

DESCRIPCION GENERAL DE LOS ARCHIVOS:

PREVIO A MODIFICACIONES

SERVER:

Adapter.java
 Address.java
 serverParam.java
 serverIni.java

Adapter.java

Cuatro atributos de clase:

```
private String      file_xml      = null;
private String      file_class    = null;
private ParamsObject po           = null;
private String      worker_ip     = null;
```

El constructor tiene dos parámetros: String fx, String fc

```
file_xml      = fx;
file_class    = fc;
String client_name = null;
```

Analiza el fichero XML, sus parámetros en cuanto a la ip del worker y al cliente, y los envía por socket.

Address.java

Clase que hereda de object, crea un nuevo “address”, con un método initialize lee de un fichero “workingconfig.txt” y tras analizarlo a partir de tokens separados por # hace un map.put.

Ejemplo de fichero:

```
-----
ssea  tracer0.lcc.uma.es
sa    tracer0.lcc.uma.es
es    tracer0.lcc.uma.es
```

serverParam.java

Servidor de parámetros:

```
/*
 * serverParam.java
 *
 * Created on 8 de noviembre de 2001, 18:11
 *
 * This class executed a program servant by means of a socket
 * returning the results of the execution of a Genetic Algorithm
 * implemented in the class Exe
 */

package project.server;

import project.util.*;
import java.util.*;
import java.net.*;
import java.io.*;
```

```

/**
 *
 * @author Jose manuel Garcia Nieto
 * @version 0
 */

// This class is used instance a task for the servant that takes charge
// to listen the port
public class serverParam extends Thread {

    /** Creates new ServidorParametro */
    public serverParam() {
        // We begin the task
        start();
    }

    public void run(){
        try{
            WebStream ws = new WebStream(1141);
            System.out.println("Server listen port 80");
            while(true){
                ws.accept();
                new conectParam(ws);
            }

        }catch(Exception e){
            e.printStackTrace();
        }
    }
}

// This class is used a task that it assists a call to rush
// received through from the specified port
class conectParam extends Thread {
    WebStream wstream;
    String my ip = "150.214.214.17";
    int com_number = 0;

    conectParam(WebStream ws){
        System.out.println("Recivied a call ");
        wstream = ws;
        setPriority(NORM_PRIORITY-1);
        start();
    }

    // It throws the thread of attention to the port
    public void run(){
        System.out.println("Thrown the thread of attention ");
        String xml_file = "optimization_algorithm.xml";
        String class_file = "problem.class";
        int [] len = new int[1];
        String client_name = null;
        String type_select = null;
        String language = null;
        ParamsObject po = null;

        try{
            wstream.init();

            // get the client name in order to register a new client directory
            len = (int[])wstream.get();
            com_number = len[0];
            client_name = wstream.get string();
            System.out.println("Client Name : "+client_name);
            // Register the name of the client
            Register reg = new Register();
            reg.setRoot("register_dir");
            reg.registration(client_name);
            xml file = reg.getPath()+xml file;
            class file = reg.getPath()+class file;
            //xml file = "/C:/jwsdp-1_0-ea2/webapps/ROOT/users/nieto/" + xml_file;
            if (com_number == 1){

```



```
// Count the number of 1's in the string
private double ONEMAX(Individual indiv)
{
    double f=0.0;
    for(int i=0; i<CL; i++)
        if(indiv.get_allele(i)==1)
            f=f+1.0;
    indiv.set_fitness(f);
    return f;
}
}
```

serverni.java

Método MAIN que genera el objeto window con la imagen, etc..

WORKER:

Worker.java

NetworkClassLoader.java

workerIni.java: método MAIN que genera el objeto window con la imagen, etc..

wrapper_cea.java

wrapper_dea.java (parecido al anterior)

wrapper_ee.java

wrapper_es.java: rastrigin problem

wrapper_sa.java: oneMax problem

wrapper_simpleea.java

wrapper_ssea.java: llama con el command a javac para ejecutarse

wrapper_sseam2.java: módulo2

wrapper.java: llama a un ejecutable pasándole dos parámetros.

Worker.java

Clase Worker y WorkerConnect.

Atributos:

```
WebStream ws;
String class_file = "";
String type_of_algor;
String client_name = null;
String cn = null;
String langu;
Wrapper wp = null;
```

La clase toma el fichero "xml_received.xml" parsea los argumentos y hace un wrapper en función del tipo de algoritmo.

```
if ((type_of_algor.equalsIgnoreCase("ssea")) &&
(langu.equalsIgnoreCase("Java"))){
    wp = new Wrapper_ssea(xml_file);
    wp.setClassFile(class_file);
    wp.compile();
```

Finalmente ejecuta el algoritmo y los resultados los envía en el mismo fichero.

```
wp.run();

// Put the Xml file in the WebStream to sent it at server
ws.put_file(xml_file);
//ws.put("fileresults.txt");

if (ws.probe()) ws.flush();
ws.finalize();
```

Wrapper_cea.java

Atributos:

```

private String xml_file      = null;
private String config_file   = "modula2/fconfig1.txt";
private String results_file  = "modula2/results1.txt";
private String user_path     = null;
private ParamsObject po     = null;
private long    exec_time    = 0;
private boolean ok           = true;

