InputStreamReader(System.in) );
          do {
          System.out.println ("========Menu========");
          System.out.println ("1) Want to perform
operations on Reads");
          System.out.println ("2) Want to perform
operations on Contigs");
          System.out.println ("3) Want to perform
operations on Chromosomes");
          System.out.println ("4) Exit");
          System.out.println ();
          System.out.println ("Enter your choice : ");
          menuchoice = br_menuchoice.read();
          } while (menuchoice <= 4);
     }

     public static void doMenuChoice1() throws
IOException {
          FinalMainRead.doFinalMainRead();
     }

     public static void doMenuChoice2() throws
IOException {
          FinalMainContig.doFinalMainContig();
     }

     public static void doMenuChoice3() throws
IOException {
          FinalMainChromosome.doFinalMainChromosome();
     }

     public static void doMenuChoice4() throws
IOException {
          System.exit(0);
     }

} //End of class FinalMainMenu
```

/// End of Program: FinalMainMenu.java ///

FinalMainRead.java

//// Beginning of Program: FinalMainRead.java ////

```
/* Code that displays the menu options to perform various
operations with the Reads data */

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

class FinalMainRead {

          static int menuchoice;
          static char continuechoice;
          static ReadFromFastaFile orfff;

          static ReadDB readdb;
          static ArrayList<Read> ordal ;
          static Read temp_read;
          static ArrayList<String> ordseqal = new
ArrayList<String>();

     public static void main (String args[]) throws
IOException    {
          FinalMainRead.doFinalMainRead();
     }


     public static void doFinalMainRead() throws
IOException {
          do {
               BufferedReader br_continuechoice = new
BufferedReader(new InputStreamReader(System.in) );
               doMenu();

               if ((char)menuchoice == '1')
doMenuChoice1();
               else if ((char)menuchoice == '2')
doMenuChoice2();
               else if ((char)menuchoice == '3')
doMenuChoice3();
               else if ((char)menuchoice == '4')
doMenuChoice4();
               else if ((char)menuchoice == '5')
doMenuChoice5();
               else if ((char)menuchoice == '6')
doMenuChoice6();
```

```
                else System.out.println("Enter the choice
as per Menu");

                System.out.println("Do you want to
continue (y/n) : ");
                continuechoice = (char)
br_continuechoice.read();
          } while (continuechoice == 'y');
     }

     public static void doMenu() throws IOException {

          BufferedReader br_menuchoice = new
BufferedReader(new InputStreamReader(System.in) );
          do {
          System.out.println ("========Menu========");
          System.out.println ("1) Want to only retrieve
Reads from Fasta File");
          System.out.println ("2) Want to create
InsertScript for Reads from Fasta File");
          System.out.println ("3) Want to insert Reads
into Database");
          System.out.println ("4) Want to retrieve Reads
Sequence from Database");
//        System.out.println ("5) Want to assemble the
Reads into Contigs");
          System.out.println ("6) Exit");
          System.out.println ();
          System.out.println ("Enter your choice : ");
          menuchoice = br_menuchoice.read();
          } while (menuchoice <= 6);
     }

     public static void doMenuChoice1() throws
IOException {
          orfff = new ReadFromFastaFile();
          readdb = orfff.doReadFromFastaFile();
     }

     public static void doMenuChoice2() throws
IOException {
          doMenuChoice1();

     CreateInsertScriptFile.doCreateInsertScriptFile(read
db.insert);
     }

     public static void doMenuChoice3() throws
IOException {
          doMenuChoice1();
          ReadToDB ordtodb = new ReadToDB();
```

```
            ordtodb.getConnection();
            ordtodb.getStatement();
            ordtodb.doExecuteUpdate(readdb);
            ordtodb.doExecuteQuery();
      }

      public static void doMenuChoice4() throws
IOException {
            ReadFromDB ordfdb = new ReadFromDB();
            ordfdb.getConnection();
            ordfdb.getStatement();
            temp_read = new Read();
            ordseqal = new ArrayList<String>();


            ArrayList<String> oal =
ordfdb.doExecuteQueryCollection();

            Read temp_read = new Read();
            for (int i = 0; i< oal.size(); ++i) {
                  temp_read.read_sequence = oal.get(i);
                  ordseqal.add(temp_read.read_sequence);
                  System.out.println("Read is : " +
ordseqal.get(i) );
            }
      }

      public static void doMenuChoice5() throws
IOException {
//          System.exit(0);
      }

      public static void doMenuChoice6() throws
IOException {
            System.exit(0);
      }

} //End of class FinalMainRead
```

/// End of Program : FinalMainRead.java ///

ConvertSequenceToBinary.java

/// Beginning of Program: ConvertSequenceToBinary.java ///

/*

Code developed using Programming Language: Java

Program to convert the DNA sequence which is in string form, to Numerical form using Single Binary Indicator, so that Wavelet Transforms can be applied on this signal, that is the Numerical form of DNA Sequence

*/

