МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»

кафедра програмних засобів

Звіт

з лабораторної роботи № 4

з дисципліни «Агентно-орієнтоване програмування»

на тему:

«Взаємодія з зовнішніми процесами в Jadex»

Виконав

Ст. гр. КНТ-137 В.В. Козлов

Прийняв

Ст. викладач О.І. Качан

1. Мета роботи

Навчитися працювати із зовнішніми процесами з використанням платформи JADEX.

Вивчити принципи використання сокетів для виконання мережевої взаємодії.

1. Завдання на лабораторну роботу
   1. Варіант 9

Агент, який виконує аналіз пропозицій щодо працевлаштування та інформує про ті, що відповідають критеріям користувача та його здібностям. Набір веб-сайтів, з яких збираються пропозиції, повинен визначатися та в якості додаткової можливості коригуватися користувачем.

Агент повинен уникати дублювання подібних пропозицій. Окрім перевірки на основі прямої відповідності критеріям повинно забезпечуватись навчання за суб’єктивними оцінками користувача.

Реалізація завдання має ґрунтуватися на наступних особливостях: агент повинен мати певну мету, на яку буде направлена вся його діяльність, необхідно забезпечувати використання подій.

Агенти повинні обмінюватися запитами на пошук та даними про веб-сайти, з яких завантажуються пропозиції. Розділити графічний інтерфейс та реалізацію пошуку в різні агенти та організувати їх взаємодію.

1. Результати виконання роботи
   1. Код програми

Програма складається з різних пакетів (для кожної лабораторної роботи окремий пакет з класами) та класу Main, який відповідає за запуск агентів.

* + 1. Main.java

import jadex.base.PlatformConfiguration;  
import jadex.base.Starter;  
  
public class Main {  
 public static void main(String[] args) {  
 PlatformConfiguration config = PlatformConfiguration.getDefaultNoGui();  
  
 //Lab4  
 config.addComponent("Lab4.JobAgentBDI.class");  
 config.addComponent("Lab4.GUIAgentBDI.class");  
 config.addComponent("Lab4.GUIAgentBDI.class");  
  
 Starter.createPlatform(config).get();  
 }  
}

* + 1. Algorithms.java

package Lab4;  
  
public class Algorithms {  
 public static String rightPadding(String text, int length) {  
 return String.format("%" + length + "." + length + "s", text);  
 }  
  
 public static String leftPadding(String text, int length) {  
 return String.format("%-" + length + "." + length + "s", text);  
 }  
  
 public static int portForSocket = 9099;  
}

* + 1. ClassifierJob.java

package Lab4;  
  
import weka.classifiers.Classifier;  
import weka.classifiers.lazy.IBk;  
import weka.core.Attribute;  
import weka.core.DenseInstance;  
import weka.core.Instance;  
import weka.core.Instances;  
  
import java.util.ArrayList;  
  
public class ClassifierJob {  
 ArrayList<String> classVal;  
 ArrayList<Attribute> attributes;  
  
 Attribute salaryAttribute;  
 Attribute classAttribute;  
  
 Classifier classifier;  
  
 public ClassifierJob() {  
 classVal = new ArrayList<String>();  
 classVal.add("Yes");  
 classVal.add("No");  
  
 attributes = new ArrayList<Attribute>();  
  
 salaryAttribute = new Attribute("salary");  
 attributes.add(salaryAttribute);  
  
 classAttribute = new Attribute("class", classVal);  
 attributes.add(classAttribute);  
  
 classifier = null;  
 }  
  
 public void trainClassifier(ClassifierTrainData trainData) {  
 Instances trainingDataset = new Instances("Train dataset", attributes, trainData.questionParameter.size());  
 trainingDataset.setClassIndex(trainingDataset.numAttributes() - 1);  
  
 for(int i = 0; i < trainData.questionParameter.size(); ++i) {  
 Instance newInstance = new DenseInstance(2);  
 newInstance.setValue(salaryAttribute, trainData.questionParameter.get(i));  
 newInstance.setValue(classAttribute, trainData.questionAnswer.get(i) ? "Yes" : "No");  
  
 trainingDataset.add(newInstance);  
 }  
  
 try {  
 Classifier newClassifier = new IBk();  
 newClassifier.buildClassifier(trainingDataset);  
 classifier = newClassifier;  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
  
 public String classify(Job job) {  
 try {  
 if (classifier == null) return "";  
  
 Instances classifyDataset = new Instances("Clasify dataset", attributes, 0);  
 classifyDataset.setClassIndex(classifyDataset.numAttributes() - 1);  
  
 Instance newInstance = new DenseInstance(2);  
 newInstance.setValue(salaryAttribute, job.salary);  
 newInstance.setDataset(classifyDataset);  
  
 int res = (int) classifier.classifyInstance(newInstance);  
 return classVal.get(res);  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
  
 return "";  
 }  
  
 public boolean isReadyToClassify() {  
 return classifier != null;  
 }  
}

* + 1. ClassifierTrainData.java

package Lab4;  
  
import java.util.List;  
  
public class ClassifierTrainData {  
  
 protected List<Integer> questionParameter;  
 protected List<Boolean> questionAnswer;  
  
 public ClassifierTrainData(List<Integer> questionParameter, List<Boolean> questionAnswer) {  
 this.questionParameter = questionParameter;  
 this.questionAnswer = questionAnswer;  
 }  
}