```

La clase wrapper toma el fichero XML realiza la ejecución y emite los resultados.

```

public Wrapper_cea(String xf) {
    xml_file = xf;
}

public void run() {

    this.execute(config_file, results_file);
    this.getXmlResult(results_file, xml_file);
}

private void getConfigurationFile(String xmlf, String conf) {
    DataOutputStream dos;
    DataInputStream  dis;
    FileOutputStream fos;

    XmlParser xp = new XmlParser(xmlf);
    po = xp.getParamsObject();
    try {
        String line = null;
        int ln = 1;
        dos = new DataOutputStream((new FileOutputStream("modula2/cssga.def")));
        dis = new DataInputStream((new FileInputStream("cssga.def")));
        while (ln < 96) {
            line = dis.readLine();
            if (ln == 12) {
                dos.writeBytes("\tGENE_LENGTH =
"+po.get("length_of_gene").trim()+";    (*Length of every gene-vble*)\n");
                dos.writeBytes("\tGENE_NUMBER =
"+po.get("number_of_genes").trim()+";    (*Number of genes per chromosome*)\n");
                dos.writeBytes("\tCHROM_LENGTH = GENE_NUMBER * GENE_LENGTH;
(*Length of the chromosome*)\n");
                dos.writeBytes("\tWIDTH =
"+po.get("population_width").trim()+";    (*Width of the grid*)\n");
                dos.writeBytes("\tHEIGHT =
"+po.get("population_height").trim()+";    (*Height of the grid*)\n");
            } else if ((ln < 12) || (ln > 16))
                dos.writeBytes(line+"\n");

            ln++;
        }

        dos.close();
        dis.close();

        // Update the "ssga.def" file with news params
        line = null;
        ln = 1;
        dos = new DataOutputStream((new FileOutputStream("modula2/cssga.mod")));
        dis = new DataInputStream((new FileInputStream("cssga.mod")));
        while (ln < 492) {
            line = dis.readLine();
            if (ln == 14) {
                dos.writeBytes("\tNEIGH_SIZE = "+po.get("neighb").trim()+";
(*Number of neighbors*)\n");
            } else
                dos.writeBytes(line+"\n");
            ln++;
        }

        dos.close();
        dis.close();
    }
}

```

```

StringBuffer sb = new StringBuffer();
sb.append(po.get("crossover_prob").trim()+"\n");
sb.append(po.get("mutation_prob").trim()+"\n");
sb.append(po.get("replacement","type").trim()+"\n");
sb.append(po.get("Max n evals").trim()+"\n");
sb.append(po.get("fitness_function").trim()+"\n");

fos = new FileOutputStream(conf);
fos.write(sb.toString().getBytes());
fos.close();

} catch(FileNotFoundException fnfe){
    fnfe.printStackTrace();
} catch(IOException e){
    e.printStackTrace();
}
}

public void compile(){
    Runtime rtcomp = Runtime.getRuntime();
    Process pcomp = null;
    String comand[] = {"compile.bat"};

    try{
        this.getConfiguratiOnFile(xml_file,config_file);
        System.out.println("COMPILATION BEGIN");
        pcomp = rtcomp.exec(comand);
        int i = pcomp.waitFor();
        System.out.println("COMPILATION END");
    } catch(Exception e){
        System.out.println("Error in execution of : "+comand[0]+" "+comand[1]);
        ok = false;
    }
}

private void execute(String conf, String resf){
    Runtime rt = Runtime.getRuntime();
    Process p = null;
    long init_time, end_time;

    String comand[] = {"exe_cfxx.bat"};

    try{
        System.out.println("PROCESS BEGIN");
        init_time = System.currentTimeMillis();
        p = rt.exec(comand);
        int i = p.waitFor();
        end_time = System.currentTimeMillis();
        exec_time = (end_time - init_time)/1000;
        System.out.println("PROCESS END ... TIME "+exec_time );
    } catch(Exception e){
        System.out.println("Error in execution of : "+comand[0]+" "+comand[1]);
    }
}

private void getXmlResult(String resf, String xmlf){
    DataInputStream in;
    String line = null;
    try{
        po.set("language",po.get("language").trim());
        in = new DataInputStream(new FileInputStream(resf));
        line = in.readLine();
        po.set("individual",in.readLine().trim());
        line = in.readLine();
        po.set("chromosome",in.readLine().trim());
        po.set("fitness",in.readLine().trim());
        po.set("computation_time",new Long(exec_time).toString());
        poToFile(xmlf);
        in.close();
    } catch(IOException ioe){
        ioe.printStackTrace();
    } catch(Exception e){

```

```

        e.printStackTrace();
    }
}

public String getConfigName(){return config_file;}
public String getResultsName(){return results_file;}
public String getUserPath(){return user_path;}
public boolean getok(){return ok;}

public void setConfigName(String cn){config_file = cn;}
public void setResultsName(String rn){results_file = rn;}
public void setUserPath(String up){user_path = up;}
}