```
import java.util.*;
import java.util.regex.*;

public class ConvertSequenceToBinary {


     static int[] convertSequenceToBinary(String s) {

     StringBuffer dna_seqn = new StringBuffer(s);

     GenerateDNASequenceIndexBinary.generateDNASequence()
;
     int i = 0;
     GenerateDNAToken ogdt = new GenerateDNAToken();
     int binary_seqn[] = new int[(dna_seqn.length() /
4)];

     int index=0, fromIndex = 0;
     String sb = new String();
     int fr = 0;

try {

     do {
          sb = dna_seqn.substring(fromIndex, fromIndex +
4);
          ogdt = GenerateDNALookupTable.getValue(sb);
```

```
            binary_seqn[i] = ogdt.index;

            fromIndex = fromIndex+4;
            i++;

      } while (fromIndex < (dna_seqn.length() -
dna_seqn.length()%4) - 4 +1);

      }
      catch(Exception e ) {System.out.println(e);}

      return(binary_seqn);

      }
```

/// End of Program: ConvertSequenceToBinary.java ///

GenerateDNAToken.java

////// Beginning of the Program : GenerateDNAToken.java ///

```
public class GenerateDNAToken {

     int id;
     int index;
     String sequence;
     int binary;

     GenerateDNAToken() {}

     GenerateDNAToken(int i, String s, int b)
     {
          this.index = i;
          this.sequence = s;
          this.binary = b;
     }

     GenerateDNAToken(int id, int i, String s, int b)
     {
          this.id = id;
          this.index = i;
          this.sequence = s;
          this.binary = b;
     }


     static void addDNATokenToDNALookupTable
(GenerateDNAToken ogdt){


     GenerateDNALookupTable.createDNALookupTable(ogdt.seq
uence.toString(), ogdt);

     }
```

////// End of the Program : GenerateDNAToken.java ///

CreateInsertScriptFile.java

/// Beginning of the Program : CreateInsertScriptFile.java ///

```
import java.io.*;

class CreateInsertScriptFile {

     static void doCreateInsertScriptFile(String[]
insert_script) throws IOException {
          System.out.println("Enter the filename, in
which to save the insert script (Eg. reads.sql : ");
          BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
          String filename = br.readLine();
          File of = new File(filename);
          FileWriter ofw = new FileWriter(of);
          for (int i = 0; i < insert_script.length; ++i)
{
               ofw.write(insert_script[i]+"\n");
          }
          ofw.close();
     }
```

/// End of the Program : CreateInsertScriptFile.java ///

IPDemo.java

/// Beginning of the Program: IPDemo.java ///

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.util.ArrayList;
import java.util.Enumeration;
import java.util.Properties;

public class IPDemo
{
     static ArrayList<String> al=new ArrayList<String>();
     static File file=null;
     FileInputStream fis=null;
     FileOutputStream fos=null;
     Properties prop=null,prop1=null;
     static ArrayList< String > ipMacList=null;

     public static void main(String args[]) throws
InterruptedException
     {

          if (args.length < 2)
               {

                    System.out.println("Enter command as
: java IPDemo <start_ip> <end_ip> ");
                    return;
               }

          IPDemo ip=new IPDemo();


//        String startRange="192.82.36.1";
//        String endRange="192.82.36.255";

          String startRange= args[0];
          String endRange= args[1];


          String startArr[]=startRange.split("\\.");
          String endArr[]=endRange.split("\\.");

          int a1=Integer.valueOf(startArr[0]);
          int a2=Integer.valueOf(startArr[1]);
          int a3=Integer.valueOf(startArr[2]);
          int a4=Integer.valueOf(startArr[3]);
```

```
            int a5=Integer.valueOf(endArr[0]);
            int a6=Integer.valueOf(endArr[1]);
            int a7=Integer.valueOf(endArr[2]);
            int a8=Integer.valueOf(endArr[3]);

            ip.split(a1,a2,a3,a4,a5,a6,a7,a8);

            Scan scan=new Scan();
            scan.makeThread(a1);
            for(Thread t:scan.t)
              t.join();

            ip.writePropData(Scan.final_result);
            ArrayList< String
>maciplist=ip.readPropData(file);
            /*for(int i=0;i<maciplist.size();i++)
            {
                 System.out.println(maciplist.get(i));
            }*/

            IPPort ipPort=new IPPort();
            ArrayList<String >
ipportlist=ipPort.writeIPPort(ipMacList);
            ipPort.readIPPort(ipportlist);
            ArrayList<String>
macUnamePasswordList=ipPort.writeMacUnamePassword(ipMacLi
st);

      ipPort.readMacUnamePassword(macUnamePasswordList);

            IPPortMac ipPortMac=new IPPortMac();
            ArrayList<String>
finalmap=ipPortMac.readIPPortMac(maciplist, ipportlist,
macUnamePasswordList);

            ipPortMac.writeProp(finalmap);
            ipPortMac.readProp(finalmap);

      }

      public void split(int a,int b,int c,int d,int a1,int
b1,int c1,int d1) throws InterruptedException
      {
            while(true)
            {
                 if(a==a1)
                 {
                      if(b==b1)
                      {
                           if(c==c1)
                           {
```

```
                                        if(d==d1)
                                        {
                                             break;
                                        }
                                        else
                                        {
                                             RangeScan sc=new
RangeScan(a,b,c);
                                             break;
                                        }
                                   }
                                   else
                                   {
                                        if(c==255)
                                        {
                                             c=1;
                                        }
                                        else
                                        {
                                             RangeScan sc=new
RangeScan(a,b,c);
                                             c++;
                                        }
                                   }
                              }
                              else
                              {
                                   if(c==255)
                                   {
                                        b++;
                                        c=1;
                                   }
                                   else
                                   {
                                        RangeScan sc=new
RangeScan(a,b,c);
                                        c++;
                                   }
                              }
                         }
                    }
               }

     public void writePropData(ArrayList<String>
ipmaclist)
     {
          try
          {
               ipMacList =new ArrayList<String>();
               file=new File("D:\\ip_mac.prop");
```

```
                fos=new FileOutputStream(file);
                prop=new Properties();

                for(int i=0;i<ipmaclist.size();i++)
                {

     prop.setProperty(ipmaclist.get(i).split(":")[1],ipma
clist.get(i).split(":")[0]);

     ipMacList.add(ipmaclist.get(i).split(":")[1]+":"+ipm
aclist.get(i).split(":")[0]);
                }

                prop.store(fos, null);
                fos.close();
           }
           catch(Exception e)
           {
                System.out.println(e.toString());
           }
     }
     public ArrayList<String> readPropData(File
ipmacfile)
     {
          ArrayList< String > macipList=new
ArrayList<String>();
           try
           {
                fis=new FileInputStream(ipmacfile);
                prop.load(fis);

                Enumeration<Object> ipmacenum=prop.keys();
                String s=null;
                while(ipmacenum.hasMoreElements())
                {
                     s=ipmacenum.nextElement().toString();

     macipList.add(s.trim()+":"+prop.getProperty(s).trim(
));
                }
           }
           catch(Exception e)
           {
                System.out.println(e.toString());
           }
           return macipList;
     }
}
```