* + 1. FindJobRequest.java

package Lab4;  
  
import java.io.Serializable;  
import java.util.ArrayList;  
import java.util.List;  
  
public class FindJobRequest implements Serializable {  
 public String jobName;  
 public List<String> sitesForSearch = new ArrayList<String>();  
 public String clientInfo = "";  
 public int numberOfRequestOnServer = 1;  
  
 FindJobRequest() { ; }  
  
 @Override  
 public String toString() {  
  
 return numberOfRequestOnServer + ") Client: " + clientInfo + " Request: \"" + jobName + "\" Resources: " + sitesForSearch.toString();  
 }  
}

* + 1. GUIAgentBDI.java

package Lab4;  
  
import jadex.bdiv3.annotation.\*;  
import jadex.bdiv3.features.IBDIAgentFeature;  
import jadex.micro.annotation.Agent;  
import jadex.micro.annotation.AgentBody;  
import jadex.micro.annotation.AgentCreated;  
import jadex.micro.annotation.AgentFeature;  
  
import java.util.ArrayList;  
import java.util.List;  
  
@Agent  
@Plans(@Plan(body=@Body(GUIAgentCapability.openJobSite.class)))  
public class GUIAgentBDI {  
 protected JobFinderForm window;  
  
 @Belief  
 protected List<Job> jobList = new ArrayList<Job>();  
  
 @AgentFeature  
 protected IBDIAgentFeature bdiFeature;  
  
 @Capability(beliefmapping = @Mapping(value = "jobList"))  
 protected GUIAgentCapability capability = new GUIAgentCapability();  
  
 @AgentCreated  
 public void created() {  
 window = new JobFinderForm(this);  
 }  
  
 @AgentBody  
 public void body() {  
 window.show();  
 }  
  
 @Plan(trigger = @Trigger(factchangeds = "jobList"))  
 public void showchanges() {  
 window.setJobJList(jobList);  
  
 System.out.println("Updated job list size: " + jobList.size());  
 System.out.println("Updated job list: ");  
 int i = 1;  
 for (Job job: jobList ) {  
 System.out.print(i++ + ") ");  
 System.out.println(job.toString());  
 }  
 System.out.println();  
 }  
}

* + 1. GUIAgentCapability

package Lab4;  
  
import jadex.bdiv3.annotation.\*;  
  
import java.awt.\*;  
import java.io.\*;  
import java.net.Socket;  
import java.net.URL;  
import java.util.ArrayList;  
import java.util.List;  
  
@Capability  
public class GUIAgentCapability {  
 @Belief  
 public native List<Job> getJobList();  
  
 @Belief  
 public native void setJobList(List<Job> jobList);  
  
 @Plan  
 class openJobSite {  
 private String url;  
  
 openJobSite(Job job) {  
 this.url = job.reference;  
 }  
  
 @PlanBody  
 public void main() {  
 System.out.println("Go to reference: " + url);  
 System.out.println();  
  
 try {  
 Desktop.getDesktop().browse(new URL(url).toURI());  
 } catch (Exception exc) {  
 exc.printStackTrace();  
 }  
 }  
 }  
  
 @Goal  
 public class FindJob {  
 @GoalParameter  
 protected FindJobRequest request;  
  
 public FindJob(FindJobRequest request) {  
 this.request = request;  
 }  
 }  
  
 @Goal  
 public class FilterJob {  
 @GoalParameter  
 protected ClassifierTrainData trainData;  
  
 public FilterJob(ClassifierTrainData trainData) {  
 this.trainData = trainData;  
 }  
 }  
  
 @Plan(trigger = @Trigger(goals = FilterJob.class))  
 protected void filterJob(ClassifierTrainData trainData) {  
 System.out.println("Start filtering");  
  
 ArrayList<Job> filteredJobs = new ArrayList<Job> ();  
  
 System.out.println("Start train classifier");  
  
 ClassifierJob classifier = new ClassifierJob();  
 classifier.trainClassifier(trainData);  
  
 System.out.println("End train classifier");  
 System.out.println("Start classifying");  
  
 if(classifier.isReadyToClassify()) {  
 for(Job job : getJobList()) {  
 if(classifier.classify(job).equals("Yes")) {  
 filteredJobs.add(job);  
 }  
 }  
 }  
  
 System.out.println("End classifying");  
 System.out.println("Start deleting same jobs");  
  
 for(int i = 0; i < filteredJobs.size(); ++i) {  
 for(int j = i + 1; j < filteredJobs.size(); ++j) {  
 if(filteredJobs.get(i).name.equals(filteredJobs.get(j).name)) {  
 filteredJobs.remove(j);  
 --j;  
 }  
 }  
 }  
  
 System.out.println("End deleting same jobs");  
 System.out.println();  
  
 setJobList(filteredJobs);  
 }  
  
 @Plan(trigger = @Trigger(goals = FindJob.class))  
 protected void findJob(FindJobRequest request) {  
 try {  
 Socket clientSocket = new Socket("localhost", Algorithms.portForSocket);  
  
 System.out.println("Send request to server");  
  
 ObjectOutputStream outStream = new ObjectOutputStream(clientSocket.getOutputStream());  
 outStream.writeObject(request);  
 outStream.flush();  
  
 System.out.println("Get answer from server");  
  
 ObjectInputStream inStream = new ObjectInputStream(clientSocket.getInputStream());  
 try {  
 ArrayList<Job> jobList = (ArrayList<Job>)inStream.readObject();  
 setJobList(jobList);  
 } catch (ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
  
 clientSocket.close();  
 inStream.close();  
 outStream.close();  
 }  
 catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
}