```

Wrapper_ee.java

```

public class Wrapper_ee extends Wrapper{
    private String xml_file = null;
    private String config_file = "codigoc++/ess.param";
    private String results_file = "es_bc.res";
    private String user_path = null;
    private ParamsObject po = null;
    private long exec_time = 0;

    /** Creates new Wrapper */
    public Wrapper_ee() {}

    public Wrapper_ee(String xf) {
        xml_file = xf;
    }

    public void run(){

        this.execute(config_file,results_file);
        this.getXmlResult(results_file,xml_file);
    }

    //public void stop(){
    //}

    private void getConfigurationFile(String xmlf,String conf){
        DataOutputStream dos;
        DataInputStream dis;
        FileOutputStream fos;

        XmlParser xp = new XmlParser(xmlf);
        po = xp.getParamsObject();
        try{
            String line = null;
            int ln = 1;
            dos = new DataOutputStream((new FileOutputStream("t2.cpp")));
            dis = new DataInputStream((new FileInputStream("codigoc++/t2.cpp")));
            while (ln < 67){
                line = dis.readLine();
                if (ln == 25) {
                    dos.writeBytes("\t#define TAM_POP
"+po.get("population size").trim()+"\n");
                    dos.writeBytes("\t#define N_GENES
"+po.get("number_of_genes").trim()+"\n");
                    dos.writeBytes("\t#define GENE_LENGTH
"+po.get("length of gene").trim()+"\n");
                    dos.writeBytes("\tEVAL NUM "+po.get("max_n_evals").trim()+"\n");
                }else if ((ln < 25) || (ln > 28))
                    dos.writeBytes(line+"\n");

                ln++;
            }

            dos.close();
            dis.close();

```

```

StringBuffer sb = new StringBuffer();
sb.append("[NUMBER_OF_OPERATORS]");
sb.append("5");
sb.append("[OPERATORS]");
if (po.get("selection", "type").trim().equalsIgnoreCase("roulette_wheel"))
    sb.append("RW\n");
else
    sb.append("RANKING\n");
sb.append("RANDOM\n");
if (po.get("recombine", "type").trim().equalsIgnoreCase("ee"))
    sb.append("RECOMBINE_ES 1\n");
else
    sb.append("RECOMBINE_ES 2\n");
sb.append("MUTA_ES "+po.get("mutation_prob").trim()+"\n");
if (po.get("replacement", "type").trim().equalsIgnoreCase("worst"))
    sb.append("REP_LEAST_FIT\n");
else
    sb.append("REP_RAND\n");
sb.append("[OUTPUT_DISPLAY]\n");
sb.append("NOTHING\n");
sb.append("[PROBLEM CONFIG FILE]\n");
sb.append("bc train.net\n");
sb.append("[OUTPUT_FILE]\n");
sb.append("pp.kk\n");

fos = new FileOutputStream(conf);
fos.write(sb.toString().getBytes());
fos.close();

} catch (FileNotFoundException fnfe) {
    fnfe.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
}
}

public void compile() {
    Runtime rtcomp = Runtime.getRuntime();
    Process pcomp = null;
    String comand[] = {"make t2g"};

    try {
        this.getConfigurationFile(xml_file, config_file);
        System.out.println("COMPILATION BEGIN");
        pcomp = rtcomp.exec(comand);
        int i = pcomp.waitFor();
        System.out.println("COMPILATION END");
    } catch (Exception e) {
        System.out.println("Error in execution of : "+comand[0]+" "+comand[1]);
    }
}

private void execute(String conf, String resf) {
    Runtime rt = Runtime.getRuntime();
    Process p = null;
    long init_time, end_time;

    String comand[] = {"/t2"};

    try {
        System.out.println("PROCESS BEGIN");
        init_time = System.currentTimeMillis();
        p = rt.exec(comand);
        int i = p.waitFor();
        end_time = System.currentTimeMillis();
        exec_time = (end_time - init_time)/1000;
        System.out.println("PROCESS END ... TIME "+exec_time);
    } catch (Exception e) {
        System.out.println("Error in execution of : "+comand[0]+" "+comand[1]);
    }
}
}

```

```

private void getXmlResult(String resf, String xmlf){
    DataInputStream in;
    String line = null;
    String token = null;
    try{
        po.set("language",po.get("language").trim());
        in = new DataInputStream(new FileInputStream(resf));
        line = in.readLine();
        StringTokenizer stz = new StringTokenizer(line,"\t");
        token = stz.nextToken();
        token = stz.nextToken();
        token = stz.nextToken();
        po.set("best_fitness",stz.nextToken());
        po.set("worst_fitness",stz.nextToken());
        po.set("avg_fitness",stz.nextToken());
        po.set("computation_time",new Long(exec_time).toString());
        poToFile(xmlf);
        in.close();
    }catch(IOException ioe){
        ioe.printStackTrace();
    }catch(Exception e){
        e.printStackTrace();
    }
}

public String getConfigName(){return config_file;}
public String getResultsName(){return results_file;}
public String getUserPath(){return user_path;}

public void setConfigName(String cn){config_file = cn;}
public void setResultsName(String rn){results_file = rn;}
public void setUserPath(String up){user_path = up;}