////End of the Program: IPDemo.java //

Interaction.java :

/// Beginning of Program: DistServer.java ///

```
import java.io.*;
import java.rmi.*;
import java.rmi.registry.*;

public class DistServer
{

    public static void main(String args[]) throws
IOException
    {

          DistServer ods = new DistServer();
          String ip=null;
          int port=0;

        if(args.length<2)
        {
          BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
          char ans = 'n';

          do {
               System.out.println("Are you sure you want
to start RMI Server on local machine : y/n ");
               ans = (char)br.read();
               if (ans == 'y') {
                    ip = "127.0.0.1";
                    port = 1099;
                    ods.startRMIServer();
                    }
               else if (ans == 'n')
                    { System.out.println("Usage: java
DistServer <ip> <port>");
                    return;
                    }

          }while( args.length >= 2 );

        }


          if(args.length >= 2)
          {

        try
```

```
        {
          ip = args[0];
            port=Integer.parseInt(args[1]);
        }
        catch(Exception e)
        {
            System.out.println("port number should be int
number");
            return;
        }

               ods.startRMIServer(ip, port);

          }

    } //End of main()


     public void startRMIServer(String ip, int port) {

     System.setProperty("java.rmi.server.hostname", ip);
     //or java -Djava.rmi.server.hostname=192.168.1.35
server 192.168.1.35

        try
        {
            LocateRegistry.createRegistry(port);
            System.out.println("RMI Registry is
created.");
        }
        catch(RemoteException e)
        {
            System.out.println("RMI Registry has already
been started.");
        }
        try
        {
            DistImpl di=new DistImpl();
            String url="//" + ip + ":" + port +
"/DistServer";
            Naming.rebind(url, di);
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }

     } //End of startRMIServer(String ip, int port)

} //End of class
```

/// End of Program: DistServer.java ///

Interaction.java :