* + 1. Job.java

package Lab4;  
  
import java.io.Serializable;  
  
public class Job implements Serializable {  
 public String name;  
 public String company;  
 public int salary = -1;  
 public String siteName;  
 public String reference;  
  
 Job() {  
 ;  
 }  
  
 @Override  
 public String toString() {  
 String res = "";  
 res += Algorithms.rightPadding(siteName, 10) + " - ";  
 res += Algorithms.rightPadding(company, 16) + " - ";  
 if(salary == -1)  
 res += Algorithms.rightPadding(" ", 10) + " - ";  
 else  
 res += Algorithms.rightPadding(String.valueOf(salary) + " грн.", 10) + " - ";  
 res += name.length() > 75 ? name.substring(0, 75) : name;  
  
 return res;  
 }  
}

* + 1. JobAgentBDI.java

package Lab4;  
  
import jadex.bdiv3.annotation.\*;  
import jadex.bdiv3.features.IBDIAgentFeature;  
import jadex.bdiv3.runtime.ChangeEvent;  
import jadex.micro.annotation.Agent;  
import jadex.micro.annotation.AgentBody;  
import jadex.micro.annotation.AgentCreated;  
import jadex.micro.annotation.AgentFeature;  
import jadex.rules.eca.ChangeInfo;  
import org.jsoup.HttpStatusException;  
import org.jsoup.Jsoup;  
import org.jsoup.nodes.Document;  
import org.jsoup.nodes.Element;  
import org.jsoup.select.Elements;  
  
import java.io.\*;  
import java.net.ServerSocket;  
import java.net.Socket;  
import java.util.ArrayList;  
import java.util.List;  
  
@Agent  
public class JobAgentBDI {  
 protected ServerRequestHistoryForm window;  
 private static ServerSocket server;  
  
 @Belief  
 protected List<FindJobRequest> requestHistoryList = new ArrayList<FindJobRequest>();  
  
 @AgentFeature  
 protected IBDIAgentFeature bdiFeature;  
  
 @AgentCreated  
 public void created() {  
 window = new ServerRequestHistoryForm();  
  
 Runnable run = new Runnable() {  
 public void run() {  
 if(server!=null) {  
 try {  
 server.close();  
 }  
 catch(Exception e) {  
 ;  
 }  
 }  
 try {  
 server = new ServerSocket(Algorithms.portForSocket);  
  
 while(true) {  
 Socket clientSocket = server.accept();  
  
 bdiFeature.dispatchTopLevelGoal(new FindJob(clientSocket)).get();  
 }  
 }  
 catch(IOException e)  
 {  
 throw new RuntimeException(e.getMessage());  
 }  
 }  
 };  
  
 Thread newThread = new Thread(run);  
 newThread.start();  
 }  
  
 @AgentBody  
 public void body() {  
 window.show();  
 }  
  
 @Plan(trigger=@Trigger(factaddeds="requestHistoryList"))  
 public void showchanges(ChangeEvent event)  
 {  
 FindJobRequest addedRequest = ((ChangeInfo<FindJobRequest>)event.getValue()).getValue();  
 window.addRequestToHistoryList(addedRequest);  
 }  
  
 @Goal  
 public class FindJob {  
 @GoalParameter  
 protected Socket clientSocket;  
  
 public FindJob(Socket clientSocket) {  
 this.clientSocket = clientSocket;  
 }  
 }  
  
 @Plan(trigger = @Trigger(goals = FindJob.class))  
 protected void findJob(Socket clientSocket) {  
 try {  
 try {  
 ObjectInputStream inStream = new ObjectInputStream(clientSocket.getInputStream());  
 FindJobRequest request = (FindJobRequest) inStream.readObject();  
  
 System.out.println("Request: \"" + request.jobName + "\"");  
 System.out.print("Resources: ");  
 System.out.println(request.sitesForSearch.toString());  
 System.out.println();  
  
 request.clientInfo = clientSocket.toString();  
 request.numberOfRequestOnServer = requestHistoryList.size() + 1;  
 requestHistoryList.add(request);  
  
 List<Job> jobList = new ArrayList<Job>();  
 if(request.sitesForSearch.contains("Rabota.ua"))  
 jobList.addAll(getJobsRabotaUA(request.jobName));  
 if(request.sitesForSearch.contains("Work.ua"))  
 jobList.addAll(getJobsWorkUA(request.jobName));  
  
 ObjectOutputStream outStream = new ObjectOutputStream(clientSocket.getOutputStream());  
 outStream.writeObject(jobList);  
 outStream.flush();  
  
 clientSocket.close();  
 inStream.close();  
 outStream.close();  
 } catch (ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public List<Job> getJobsRabotaUA(String jobName) {  
 List<Job> jobList = new ArrayList<Job>();  
  
 try {  
 Document doc = Jsoup.connect("https://rabota.ua/zapros/" + jobName + "/Запорожье").get();  
  
 for(Element card : doc.getElementsByClass("card")) {  
 Job newJob = new Job();  
  
 newJob.name = card.getElementsByClass("ga\_listing").first().attr("title");  
 newJob.company = card.getElementsByClass("company-profile-name").first().attr("title");  
 String salary = card.getElementsByClass("salary").first().text();  
 salary = salary.split("—")[0].replaceAll("[^0-9]", "");  
 if(salary.equals("")) salary = "-1";  
 newJob.salary = Integer.parseInt(salary);  
 newJob.siteName = "Rabota.ua";  
 newJob.reference = "https://rabota.ua" + card.getElementsByClass("ga\_listing").first().attr("href");  
  