```

Wrapper_es.java

```

/*
 * Wrapper_es.java
 *
 * Created on 1 de Abril de 2003, 11:54
 */

package project.worker;

import java.io.*;
import java.lang.*;
import java.util.*;
import project.util.*;
/**
 *
 * @author Jose Manuel Garcia N
 * @version
 */
public class Wrapper_es extends Wrapper{
    private String xml_file = null;
    private String config_file = "../c++/Mallba/rep/ES/";
    private String results_file = "../c++/Mallba/rep/ES/";
    private String user_path = null;
    private ParamsObject po = null;
    private long exec_time = 0;

    /** Creates new Wrapper */
    public Wrapper_es() {}

    public Wrapper_es(String xf) {
        xml_file = xf;
    }

    public void run() {
        this.getConfigurationFile(xml_file,config_file);
        this.execute(config_file,results_file);
        this.getXmlResult(results_file,xml_file);
    }

    //public void stop(){

```

```

//}

private void getConfigurationFile(String xmlf,String conf){
    DataOutputStream dos;
    DataInputStream dis;
    FileOutputStream fos;
    String conf_f;

    XmlParser xp = new XmlParser(xmlf);
    po = xp.getParamsObject();
    try{
        StringBuffer sb = new StringBuffer();
        sb.append("5 // number of independent runs\n");
        sb.append(po.getNumberOfGenerations().trim()+" // number of
generations\n");
        sb.append(po.getPopulationSize().trim()+" // number of
individual\n");
        sb.append(po.getOffspringsSize().trim()+" // size of offsprings
in each generation\n");
        int replace = 1;
        if (po.getReplacementType().trim().equalsIgnoreCase("inclusion"))
            replace = 0; //inclusion (a+b) . insercion (a,b)

        sb.append(replace+" // if replaces parents for offsprings, or
only offsprings may be new parents\n");
        sb.append("0 // display state ?\n");

        sb.append("Selections // selections to apply\n");
        int selection = 1; //tournament default
        if (po.getSelectionType().trim().equalsIgnoreCase("random"))
            selection = 0;
        else if
(po.getSelectionType().trim().equalsIgnoreCase("roulette_wheel"))
            selection = 2;
        else if (po.getSelectionType().trim().equalsIgnoreCase("ranking"))
            selection = 3;
        else if
(po.getSelectionType().trim().equalsIgnoreCase("best_individual"))
            selection = 4;
        else if
(po.getSelectionType().trim().equalsIgnoreCase("worst_individual"))
            selection = 5;
        else // tournament
            selection = 1;
        sb.append(selection+" 2 // selection of parents\n");

        sb.append("Intra-Operators // operators to apply in the
population\n");
        sb.append("0 "+po.getCrossoverProb().trim()+" //
crossover & its probability\n");
        sb.append("1 "+po.getMutationProb().trim()+" 0.0873 0.312 0.334 //
mutation & its probability (change only second parameter == use probability)\n");
        sb.append("Inter-Operators // operators to apply between this
population and others\n");
        sb.append("0 "+po.getMigrationRate().trim()+"
"+po.getMigrationNumber().trim()+" 1 3 5 // operator number, operator rate,
number of individuals, selection of individual to send and replace\n");

        sb.append("LAN-configuration\n");
        sb.append("10 // refresh global state\n");
        sb.append("0 // 0: running in asynchronized mode / 1: running in
synchronized mode\n");
        sb.append("1 // interval of generations to check solutions from
other populations\n");

        if (po.getFitnessFunction().trim().equalsIgnoreCase("rastrigin"))
            conf_f = conf + "ras/ES.cfg";
        else
            conf_f = conf + "sphere/ES.cfg";
        fos = new FileOutputStream(conf_f);
        fos.write(sb.toString().getBytes());
        fos.close();

    }catch(FileNotFoundException fnfe){
        fnfe.printStackTrace();
    }catch(IOException e){

```

```

        e.printStackTrace();
    }
}

public void compile(){
    Runtime rtcomp = Runtime.getRuntime();
    Process pcomp = null;
    String com = null;

    if (po.getFitnessFunction().trim().equalsIgnoreCase("rastrigin"))
        com = "COMPESTRAS";
    else
        com = "COMPESSPHERE";

    String comand[] = {"make", com};

    try{

        System.out.println("COMPILATION BEGIN");
        pcomp = rtcomp.exec(comand);
        int i = pcomp.waitFor();
        System.out.println("COMPILATION END");
    } catch (Exception e){
        System.out.println("Error in execution of : "+comand[0]+" "+comand[1]);
    }
}

private void execute(String conf, String resf){
    Runtime rt = Runtime.getRuntime();
    Process p = null;
    long init_time, end_time;

    String com = null;

    if (po.getFitnessFunction().trim().equalsIgnoreCase("rastrigin"))
        com = "EXEESTRAS";
    else
        com = "EXEESSPHERE";

    String comand[] = {"make", com};

    try{
        System.out.println("PROCESS BEGIN");
        init_time = System.currentTimeMillis();
        p = rt.exec(comand);
        int i = p.waitFor();
        end_time = System.currentTimeMillis();
        exec_time = (end_time - init_time)/1000;
        System.out.println("PROCESS END ... TIME "+exec_time );
    } catch (Exception e){
        System.out.println("Error in execution of : "+comand[0] + comand[1]);
    }
}

private void getXmlResult(String resf, String xmlf){
    DataInputStream in;
    String line = null;
    String token = null;
    String fresults = null;
    try{

        if (po.getFitnessFunction().trim().equalsIgnoreCase("rastrigin"))
            fresults = resf + "ras/res/res.txt";
        else
            fresults = resf + "sphere/res/res.txt";

        po.setLanguage(po.getLanguage().trim());
        in = new DataInputStream(new FileInputStream(fresults));
        line = in.readLine();
        po.setBestFitness(in.readLine().trim());
        po.setWorstFitness(in.readLine().trim());
        line = in.readLine();
        po.setGeneration(in.readLine().trim());
        po.setComputationTime(new Long(exec_time).toString());
        poToFile(xmlf);
        in.close();
    }
}