/// Beginning of Program: Interaction.java ///

```
/*

Code developed using Programming Language: Java


Program to do database related operations like
Insert/Update/Delete/ Query to various tables like User,
Species, Run, Reads. The code is used to perform queries
on basis of various parameters.


This code also has the logic to extract several fields
from a raw data file which contains DNA sequence Reads
which is in a FASTA format. The logic defined, makes use
of org.biojavax package to read the FASTA file and
java.util.regex to use regular expression features to
extract features from a give Read, which is in a string
form.

Org.hibernate package is use for database related
operations and com.myapp.struts package is used to
develop classes for User Interface.

*/


package hibernateclasses;


import com.myapp.struts.LoginForm;
```

```
import com.myapp.struts.SelectQueryForm;

import com.myapp.struts.SqlQueryForm;

import java.io.BufferedReader;

import java.io.ByteArrayOutputStream;

import java.io.File;

import java.io.FileReader;

import java.math.BigDecimal;

import java.util.*;

import org.biojavax.SimpleNamespace;

import org.biojavax.bio.seq.RichSequence;

import org.biojavax.bio.seq.RichSequenceIterator;

import org.hibernate.*;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

import org.hibernate.transform.*;

import hibernateclasses.User;


public class Interaction {

static          SessionFactory          factory          =
HibernateUtilMysql.getSessionFactory();

Session session = factory.openSession();
```

```
Transaction tx = null;

private List aauspecies,aaurun,searchbylength,
searchbylengthrange, role;


    public List getRole() {

        return role;

    }


    public void setRole(List role) {

        this.role = role;

    }

     public List getSearchbylengthrange() {

        return searchbylengthrange;

    }

    public void setSearchbylengthrange(List
searchbylengthrange) {

        this.searchbylengthrange = searchbylengthrange;

    }


    public List getAaurun() {

        return aaurun;

    }
```

```
public void setAaurun(List aaurun) {
    this.aaurun = aaurun;
}

public List getSearchbylength() {
    return searchbylength;
}

public void setSearchbylength(List searchbylength) {
    this.searchbylength = searchbylength;
}

public List getAauspecies() {
    return aauspecies;
}

public void setAauspecies(List aauspecies) {
    this.aauspecies = aauspecies;
}
```

```
    public int searchUser(String username,String
password){
          List l = Collections.EMPTY_LIST;
     Query q = session.createQuery("from User");
          l = q.list();
        if(!l.isEmpty()){
            for(int i=0;i<l.size();i++){
            User u = (User)l.get(i);
            if(username.equals(u.getUsername())){
                if(password.equals(u.getPassword())){
                    return u.getRole().getRole();
            }
        }
      }
      }
        return -1;

  }

    public void insertUser(User u){
        tx=session.beginTransaction();
        session.save(u);
```

```
        tx.commit();
    }

    public List getAllRole(){
        Query q1 = session.createQuery("from Role where
role>0");
        this.setRole(q1.list());
        return q1.list();
    }

    public String getRoleType(int role){
        Role r = (Role)session.get(Role.class,role);
        return r.getType();
    }
    public User getUserData(String uname){
        try{
        Query q1 = session.createQuery("from User u where
u.username like "+"'"+uname+"'");
        User u = (User)q1.list().get(0);
        return u;
        }
        catch(Exception e){
```

```
        return null;
    }
}
public boolean approveUser(int role,String username){
    tx=session.beginTransaction();
    User u = (User)session.get(User.class,username);
    u.setRole(new Role(role));
    session.update(u);
    try{
        tx.commit();
        return true;
    }
    catch(Exception e){e.printStackTrace();
        return false;
    }
    }

public boolean rejectUser(String username){
    tx=session.beginTransaction();
    User u = (User)session.get(User.class,username);
    session.delete(u);
    try{
```

```
            tx.commit();
            return true;
        }
        catch(Exception e){e.printStackTrace();
            return false;
        }
    }

    public List getNewUser(){
        Query q1 = session.createQuery("from User where
role<0");
        return q1.list();
    }

    public void insertInToAauRead(Read[] oread){
        String filename = oread[0].read_filename;// to
get the file name only once
        int spicesid = oread[0].species_id;// to get the
speciesid
        int runid = oread[0].run_id;// to get the runid
        int id = oread[0].id;
        AauRunId oarunid = new AauRunId(speciesid,
runid);
```

```
        AauSpecies oaspecies = new AauSpecies(speciesid);
        AauRun oarun = new AauRun(oarunid, oaspecies);
        AauReadId oareadid;
        AauRead oaread;
        for(int i=0;i<(oread.length-1);i++){
        int x=i+1;
        oareadid = new AauReadId(spicesid,