 jobList.add(newJob);  
 }  
 }  
 catch (Exception e) {  
 if(e instanceof HttpStatusException) {  
 System.out.print(e.toString());  
 }  
 else if(e instanceof IOException) {  
 e.printStackTrace();  
 }  
 }  
  
 return jobList;  
 }  
  
 public List<Job> getJobsWorkUA(String jobName) {  
 List<Job> jobList = new ArrayList<Job>();  
  
 try {  
 Document doc = Jsoup.connect("https://www.work.ua/ru/jobs-zaporizhzhya-" + jobName).get();  
  
 for(Element card : doc.getElementsByClass("card card-hover card-visited wordwrap job-link")) {  
 Job newJob = new Job();  
  
 newJob.name = card.getElementsByTag("a").first().text();  
 newJob.company = card.getElementsByClass("add-top-xs").first().getElementsByTag("b").first().text();  
 String salary = "-1";  
 Elements divNoClass = card.select("div:not([class])");  
 if(!divNoClass.isEmpty()) {  
 Elements bTag = divNoClass.first().getElementsByTag("b");  
 if(!bTag.isEmpty()) {  
 salary = bTag.first().text();  
 salary = salary.split("–")[0].replaceAll("[^0-9]", "");  
 }  
 }  
 newJob.salary = Integer.parseInt(salary);  
 newJob.siteName = "Work.ua";  
 newJob.reference = "https://work.ua" + card.getElementsByTag("a").first().attr("href");  
  
 jobList.add(newJob);  
 }  
 }  
 catch (Exception e) {  
 if(e instanceof HttpStatusException ) {  
 System.out.print(e.toString());  
 }  
 else if(e instanceof IOException) {  
 e.printStackTrace();  
 }  
 }  
  
 return jobList;  
 }  
}

* + 1. FilterParametersForm.java

package Lab4;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.util.ArrayList;  
import java.util.List;  
  
public class FilterParametersForm {  
 private JPanel mainPanel;  
 private JLabel windowDescription;  
 private JCheckBox question1;  
 private JCheckBox question2;  
 private JCheckBox question3;  
 private JCheckBox question4;  
 private JCheckBox question5;  
 private JButton filterButton;  
  
 protected GUIAgentBDI parentBDI;  
  
 public FilterParametersForm(GUIAgentBDI parentBDI) {  
 this.parentBDI = parentBDI;  
  
 filterButton.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
  
 List<Integer> questionParameter = new ArrayList<Integer>();  
 List<Boolean> questionAnswer = new ArrayList<Boolean>();  
  
 questionParameter.add(-1);  
 questionAnswer.add(question1.isSelected());  
 questionParameter.add(5000);  
 questionAnswer.add(question2.isSelected());  
 questionParameter.add(10000);  
 questionAnswer.add(question3.isSelected());  
 questionParameter.add(20000);  
 questionAnswer.add(question4.isSelected());  
 questionParameter.add(30000);  
 questionAnswer.add(question5.isSelected());  
  
 ClassifierTrainData dataToTrainClassifier = new ClassifierTrainData(questionParameter, questionAnswer);  
 parentBDI.bdiFeature.dispatchTopLevelGoal(parentBDI.capability.new FilterJob(dataToTrainClassifier)).get();  
  
 SwingUtilities.getWindowAncestor(mainPanel).dispose();  
 }  
 });  
 }  
  
 public void show() {  
 JDialog frame = new JDialog(new JFrame(), "Filter", true);  
 Toolkit toolkit = Toolkit.getDefaultToolkit();  
 Dimension dimension = toolkit.getScreenSize();  
 frame.setContentPane(mainPanel);  
 frame.pack();  
 frame.setBounds((dimension.width - frame.getWidth()) / 2, (dimension.height - frame.getHeight()) / 2, frame.getWidth(), frame.getHeight());  
 frame.setVisible(true);  
 }  
}

* + 1. FilterParametersForm.form

<?xml version="1.0" encoding="UTF-8"?>

<form xmlns="http://www.intellij.com/uidesigner/form/" version="1" bind-to-class="Lab4.FilterParametersForm">

<grid id="27dc6" binding="mainPanel" layout-manager="GridLayoutManager" row-count="7" column-count="1" same-size-horizontally="false" same-size-vertically="false" hgap="-1" vgap="-1">

<margin top="10" left="10" bottom="10" right="10"/>

<constraints>

<xy x="20" y="20" width="500" height="400"/>

</constraints>

<properties>

<font name="Monospaced"/>

</properties>

<border type="none"/>

<children>

<component id="37c2" class="javax.swing.JLabel" binding="windowDescription">

<constraints>

<grid row="0" column="0" row-span="1" col-span="1" vsize-policy="0" hsize-policy="0" anchor="0" fill="1" indent="0" use-parent-layout="false"/>

</constraints>

<properties>

<font name="Monospaced"/>

<horizontalAlignment value="0"/>

<horizontalTextPosition value="0"/>

<text value="Select statements that match your search goal"/>

</properties>

</component>

<component id="d4fa8" class="javax.swing.JCheckBox" binding="question1">

<constraints>

<grid row="1" column="0" row-span="1" col-span="1" vsize-policy="0" hsize-policy="3" anchor="8" fill="0" indent="0" use-parent-layout="false"/>