```

```

    } catch (IOException ioe) {
        ioe.printStackTrace();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

public String getConfigName() {return config_file;}
public String getResultsName() {return results_file;}
public String getUserPath() {return user_path;}

public void setConfigName(String cn) {config_file = cn;}
public void setResultsName(String rn) {results_file = rn;}
public void setUserPath(String up) {user_path = up;}
}

```

Wrapper_sa.java

```

/*
 * Wrapper_cea.java
 *
 * Created on 1 de julio de 2002, 11:54
 */

package project.worker;

import java.io.*;
import java.lang.*;
import java.util.*;
import project.util.*;
/**
 *
 * @author Jose Manuel Garcia N
 * @version
 */
public class Wrapper_sa extends Wrapper {
    private String xml_file = null;
    private String config_file = "../c++/Mallba/rep/SA/";
    private String results_file = "../c++/Mallba/rep/SA/";
    private String user_path = null;
    private ParamsObject po = null;
    private long exec_time = 0;

    /** Creates new Wrapper */
    public Wrapper_sa() {}

    public Wrapper_sa(String xf) {
        xml_file = xf;
    }

    public void run() {
        this.getConfigurationFile(xml_file, config_file);
        this.execute(config_file, results_file);
        this.getXmlResult(results_file, xml_file);
    }

    //public void stop(){
    //}

    private void getConfigurationFile(String xmlf, String conf) {
        DataOutputStream dos;
        DataInputStream dis;
        FileOutputStream fos;
        String conf_f;

        XmlParser xp = new XmlParser(xmlf);
        po = xp.getParamsObject();
        try {
            StringBuffer sb = new StringBuffer();
            sb.append("5 // number of independent runs\n");
            sb.append(po.getMaxNEvals().trim()+" // number of evaluations\n");
        }
    }
}

```

```

        sb.append(po.getMarkovChainLength().trim()+"           // Markov-Chain
Length\n");
        sb.append(po.getTemperatureDecay().trim()+"           // temperature
Decay\n");
        sb.append("0           // display state ?");
        sb.append("LAN-configuration\n");
        sb.append("200           // the global state is updated in this number of
evaluations\n");
        sb.append("0           // 0: asynchronized mode // 1: synchronized
mode\n");
        sb.append("500           // interval of iterations to cooperate ( if 0 no
cooperation)\n");

        conf_f = conf + po.getFitnessFunction().trim() + "/SA.cfg";
        fos = new FileOutputStream(conf_f);
        fos.write(sb.toString().getBytes());
        fos.close();

    } catch (FileNotFoundException fnfe) {
        fnfe.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

public void compile() {
    Runtime rtcomp = Runtime.getRuntime();
    Process pcomp = null;
    String com = null;

    if (po.getFitnessFunction().trim().equalsIgnoreCase("onemax"))
        com = "COMPONEMAX";
    else
        com = "COMPMAXSAT";

    String comand[] = {"make", com};

    try {
        System.out.println("COMPILATION BEGIN");
        pcomp = rtcomp.exec(comand);
        int i = pcomp.waitFor();
        System.out.println("COMPILATION END");
    } catch (Exception e) {
        System.out.println("Error in execution of : "+comand[0]+" "+comand[1]);
    }
}

private void execute(String conf, String resf) {
    Runtime rt = Runtime.getRuntime();
    Process p = null;
    long init_time, end_time;

    String com = null;

    if (po.getFitnessFunction().trim().equalsIgnoreCase("onemax"))
        com = "EXEONEMAX";
    else
        com = "EXEMAXSAT";

    String comand[] = {"make", com};

    try {
        System.out.println("PROCESS BEGIN");
        init_time = System.currentTimeMillis();
        p = rt.exec(comand);
        int i = p.waitFor();
        end_time = System.currentTimeMillis();
        exec_time = (end_time - init_time)/1000;
        System.out.println("PROCESS END ... TIME "+exec_time);
    } catch (Exception e) {
        System.out.println("Error in execution of : "+comand[0] + comand[1]);
    }
}
}