runid,filename,x);
        tx = session.beginTransaction();

         oaread= new AauRead();
         oaread.setId(oareadid);
         oaread.setAauRun(oarun);
         oaread.setId_1(id);
         oaread.setReadName(oread[i].read_name);
         oaread.setReadRank(oread[i].read_rank);
         oaread.setReadX(new
BigDecimal(Float.toString(oread[i].read_x)));
         oaread.setReadY(new
BigDecimal(Float.toString(oread[i].read_y)));
         oaread.setReadLength(oread[i].read_length);
         oaread.setReadSequence(oread[i].read_sequence);
```

```
            session.save(oaread);
            tx.commit();
            session.clear();
          }
      }

    public Read[] readFromFastaFile(String filepath,
String filename, int id, Int runid, int speciesid){
          File of = new File(filepath+"/"+filename);
          BufferedReader br=null;
          Read temp_read_array[] = null;
          try{
          br = new BufferedReader(new FileReader(of));
          }
          catch(Exception e){e.printStackTrace();}
          try {
      SimpleNamespace myreads = new
SimpleNamespace("myreads");
      RichSequenceIterator myFasta =
      RichSequence.IOTools.readFastaDNA(br, myreads);
          int i=0;
          while (myFasta.hasNext()) {
```

```
    ByteArrayOutputStream baos = new
ByteArrayOutputStream();


    ///Extract Sequence from Fasta File ///

        RichSequence.IOTools.writeFasta(baos, myFasta,
myreads);

        String seqn = baos.toString();

        StringTokenizer stk2 = new
StringTokenizer(seqn, ">");

        int totnumofreads = stk2.countTokens();

        StringTokenizer stk = new
StringTokenizer(seqn);

        String tk[ ] = new String[totnumofreads];

        Read read_array[] = new Read[totnumofreads +
1];

int readcount = 0;

        int readindex = 0;

while (stk.hasMoreTokens()) {

        tk[readcount] = stk.nextToken(">");

        StringTokenizer stk1 = new
StringTokenizer(tk[readcount], "\n");

        int totnumoflines = stk1.countTokens();

        String tk1[ ] = new String[totnumoflines];

        int linecount = 0;
```

```
            StringBuffer sbhdr = new StringBuffer();
            StringBuffer sbseqn = new StringBuffer(" ");

while (stk1.hasMoreTokens())
                {
                tk1[linecount] = stk1.nextToken("\n");
                if (linecount == 0) {
                     sbhdr = sbhdr.append(tk1[linecount]);
                     }
                else if (linecount != 0) {
                     sbseqn                                    =
sbseqn.append(tk1[linecount]);
                     }
                ++linecount;
                }
System.out.println();
String readname = sbhdr.substring(0, sbhdr.indexOf(" ")
);
readname = readname.substring(readname.indexOf("|") + 1,
readname.lastIndexOf(".") );
                String                readrank                =
sbhdr.substring(sbhdr.indexOf("rank"), sbhdr.indexOf(" ",
sbhdr.indexOf("rank")) );
```

```
                readrank                                        =
readrank.substring(readrank.indexOf("=") + 1);

                String                  readx                   =
sbhdr.substring(sbhdr.indexOf("x="),   sbhdr.indexOf("   ",
sbhdr.indexOf("x=")) );

                readx = readx.substring(readx.indexOf("=")
+ 1);

                String                  ready                   =
sbhdr.substring(sbhdr.indexOf("y="),   sbhdr.indexOf("   ",
sbhdr.indexOf("y=")) );

                ready = ready.substring(ready.indexOf("=")
+ 1);

                String              readlength                  =
sbhdr.substring(sbhdr.indexOf("length"),    sbhdr.length()
);

                readlength                                      =
readlength.substring(readlength.indexOf("=")      +      1,
readlength.length() );


                int rank = Integer.parseInt(readrank);

                float               len_temp                    =
Float.parseFloat(readlength);

                int len = (int) len_temp;

                float x = Float.parseFloat(readx);

                float y = Float.parseFloat(ready);

                sbseqn = sbseqn.deleteCharAt(0);
```

```
//System.out.println("***********id is "+id);

//System.out.println("***********sp is "+speciesid);

//System.out.println("***********run is "+runid);

//System.out.println("***********file is "+filename);

//System.out.println("***********inc is "+readcount+1);

                        Read      tempread      =      new
Read(id,speciesid,runid,   filename,   readcount   +   1,
readname,         rank,         x,         y,         len,
sbseqn.toString().toUpperCase() );


//System.out.println("***********id is "+tempread.id);

//System.out.println("tempread  "  +  tempread.read_id  +
tempread.read_name      +      tempread.read_length      +
tempread.read_sequence );


            read_array[readcount] = tempread;

//
     System.out.println(read_array[readcount].read_id    +
read_array[readcount].read_name                           +
read_array[readcount].read_length                         +