</constraints>

<properties>

<font name="Monospaced"/>

<text value="Salary not specified"/>

</properties>

</component>

<component id="4e719" class="javax.swing.JCheckBox" binding="question2">

<constraints>

<grid row="2" column="0" row-span="1" col-span="1" vsize-policy="0" hsize-policy="3" anchor="8" fill="0" indent="0" use-parent-layout="false"/>

</constraints>

<properties>

<font name="Monospaced"/>

<text value="Salary 5000 UAH"/>

</properties>

</component>

<component id="403ee" class="javax.swing.JCheckBox" binding="question3">

<constraints>

<grid row="3" column="0" row-span="1" col-span="1" vsize-policy="0" hsize-policy="3" anchor="8" fill="0" indent="0" use-parent-layout="false"/>

</constraints>

<properties>

<font name="Monospaced"/>

<text value="Salary 10000 UAH"/>

</properties>

</component>

<component id="74b5" class="javax.swing.JCheckBox" binding="question4">

<constraints>

<grid row="4" column="0" row-span="1" col-span="1" vsize-policy="0" hsize-policy="3" anchor="8" fill="0" indent="0" use-parent-layout="false"/>

</constraints>

<properties>

<font name="Monospaced"/>

<text value="Salary 20000 UAH"/>

</properties>

</component>

<component id="71c7a" class="javax.swing.JCheckBox" binding="question5">

<constraints>

<grid row="5" column="0" row-span="1" col-span="1" vsize-policy="0" hsize-policy="3" anchor="8" fill="0" indent="0" use-parent-layout="false"/>

</constraints>

<properties>

<font name="Monospaced"/>

<text value="Salary 35000 UAH"/>

</properties>

</component>

<component id="cc995" class="javax.swing.JButton" binding="filterButton">

<constraints>

<grid row="6" column="0" row-span="1" col-span="1" vsize-policy="0" hsize-policy="3" anchor="0" fill="1" indent="0" use-parent-layout="false"/>

</constraints>

<properties>

<font name="Monospaced"/>

<text value="Filter"/>

</properties>

</component>

</children>

</grid>

</form>

* + 1. JobFinderForm.java

package Lab4;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.util.List;  
  
public class JobFinderForm {  
 private JPanel mainPanel;  
 private JTextField searchRequest;  
 private JCheckBox isRabotaUA;  
 private JCheckBox isWorkUA;  
 private JButton searchButton;  
 private DefaultListModel jobListModel;  
 private JList JobJList;  
 private JButton openJobSiteButton;  
 private JPanel searchParametersPanel;  
 private JScrollPane JListScrollPanel;  
 private JButton filterJobListButton;  
  
 protected GUIAgentBDI parentBDI;  
  
 public JobFinderForm(GUIAgentBDI parentBDI) {  
 this.parentBDI = parentBDI;  
  
 jobListModel = new DefaultListModel();  
 JobJList.setModel(jobListModel);  
  
 searchButton.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 FindJobRequest request = new FindJobRequest();  
  
 request.jobName = searchRequest.getText();  
 if(isRabotaUA.isSelected())  
 request.sitesForSearch.add("Rabota.ua");  
 if(isWorkUA.isSelected())  
 request.sitesForSearch.add("Work.ua");  
  
 parentBDI.bdiFeature.dispatchTopLevelGoal(parentBDI.capability.new FindJob(request)).get();  
  
 searchRequest.setText("");  
 JListScrollPanel.getHorizontalScrollBar().setValue(0);  
 JListScrollPanel.getVerticalScrollBar().setValue(0);  
 }  
 });  
  
 openJobSiteButton.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 for(Job job : (List<Job>)JobJList.getSelectedValuesList()) {  
 parentBDI.bdiFeature.adoptPlan(parentBDI.capability.new openJobSite(job));  
 }  
 }  
 });  
 filterJobListButton.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 FilterParametersForm window = new FilterParametersForm(parentBDI);  
 window.show();  
 }  
 });  
 }  
  
 public void show() {  
 JFrame mainFrame = new JFrame("GUIAgent");  
 Toolkit toolkit = Toolkit.getDefaultToolkit();  
 Dimension dimension = toolkit.getScreenSize();  
 mainFrame.setBounds((dimension.width - 850) / 2, (dimension.height - 500) / 2, 850, 500);  
 mainFrame.setContentPane(mainPanel);  
 //mainFrame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  
 mainFrame.setVisible(true);  
 }  
  
 public void setJobJList(List<Job> jobList) {  
 jobListModel.clear();  
  
 for(Job job : jobList) {  
 jobListModel.addElement(job);  
 }  
  
 // May need to use if JList don`t updates  
 //JobJList.revalidate();  
 //JobJList.repaint();  
  
 // Instead of using DefaultListModel jobListModel  
 // setListData constructs a read-only ListModel from an array of items, and calls setModel with this model  
 //JobJList.setListData(jobList.toArray());  
 }  
}

* + 1. JobFinderForm.form

<?xml version="1.0" encoding="UTF-8"?>

<form xmlns="http://www.intellij.com/uidesigner/form/" version="1" bind-to-class="Lab4.JobFinderForm">

<grid id="27dc6" binding="mainPanel" layout-manager="GridLayoutManager" row-count="5" column-count="1" same-size-horizontally="false" same-size-vertically="false" hgap="-1" vgap="-1">

<margin top="10" left="10" bottom="10" right="10"/>

<constraints>

<xy x="20" y="20" width="500" height="400"/>

</constraints>

<properties>

<font name="Monospaced"/>

</properties>

<border type="none"/>

<children>

<grid id="e4925" binding="searchParametersPanel" layout-manager="GridLayoutManager" row-count="1" column-count="3" same-size-horizontally="false" same-size-vertically="false" hgap="-1" vgap="-1">