```

```

private void getXmlResult(String resf, String xmlf){
    DataInputStream in;
    String line = null;
    String token = null;
    String fresults = null;
    try{
        results = resf + po.getFitnessFunction().trim() + "/res/results.txt";
        po.setLanguage(po.getLanguage().trim());
        in = new DataInputStream(new FileInputStream(fresults));
        line = in.readLine();
        StringTokenizer stz = new StringTokenizer(line,"$");
        token = stz.nextToken();

        po.setChromosome(stz.nextToken());
        token = stz.nextToken();
        po.setFitness(stz.nextToken());
        po.setComputationTime(new Long(exec_time).toString());
        poToFile(xmlf);
        in.close();
    }catch(IOException ioe){
        ioe.printStackTrace();
    }catch(Exception e){
        e.printStackTrace();
    }
}

public String getConfigName(){return config_file;}
public String getResultsName(){return results_file;}
public String getUserPath(){return user_path;}

public void setConfigName(String cn){config_file = cn;}
public void setResultsName(String rn){results_file = rn;}
public void setUserPath(String up){user_path = up;}
}

```

Wrapper_ssea.java

```

/*
 * Wrapper.java
 *
 * Created on 9 de abril de 2002, 12:18
 */

package project.worker;

import java.io.*;
import java.lang.*;
import project.util.*;
/**
 *
 * @author Jose Manuel García Nieto
 * @version
 */

public class Wrapper_ssea extends Wrapper{
    private String xml_file = null;
    private String config_file = "fconfig.txt";
    private String results_file = "results.txt";
    private String user_path = null;
    private String class_file = null;
    private ParamsObject po = null;
    private long exec_time = 0;
    private String error_comp = "-ERROR-";
    private String error_exe = null;
    private boolean ok = true;

    /** Creates new Wrapper */
    public Wrapper_ssea() {}

    public Wrapper_ssea(String xf) {
        xml_file = xf;
    }
}

```

```

}

public void run() {

    this.execute(config_file, results_file);
    this.getXmlResult(results_file, xml_file);
}

//public void stop(){
//}

private void getConfigurationFile(String xmlf, String conf) {
    FileOutputStream fos;

    XmlParser xp = new XmlParser(xmlf);
    po = xp.getParamsObject();

    StringBuffer sb = new StringBuffer();
    sb.append("#####\n");
    sb.append("## CONFIGUTRATION FILE : Steady Stay Genetic Algorithm    ##\n");
    sb.append("## Use the '#' character at begenning of a line for coments ##\n");
    sb.append("#####\n");
    sb.append("##          Parametres of Algorithm          ##\n");
    sb.append("#####\n");
    sb.append("number_of_genes      ="+po.get("number_of_genes")+" \n");
    sb.append("# \n");
    sb.append("gene length          ="+po.get("length_of_gene")+" \n");
    sb.append("# \n");
    sb.append("population_size      ="+po.get("population_size")+" \n");
    sb.append("# \n");
    sb.append("crossover_prob       ="+po.get("crossover_prob")+" \n");
    sb.append("# \n");
    sb.append("mutation_prob        ="+po.get("mutation_prob")+" \n");
    sb.append("# \n");
    sb.append("max n evals          ="+po.get("max_n_evals")+" \n");
    sb.append("# \n");
    sb.append("fitness_function     ="+po.get("fitness_function").trim()+" \n");
    sb.append("# \n");

    try{
        fos = new FileOutputStream(conf);
        fos.write(sb.toString().getBytes());
        fos.close();
    }catch(IOException e){
        e.printStackTrace();
    }
}

public void compile(){
    Runtime rtcomp = Runtime.getRuntime();
    Process pcomp = null;
    String comand[] = {"javac", "project/ssGA/"+class_file};
    String line = null;
    try{
        System.out.println("FC"+class file);
        this.getConfigurationFile(xml_file, config_file);
        System.out.println("COMPILATION BEGIN");
        pcomp = rtcomp.exec(comand);
        int i = pcomp.waitFor();
        DataInputStream dis = new DataInputStream(pcomp.getErrorStream());

        while ((line = dis.readLine()) != null)
            error_comp = error_comp + line;
        System.out.println("ERR "+error_comp);
        if (!error_comp.equalsIgnoreCase("-ERROR-"))
            ok = false;
        System.out.println("COMPILATION END");
    }catch(IOException ioe){
        error_comp = "Error in execution of : "+comand[0]+comand[1];
        ok = false;
    }catch(Exception e){
        error_comp = "Error in execution of : "+comand[0]+comand[1];
        ok = false;
    }
}
}