read_array[readcount].read_sequence );


            ++readcount;

          }

          temp_read_array = read_array;
```

```
        }
           br.close();
        }
        catch (NullPointerException ne) {
             System.out.println("Null Pointer " );
             ne.printStackTrace();
        }
        catch (Exception e) {
             System.out.println(e);
        }
        finally {
        }
//of.deleteOnExit();
 //     System.out.println("$$$$$$$deletion     of     file
"+of.exists()+of.delete()+of.exists());
           return(temp_read_array);
}

    public List getSpeciesId(){
        //tx=session.beginTransaction();
        //List xyz = null;
        //aauspecies.clear();
```

```
        Query q1 = session.createQuery("from
AauSpecies");
        //int siz=q1.list().size();
        this.setAauspecies(q1.list());
        //System.out.println("size of list"+siz);
        //int[] speciesid=new int[siz];
        /* for(int i=0;i<siz;i++){
                                AauSpecies oasp =
(AauSpecies)q1.list().get(i);
                                xyz.add(oasp);

//speciesid[i]=oasp.getSpeciesId();
                                //aauspecies.add(oasp);
                                //
System.out.println("**********");
                                }
         //return speciesid;*/
         return q1.list();
    }

    public int[] getRunId(int sid){
        Query q;
        //tx=session.beginTransaction();
```

```
        if(sid==0){
            q=session.createQuery("from AauRun");
        }
        else{
        q = session.createQuery("from AauRun a where
a.id.speciesId="+sid);
        }
        int siz=q.list().size();
        this.setAaurun(q.list());
        int[] runid=new int[siz];
         for(int i=0;i<siz;i++){
                                AauRun oarunid =
(AauRun)q.list().get(i);

runid[i]=oarunid.getId().getRunId();
                                }
          return runid;
    }

    public String getSpeciesName(int spid){
        Query q1 = session.createQuery("from AauSpecies
where speciesId="+spid);
```

```
        AauSpecies oaspecies =
(AauSpecies)q1.list().get(0);
        return oaspecies.getSpeciesName();
    }

    public boolean insertInToAauSpecies(int spid,String
spname){
        tx=session.beginTransaction();
        AauSpecies oaaus = new AauSpecies(spid);
        oaaus.setSpeciesName(spname);
        session.save(oaaus);
        try{
            tx.commit();
            return true;
        }
        catch(Exception e){e.printStackTrace();
            return false;
        }
    }

    public boolean updateAauSpecies(int spid,String
spname){
        tx=session.beginTransaction();
```

```
    AauSpecies oaaus = new AauSpecies(spid);
    oaaus.setSpeciesName(spname);
    session.update(oaaus);
    try{
        tx.commit();
        return true;
    }
    catch(Exception e){e.printStackTrace();
        return false;
    }
}
    public boolean deleteAauSpecies(int spid){
    tx=session.beginTransaction();
    AauSpecies oaaus = new AauSpecies(spid);
    session.delete(oaaus);
    try{
        tx.commit();
        return true;
    }
    catch(Exception e){e.printStackTrace();
        return false;
    }
```

```
    }

    public boolean insertAauRun(AauRunId oid,Date
dtofrun){
        tx=session.beginTransaction();
        AauRun oarun = new AauRun();
        oarun.setId(oid);
        oarun.setDtOfRun(dtofrun);
        session.save(oarun);
        try{
            tx.commit();
            return true;
        }
        catch(Exception e){e.printStackTrace();
            return false;
        }
    }

    public boolean updateAauRun(AauRunId oid,Date d){
        tx=session.beginTransaction();
        AauRun oarun = new AauRun();
        oarun.setId(oid);
```

```
    oarun.setDtOfRun(d);
    System.out.println("date in update method "+d);
    session.saveOrUpdate(oarun);
    try{
        tx.commit();
        return true;
    }
    catch(Exception e){e.printStackTrace();
        return false;
    }
    }

public boolean deleteAauRun(AauRunId runid){
    tx=session.beginTransaction();
    AauRun oaaurun = new AauRun();
    oaaurun.setId(runid);
    session.delete(oaaurun);
    //session.update(oaaus);
    try{
        tx.commit();
        return true;
    }
```

```
        catch(Exception e){e.printStackTrace();
            return false;
        }
    }

    public List searchReadsByLength(String sel,int
len,int speciesid){
        Query oq = null;
        List x=Collections.EMPTY_LIST;

        if(len==0){
            if(speciesid==0){
               return x;
            }
            else{
                oq=session.createQuery("from AauRead a
where a.id.speciesId="+speciesid);
            }
        }
        else{
               if(speciesid==0){
                       if("exact".equals(sel)){
```

```
                        oq=session.createQuery("from
AauRead where readLength="+len);

                    }

                    else if("andless".equals(sel)){

                        oq=session.createQuery("from