<margin top="0" left="0" bottom="0" right="0"/>

<constraints>

<grid row="0" column="0" row-span="1" col-span="1" vsize-policy="3" hsize-policy="3" anchor="0" fill="3" indent="0" use-parent-layout="false"/>

</constraints>

<properties>

<font name="Monospaced"/>

</properties>

<border type="none"/>

<children>

<component id="cc07f" class="javax.swing.JTextField" binding="searchRequest">

<constraints>

<grid row="0" column="0" row-span="1" col-span="1" vsize-policy="0" hsize-policy="6" anchor="8" fill="1" indent="0" use-parent-layout="false">

<preferred-size width="150" height="-1"/>

</grid>

</constraints>

<properties>

<font name="Monospaced"/>

</properties>

</component>

<component id="d01a3" class="javax.swing.JCheckBox" binding="isRabotaUA">

<constraints>

<grid row="0" column="1" row-span="1" col-span="1" vsize-policy="0" hsize-policy="3" anchor="8" fill="0" indent="0" use-parent-layout="false"/>

</constraints>

<properties>

<font name="Monospaced"/>

<selected value="true"/>

<text value="Rabota.ua"/>

</properties>

</component>

<component id="fa7f1" class="javax.swing.JCheckBox" binding="isWorkUA">

<constraints>

<grid row="0" column="2" row-span="1" col-span="1" vsize-policy="0" hsize-policy="3" anchor="8" fill="0" indent="0" use-parent-layout="false"/>

</constraints>

<properties>

<font name="Monospaced"/>

<selected value="true"/>

<text value="Work.ua"/>

</properties>

</component>

</children>

</grid>

<component id="f492a" class="javax.swing.JButton" binding="searchButton">

<constraints>

<grid row="1" column="0" row-span="1" col-span="1" vsize-policy="0" hsize-policy="3" anchor="0" fill="1" indent="0" use-parent-layout="false"/>

</constraints>

<properties>

<font name="Monospaced"/>

<text value="Search"/>

</properties>

</component>

<component id="59654" class="javax.swing.JButton" binding="filterJobListButton">

<constraints>

<grid row="2" column="0" row-span="1" col-span="1" vsize-policy="0" hsize-policy="3" anchor="0" fill="1" indent="0" use-parent-layout="false"/>

</constraints>

<properties>

<font name="Monospaced"/>

<text value="Filter"/>

</properties>

</component>

<scrollpane id="59f28" binding="JListScrollPanel">

<constraints>

<grid row="3" column="0" row-span="1" col-span="1" vsize-policy="7" hsize-policy="7" anchor="0" fill="3" indent="0" use-parent-layout="false"/>

</constraints>

<properties>

<font name="Monospaced"/>

</properties>

<border type="none"/>

<children>

<component id="b3811" class="javax.swing.JList" binding="JobJList">

<constraints/>

<properties>

<font name="Monospaced"/>

</properties>

</component>

</children>

</scrollpane>

<component id="69b27" class="javax.swing.JButton" binding="openJobSiteButton">

<constraints>

<grid row="4" column="0" row-span="1" col-span="1" vsize-policy="0" hsize-policy="3" anchor="0" fill="1" indent="0" use-parent-layout="false"/>

</constraints>

<properties>

<font name="Monospaced"/>

<text value="Go to site"/>

</properties>

</component>

</children>

</grid>

</form>

* + 1. ServerRequestHistoryForm.java

package Lab4;  
  
import javax.swing.\*;  
import java.awt.\*;  
  
public class ServerRequestHistoryForm {  
 private JPanel mainPanel;  
 private DefaultListModel requestHistoryListModel;  
 private JList requestHistoryList;  
 private JScrollPane JScrollPanel;  
  
 public ServerRequestHistoryForm() {  
 requestHistoryListModel = new DefaultListModel();  
 requestHistoryList.setModel(requestHistoryListModel);  
 }  
  
 public void show() {  
 JFrame mainFrame = new JFrame("JobAgent");  
 Toolkit toolkit = Toolkit.getDefaultToolkit();  
 Dimension dimension = toolkit.getScreenSize();  
 mainFrame.setBounds((dimension.width - 850), 0, 850, 500);  
 mainFrame.setContentPane(mainPanel);  
 mainFrame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  
 mainFrame.setVisible(true);  
 }  
  
 public void addRequestToHistoryList(FindJobRequest request) {  
 requestHistoryListModel.addElement(request);  
 }  
}

* + 1. ServerRequestHistoryForm.form

<?xml version="1.0" encoding="UTF-8"?>

<form xmlns="http://www.intellij.com/uidesigner/form/" version="1" bind-to-class="Lab4.ServerRequestHistoryForm">

<grid id="27dc6" binding="mainPanel" layout-manager="GridLayoutManager" row-count="1" column-count="1" same-size-horizontally="false" same-size-vertically="false" hgap="-1" vgap="-1">

<margin top="10" left="10" bottom="10" right="10"/>

<constraints>

<xy x="20" y="20" width="500" height="400"/>

</constraints>

<properties>

<font name="Monospaced"/>

</properties>

<border type="none"/>

<children>

<scrollpane id="422dc" binding="JScrollPanel">

<constraints>

<grid row="0" column="0" row-span="1" col-span="1" vsize-policy="7" hsize-policy="7" anchor="0" fill="3" indent="0" use-parent-layout="false"/>

</constraints>

<properties>

<font name="Monospaced"/>

</properties>

<border type="none"/>

<children>

<component id="b7495" class="javax.swing.JList" binding="requestHistoryList">

<constraints/>

<properties>

<font name="Monospaced"/>

</properties>

</component>

</children>

</scrollpane>

</children>

</grid>

</form>

* 1. Результати виконання програми

Для того, щоб зібрати проект в .jar архів для подальшого використання та запуску можна використати артефакти в IntelliJ IDEA (рис. 3.1).