```

```

private void execute(String conf, String resf){
    Runtime rt = Runtime.getRuntime();
    Process p = null;
    long init_time, end_time;

    String comand[] = {"java","project.ssGA.Exe",conf,resf};

    try{
        System.out.println("PROCESS BEGIN");
        init_time = System.currentTimeMillis();
        if (ok){
            p = rt.exec(comand);
            int i = p.waitFor();
        }
        end_time = System.currentTimeMillis();
        exec_time = (end_time - init_time)/1000;
        System.out.println("PROCESS END ... TIME "+exec_time );
    }catch(Exception e){
        error_exe = "Error in execution of : "+comand[0]+" "+comand[1];
    }
}

private void getXmlResult(String resf, String xmlf){
    DataInputStream in;

    try{
        po.set("language",po.get("language").trim());
        in = new DataInputStream(new FileInputStream(resf));
        po.set("chromosome",in.readLine());
        po.set("fitness",in.readLine());
        po.set("computation_time",new Long(exec_time).toString());
        po.set("error",error_comp+error_exe);
        poToFile(xmlf);
        in.close();
    }catch(IOException ioe){
        ioe.printStackTrace();
    }catch(Exception e){
        e.printStackTrace();
    }
}

public String getConfigName(){return config_file;}
public String getResultsName(){return results_file;}
public String getUserPath(){return user_path;}
public boolean getok(){return ok;}

public void setConfigName(String cn){config_file = cn;}
public void setClassFile(String cf){class_file = cf;}
public void setResultsName(String rn){results_file = rn;}
public void setUserPath(String up){user_path = up;}
}

```

Wrapper.java

```

*
* Wrapper.java
*
* Created on 1 de julio de 2002, 11:54
*/

package project.worker;

import project.util.*;
import java.io.*;
import java.lang.*;
/**
 *
 * @author Jose Manuel Garcia N
 * @version
 */

```

```

public class Wrapper {
    private String xml_file = null;
    private String config_file = "fconfig.txt";
    private String results_file = "results.txt";
    private String user_path = null;
    private String class_file = null;
    private ParamsObject po = null;
    private boolean ok = true;

    /** Creates new Wrapper */
    public Wrapper() {}

    public Wrapper(String xf) {
        xml_file = xf;
    }

    public void run() {

        this.getConfigurationFile(xml_file, config_file);
        this.execute(config_file, results_file);
        this.getXmlResult(results_file, xml_file);
    }

    //public void stop(){
    //}

    private void getConfigurationFile(String xmlf, String conf) {
        FileOutputStream fos;

        XmlParser xp = new XmlParser(xmlf);
        po = xp.getParamsObject();

        StringBuffer sb = new StringBuffer();
        sb.append("#####\n");
        sb.append("## CONFIGUTRATION FILE : Steady Stay Genetic Algorithm ##\n");
        sb.append("## Use the '#' character at begenning of a line for coments ##\n");
        sb.append("#####\n");
        sb.append("## Parametres of Algorithm ##\n");
        sb.append("#####\n");
        sb.append("number_of_genes ="+po.get("number_of_genes")+" \n");
        sb.append("# \n");
        sb.append("gene_length ="+po.get("length_of_gene")+" \n");
        sb.append("# \n");
        sb.append("population_size ="+po.get("population_size")+" \n");
        sb.append("# \n");
        sb.append("crossover_prob ="+po.get("crossover_prob")+" \n");
        sb.append("# \n");
        sb.append("mutation_prob ="+po.get("mutation_prob")+" \n");
        sb.append("# \n");
        sb.append("max n evals ="+po.get("max_n_evals")+" \n");
        sb.append("# \n");
        sb.append("fitness_function ="+po.get("fitness_function").trim()+" \n");
        sb.append("# \n");

        try{
            fos = new FileOutputStream(conf);
            fos.write(sb.toString().getBytes());
            fos.close();
        }catch(IOException e){
            e.printStackTrace();
        }
    }

    public void compile(){}

    private void execute(String conf, String resf){
        Runtime rt = Runtime.getRuntime();
        Process p = null;
        String comand[] = {"java", "project.ssGA.Exe", conf, resf};

        try{
            System.out.println("PROCESS BEGIN");
            p = rt.exec(comand);
            int i = p.waitFor();
            System.out.println("PROCESS END");
        }catch(Exception e){
    
```

```

        System.out.println("Error in execution of : "+comand[0]+" "+comand[1]);
        ok = false;
    }
}

private void getXmlResult(String resf, String xmlf){
    DataInputStream in;

    try{
        in = new DataInputStream(new FileInputStream(resf));
        po.set("allele",in.readLine());
        po.set("fitness",in.readLine());
        poToFile(xmlf);
        in.close();
    }catch(IOException ioe){
        ioe.printStackTrace();
    }catch(Exception e){
        e.printStackTrace();
    }
}

public String getConfigName(){return config_file;}
public String getResultsName(){return results_file;}
public String getUserPath(){return user_path;}
public boolean getok(){return ok;}

public void setConfigName(String cn){config_file = cn;}
public void setClassFile(String cf){class_file = cf;}
public void setResultsName(String rn){results_file = rn;}
public void setUserPath(String up){user_path = up;}

}

```

CLIENTE

clientWindow.java

startOpt.java: muestra las opciones iniciales del proyecto, el gráfico, Idioma Inglés y español, etc..

translate.java: se encarga de la traducción de las opciones

visualField.java: sirve para mostrar información al usuario, al estilo de textbox + label.

visualLogin.java: autenticación de los usuarios, lo hace contra el servidor que esté en la ip que escoja el usuario y una vez autenticado extrae datos de "ps_ip.txt" para realizar la conexión.

VisualOpt.java: pantalla anterior a la especificación de parámetros.