AauRead where readLength<="+len);

                    }

                    else{

                        oq=session.createQuery("from
AauRead where readLength>="+len);

                    }

                }

                else{

                    if("exact".equals(sel)){

                        oq=session.createQuery("from
AauRead      a      where      a.readLength="+len+"and
a.id.speciesId="+speciesid);

                    }

                    else if("andless".equals(sel)){

                        oq=session.createQuery("from
AauRead      a      where      readLength<="+len+"and
a.id.speciesId="+speciesid);

                    }

                    else{
```

```
                        oq=session.createQuery("from
AauRead      a      where      readLength>="+len+"and
a.id.speciesId="+speciesid);

                    }

                }

            }


             x= oq.list();

            this.setSearchbylength(x);

            //new method().WriteOutputToFile(x);

            return x;

           }

    public  List  searchReadsByRunId(int  speciesid,int
runid){

            Query oq = null;

            List x=Collections.EMPTY_LIST;


            oq=session.createQuery("from  AauRead  a  where
a.id.speciesId="+speciesid+" and a.id.runId="+runid);

            x=oq.list();

            return x;

    }
```

```
    public List searchReadsByLengthRange(int greater,int
less,int speciesid){
        Query qq = null;
        List r=Collections.EMPTY_LIST;
        if(less==0){
            if(speciesid==0){
                //x.clear();
               return r;
            }
            else{
                qq=session.createQuery("from AauRead a
where a.id.speciesId="+speciesid);
            }
        }
        else{
            if(speciesid==0){
                   qq = session.createQuery("from
AauRead where readLength>="+greater+"and
readLength<="+less);
            }
            else{
                qq = session.createQuery("from AauRead a
where a.readLength>="+greater+"and
a.readLength<="+less+"and a.id.speciesId="+speciesid);
```

```
    }

    public List sqlQueryAll(String query){

        List l = Collections.EMPTY_LIST;

        Query sql = session.createSQLQuery(query)

                .addScalar("id",Hibernate.INTEGER)

.addScalar("species_id",Hibernate.INTEGER)

                .addScalar("run_id",Hibernate.INTEGER)

                .addScalar("file_name",Hibernate.STRING)

                .addScalar("read_id",Hibernate.INTEGER)

                .addScalar("read_name",Hibernate.STRING)

.addScalar("read_rank",Hibernate.INTEGER)

.addScalar("read_x",Hibernate.BIG_DECIMAL)

.addScalar("read_y",Hibernate.BIG_DECIMAL)

.addScalar("read_length",Hibernate.INTEGER)

.addScalar("read_sequence",Hibernate.STRING)

.setResultTransformer(Transformers.aliasToBean(SqlAauRead
.class));
```

```
        //AliasToBeanResultTransformer    xyz    =    new
AliasToBeanResultTransformer(AauRead.class);

        try{

        l = sql.list();

        }

        catch(Exception e) {

            System.out.println("hello");

            new              SqlQueryForm().setSize("<span
style='color:red'>  No  record  found  for  your  query
</span>");

        }

        //List m = xyz.transformList(l);

        //String [] a =sql.getReturnAliases();

         return l;

       }

     public void selectQuery(List a,String query){

         SQLQuery qry1=null;

         if(a.contains("read_sequence")){

             System.out.println("yes");

             qry1                                          =
session.createSQLQuery(query).addScalar("read_sequence",H
ibernate.STRING);

             a.remove("read_sequence");
```

```
            List b=new ArrayList();
            try{
            b = qry1.list();
            }
            catch(Exception e){
            }
String   res   =   "<br><table   border='1'><tr><th>READ
SEQUENCE</th>";
            String rs="";
            String file;
            for(Object obj:b) {

rs=rs+"<tr><td>"+obj.toString()+"</td></tr>";
           }
            res=res+rs+"</tr></table>";
               new SelectQueryForm().setResult(res);
         }
          else{
         SQLQuery qry = session.createSQLQuery(query);

qry.setResultTransformer(Criteria.ALIAS_TO_ENTITY_MAP);

        List l = qry.list();
```

```
        System.out.println("executing***********");
        int arraysize = a.size();
        String result = "";
        String r1="";
        String value="";
        String value1="";
        for (int i=0;i<a.size();i++) {
            r1 = r1+ "<td>"+a.get(i)+"</td>";
        }

       for (Object obj:l) {
            value1 = value1+"<tr>";
            Map row = (Map)obj;
            for(int i=0;i<a.size();i++) {
                value = value+
"<td>"+row.get(a.get(i))+"</td>";
            }
            value1 = value1+value+"</tr>";
            value="";
        }
        result = "<table
border='1'><tr>"+r1+"</tr>"+value1+"</table>";
```

```
            new SelectQueryForm().setResult(result);
        }
        }
}
```