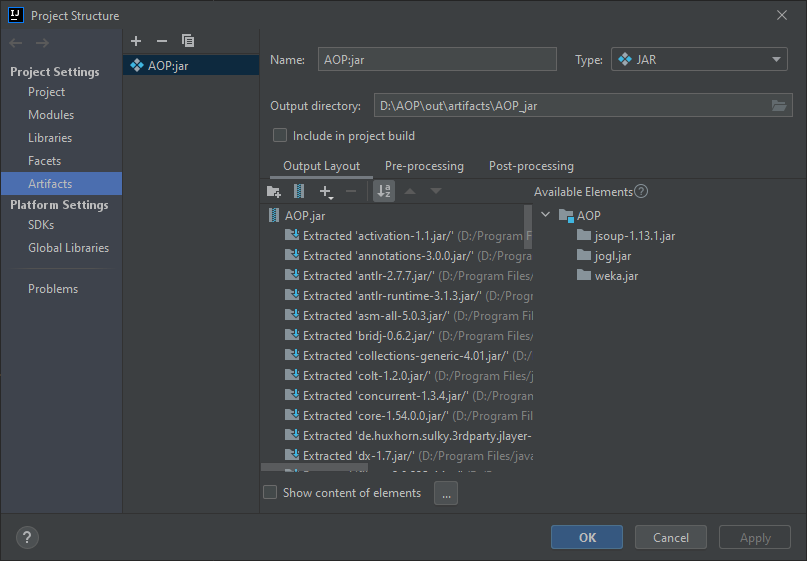


Рисунок 3.1 – Створення артефакта для отримання .jar архіва проекта

Запустити проект можна звичайним шляхом (з інтегрованого середовища розробки IntelliJ IDEA) за допомогою команд Build та Run, або за допомогою створеного .jar архіва, виконавши команду «java –jar file\_path» з командного рядка.

На рисунку 3.2 наведено результати пошуку по запиту «Програміст» на сайтах Rabota.ua та Work.ua.

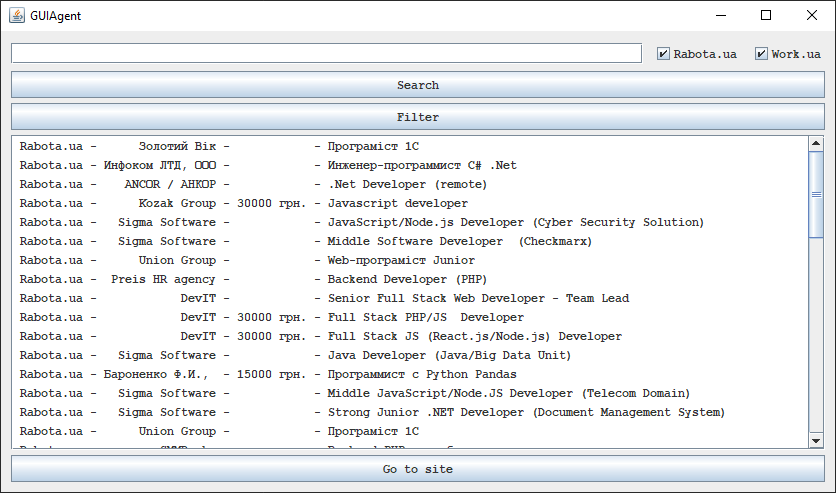


Рисунок 3.2 – Пошук по запиту «Програміст» на сайтах Rabota.ua та Work.ua

На рисунку 3.3 наведено вікно параметрів для навчання класифікатора.

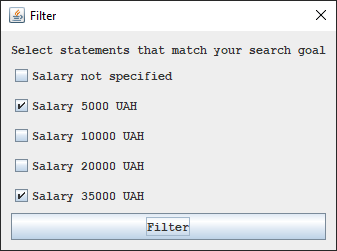


Рисунок 3.3 – Встановлення параметрів для навчання класифікатора

На рисунку 3.4 наведено результати застосування інтелектуальних здібностей агента.

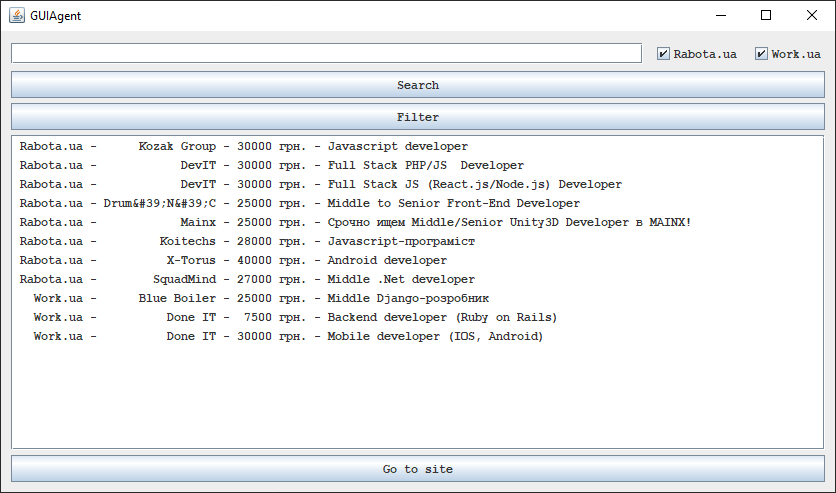


Рисунок 3.4 – Результати застосування інтелектуальних здібностей агента

На рисунку 3.5 наведено результати пошуку іншого агента по запиту «Водій» на сайтах Rabota.ua та Work.ua