Detalle sorprendente:

```
static String pdf_reader_path = "C:/Archivos de programa/Adobe/Acrobat 5.0/Reader/AcroRd32.exe";
```

```
String comand[] = {pdf_reader_path, "project/client/usersman.pdf"};
```

visualParams.java: entorno de parámetros (n°genes, Fitness, Run, examine, etc..)

vp_cea.java: especialización de parámetros respecto al anterior.

vp_dea.java

vp_ee.java

vp_es.java

vp_sa.java

vp_ssea.java

vp_sseam2.java

clientWindow.java

Clase cliente en la que se presentan las ventanas del cliente, se puede modificar textbox, listbox etc..

Métodos:

searchServer()

connectWin()

getServerFile()

actionPerformed() En este método en función del tipo de algoritmo se cargan los combo.

```
public void actionPerformed( ActionEvent evt ) {
    type_select = combotype.getSelectedItemAt().toString();
    List ip_list = (List)map.get(type_select);
    comboip.enable();
    combolan.enable();
    comboip.removeAllItems();
    for (Iterator it = ip_list.iterator();it.hasNext();){
        comboip.addItem(it.next());
    }
    combolan.removeAllItems();
    if (type_select.trim().equalsIgnoreCase("ssea")) {
        combolan.addItem("Java");
        combolan.addItem("Modula 2");
    }else if (type_select.trim().equalsIgnoreCase("cea")) {
        combolan.addItem("Modula 2");
    }else if (type_select.trim().equalsIgnoreCase("dea")) {
        combolan.addItem("Modula 2");
    }else if (type_select.trim().equalsIgnoreCase("simpleea")) {
        combolan.addItem("Java");
    }else if (type_select.trim().equalsIgnoreCase("es")) {
        combolan.addItem("C++");
    }else if (type_select.trim().equalsIgnoreCase("sa")) {
        combolan.addItem("C++");
    }
}
```

```
        }  
        button_con.enable();  
    }  
};
```

getServerFile(): se conecta con la ip de la máquina escogida, envía la solicitud xml y espera la respuesta.

Posee otros métodos relacionados con aspectos gráficos que carecen de importancia.

PRIMARY_SERVER

Primary_serverIri.java: encargado del main, de mostrar la ventana y arrancar un systemserver.
systemServer.java

systemServer.java

```

/*
 * serverParam.java
 *
 * Created on 8 de noviembre de 2001, 18:11
 *
 * This class executed a program servant by means of a socket
 * returning the results of the execution of a Genetic Algorithm
 * implemented in the class Exe
 */
package project.primary_server;

import java.util.*;
import java.net.*;
import java.io.*;
import project.util.*;

/**
 *
 * @author Jose manuel Garcia Nieto
 * @version 0
 */

// This class is used instance a task for the servant that takes charge
// to listen the port
public class SystemServer extends Thread {

    /** Creates new ServidorParametro */
    public SystemServer() {
        // We begin the task
        start();
    }

    public void run(){
        try{
            WebStream ws = new WebStream(1140);
            System.out.println("Primary Server Listen Port 80");
            while(true){
                ws.accept();
                new conectParam(ws);
            }

        }catch(Exception e){
            e.printStackTrace();
        }
    }

}

// This class is used a task that it assists a call to rush
// received through from the specified port
class conectParam extends Thread {
    WebStream wstream;
    Class clss;

    conectParam(WebStream ws){
        System.out.println("Recivied a call ");
        wstream = ws;
        setPriority(NORM_PRIORITY-1);
        start();
    }

}

// It throws the thread of attention to the port

```

```

public void run() {
    System.out.println("Thrown the thread of attention ");
    String tip_file = "typeip_file.txt";
    String client_name = null;
    char [] passw = null;
    try{
        wstream.init();
        // get the client name in order to register a new client directory
        int [] num = (int[])wstream.get();
        client_name = wstream.get_string();
        if (num[0] == 1){
            passw = (char[])wstream.get();
            FileWriter fw = new FileWriter("users_file.txt", true);
            fw.write(client_name);
            fw.write(" ");
            fw.write(passw);
            fw.write("\n");
            fw.close();
            wstream.put_string(4,"true");
        }else if (num[0] == 2){
            passw = (char[])wstream.get();
            DataInputStream dis = new DataInputStream(new
FileInputStream("users file.txt"));
            String line = null;
            String ok="false";
            String psw = null;
            while ((line=dis.readLine()) != null){
                StringTokenizer stz = new StringTokenizer(line, " \t");
                if (client_name.equalsIgnoreCase(stz.nextToken()) &&
                    psw.copyValueOf(passw).equalsIgnoreCase(stz.nextToken()))
                    ok = "true";
            }
            dis.close();
            System.out.println("-----" + ok);
            wstream.put_string(ok.length(),ok);
        }else{
            System.out.println("Client Name : "+client_name);
            Register reg = new Register();
            reg.setRoot("register dir worker");
            reg.registration(client_name);

            //Put the file result to send at client
            wstream.put_file(tip_file);
        }
        if (wstream.probe()) wstream.flush();
        wstream.finalize();

        System.out.println("Conection finalized");
    }catch(IOException e){
        e.printStackTrace();
    }catch(Exception e){
        e.printStackTrace();
    }
}
}

```