////////////////End of Program : Interaction.java////////////////

SearchByLengthAction.java

//// Beginning of the Program: SearchByLengthAction.java ///

```
/*

Code developed using Programming Language: Java


Program to fire the query for searching the DNA sequence
Reads from the database, on the basis of length. The
logic to generate the output of the results in form of
webpage using servlets and the facility to download the
result file, is also defined in this program. */


package com.myapp.struts;


import hibernateclasses.AauRead;

import hibernateclasses.Interaction;

import java.io.File;

import java.io.FileInputStream;

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletOutputStream;

import javax.servlet.http.HttpServletRequest;
```

```
import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import org.apache.struts.action.ActionForm;

import org.apache.struts.action.ActionForward;

import org.apache.struts.action.ActionMapping;


public class SearchByLengthAction extends
org.apache.struts.action.Action {

    private static final String SUCCESS = "success";

    static File download;

    /**

     * This is the action called from the Struts
framework.

     * @param mapping The ActionMapping used to select
this instance.

     * @param form The optional ActionForm bean for this
request.

     * @param request The HTTP Request we are processing.

     * @param response The HTTP Response we are
processing.

     * @throws java.lang.Exception

     * @return

     */
```

```
    @Override
    public ActionForward execute(ActionMapping mapping,
ActionForm form,
            HttpServletRequest                    request,
HttpServletResponse response)
            throws Exception {


        HttpSession ohs = request.getSession(false);
        if(ohs.getAttribute("name") ==null){
            return mapping.findForward("nosession");
         }


        Interaction oi = new Interaction();
        method om = new method();
        om.populateDropDownSpeciesid("searchbylength");
        if(request.getContentLength()<0){
             return mapping.findForward(SUCCESS);
        }


        SearchByLengthForm          formBean          =
(SearchByLengthForm)form;
        int speciesid = formBean.getSpeciesid();
```

```
        formBean.setSpeciesid(0);//this is done to get
the select.. option after submiting the form=>pankaj

        om.populateDropDownSpeciesid("searchbylength");

        int flag = formBean.getFlag();

        String resulttype = formBean.getResulttype();

                List m=Collections.EMPTY_LIST;

                if(flag==1){

                    String select = formBean.getSelect();

                    int length = formBean.getLength();

                    //System.out.println("value          of
LE"+length+select);

                    m   =   oi.searchReadsByLength(select,
length,speciesid);


System.out.println("value"+m.size()+speciesid);

                    if(m.isEmpty()){

                    if(m.isEmpty()&&(length>0          ||
speciesid>0)){

formBean.setError("<span   style='color:red'>   No   record
found for your Query.... </span>");

                    }

                    else{

formBean.setError("<span style='color:red'> You must fill
atleast 1 of the above field  </span>");
```

```
                    }

                    }

                }

                else if(flag==2){

                    int greater = formBean.getGreater();

                    int less = formBean.getLess();


m=oi.searchReadsByLengthRange(greater,less,speciesid);

                    if(m.isEmpty()){

                        if(m.isEmpty()&&((greater>-1    &&
less>0) || speciesid>0)){

formBean.setError("<span  style='color:red'>  No  record
found for your Query.... </span>");

                        }

                        else{

formBean.setError("<span style='color:red'> You must fill
atleast 1 of the above field  </span>");

                        }

                    }

                 }

                else if(flag==3||flag==0){

m = oi.searchReadsByLength("nothing",0,speciesid);

                    if(m.isEmpty()){
```

```
                    if(m.isEmpty() && speciesid>0){
formBean.setError("<span style='color:red'> No record
found for your Query.... </span>");
                    }
                    else{
                        formBean.setError("<span
style='color:red'> You must fill species id </span>");
                    }
                    }
                }
                else if(flag==4){
                    int runid = formBean.getRunid();
                    m=oi.searchReadsByRunId(speciesid,
runid);
                    if(m.isEmpty()){
                    if(m.isEmpty() && speciesid>0){
formBean.setError("<span style='color:red'> No record
found for your Query.... </span>");
                    }
                    else{
formBean.setError("<span style='color:red'> You must fill
species id </span>");
                    }
```

```
                }

            }

            if(!m.isEmpty()){formBean.setSize("<span
style='color:red'> Total Number of Record Found :
"+m.size()+"</span>");}


            if("html".equals(resulttype)){

                if(!m.isEmpty()){

                ArrayList oal = new
ArrayList<AauRead>();

                  for(int i=0;i<m.size();i++){

                      AauRead oar =
(AauRead)m.get(i);

                      oal.add(oar);

                    }

                    formBean.setResults(oal);

                }

            }


            else if("text".equals(resulttype)){

                if(!m.isEmpty()){

                    method omth = new method();
```

```
                    String pathname =
getServlet().getServletContext().getRealPath("/")+"result
download";
File of = omth.writeAndDownloadFileTextFasta(m,
pathname,resulttype,"aauread");
                        omth.downloadFile(of, response);
                    }
                }
                else if("fasta".equals(resulttype)){
                    if(!m.isEmpty()){
                        method omth = new method();
                        String pathname =
getServlet().getServletContext().getRealPath("/")+"result
download";
File of = omth.writeAndDownloadFileTextFasta(m,
pathname,resulttype,"aauread");
                        omth.downloadFile(of, response);
                    }
                }
                else if("webtext".equals(resulttype)){
                    if(!m.isEmpty()){
                        ArrayList oal = new
ArrayList<AauRead>();
                        for(int i=0;i<m.size();i++){
```

```
                    AauRead oar =
(AauRead)m.get(i);
                    oal.add(oar);
                }
                formBean.setResults(oal);

                method omth = new method();
                String pathname =
getServlet().getServletContext().getRealPath("/")+"result
download";
                String type="text";
                File of =
omth.writeAndDownloadFileTextFasta(m,
pathname,type,"aauread");
                formBean.setDownloadlink("<a
href='filedownload.jsp'  onclick='downloadfile()'>Download
Result as Text</a>");
            }
        }
        else if("webfasta".equals(resulttype)){
            if(!m.isEmpty()){
                ArrayList oal = new
ArrayList<AauRead>();
                for(int i=0;i<m.size();i++){
```

```
                    AauRead          oar          =
(AauRead)m.get(i);

                    oal.add(oar);

                }

                formBean.setResults(oal);


                method omth = new method();

String   pathname   =   getServlet().getServletContext().
getRealPath("/")+"result download";

String type="fasta";

File of = omth.writeAndDownloadFileTextFasta(m, pathname,
type,"aauread");

formBean.setDownloadlink("<a       href='filedownload.jsp'
onclick   =   'downloadfile()'>   Download   Result   as
Fasta</a>");

                }

            }

        return mapping.findForward(SUCCESS);

    }

}
```

//// End of the Program: SearchByLengthAction.java ///