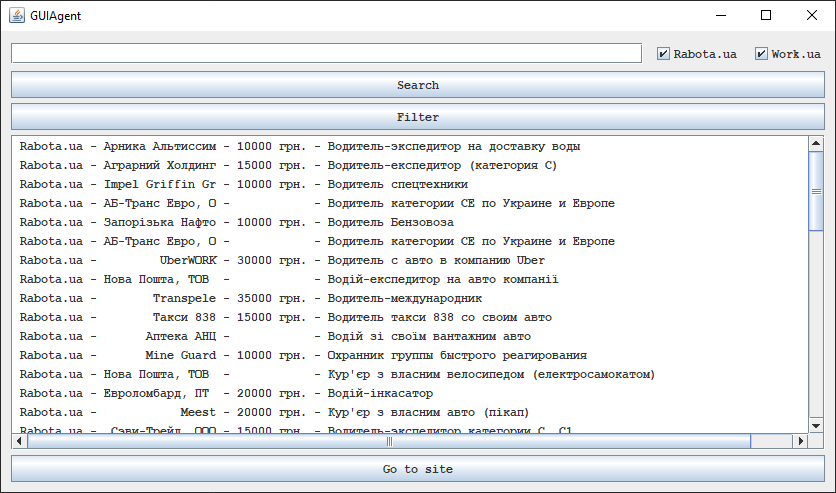


Рисунок 3.5 – Пошук іншого агента по запиту «Програміст» на сайтах Rabota.ua та Work.ua

На рисунку 3.6 наведено вікно серверного агента з усією історією запитів до сервера.

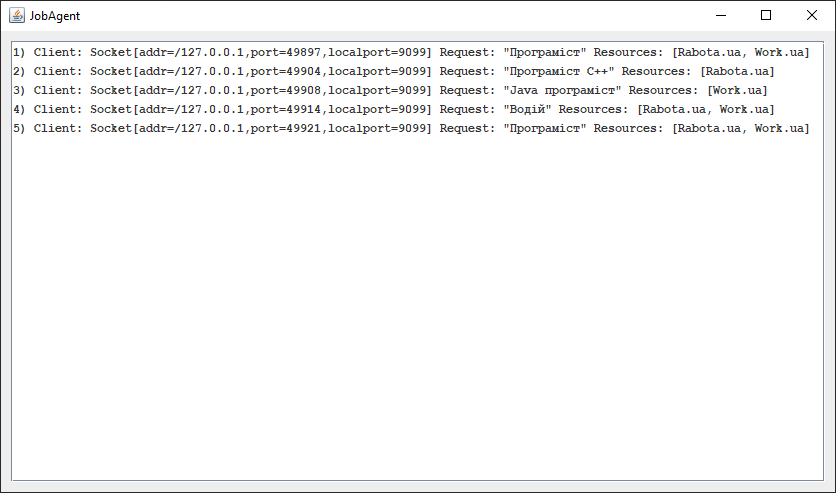


Рисунок 3.6 – Вікно серверного агента з усією історією запитів до сервера

Додатково всі дії агентів відображено в консолі (рис. 3.7).

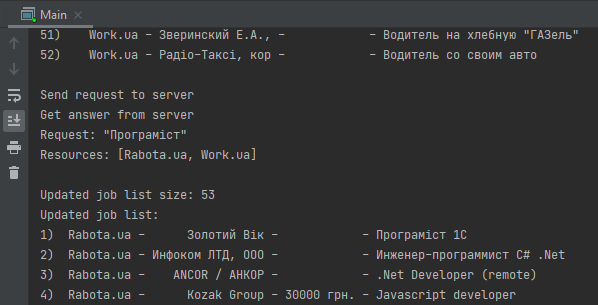


Рисунок 3.7 – Відображення дій агентів в консолі

Висновки

Протягом виконання цієї лабораторної роботи навчився працювати із зовнішніми процесами з використанням платформи JADEX та вивчив принципи використання сокетів для виконання мережевої взаємодії.

Створив агента з графічним інтерфейсом, який виконує аналіз пропозицій щодо працевлаштування та інформує про ті, що відповідають критеріям користувача та його здібностям. Набір веб-сайтів, з яких збираються пропозиції визначається та може коригуватися користувачем. Агент уникає дублювання подібних пропозицій та виконує фільтрацію результатів пошуку за допомогою навчання за суб’єктивними оцінками користувача.

Для фільтрування результатів пошуку та навчання за суб’єктивними оцінками користувача використано бібліотеку Weka та класифікатор IBk.

Реалізація агента ґрунтується на особливості, що агент має певну мету, на яку направлена вся його діяльність.

Розділив графічний інтерфейс та реалізацію пошуку в різні агенти та організувати їх взаємодію. Агенти обмінюються запитами на пошук та даними про веб-сайти, з яких завантажуються пропозиції.