МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»

кафедра програмних засобів

Звіт

з лабораторної роботи № 1

з дисципліни «Агентно-орієнтоване програмування»

на тему:

«Використання планів, переконань та можливостей при створенні агентів»

Виконав

Ст. гр. КНТ-137 В.В. Козлов

Прийняв

Ст. викладач О.І. Качан

1. Мета роботи

Вивчити основні принципи опису планів, переконань та можливостей агентів.

Навчитися реалізовувати агентів за допомогою засобів JADEX на основі опису їх планів, переконань та можливостей.

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.

1. Результати виконання роботи

Для виконання лабораторних робіт з дисципліни «Агентно-орієнтоване програмування» було завантажено та встановлено такі бібліотеки та програмне забезпечення:

- Java SE Development Kit 8 (jdk-8u281-windows-x64);

- Інтегроване середовище розробки IntelliJ IDEA Community Edition 2020.3.2 x64;

- Бібліотека jadex-3.0.117;

- Бібліотека weka-3-8-5;

- Бібліотека jsoup-1.13.1.

Було створено проєкт в IntelliJ IDEA та підключено всі необхідні .jar бібліотеки (модулі) (рис. 3.1).

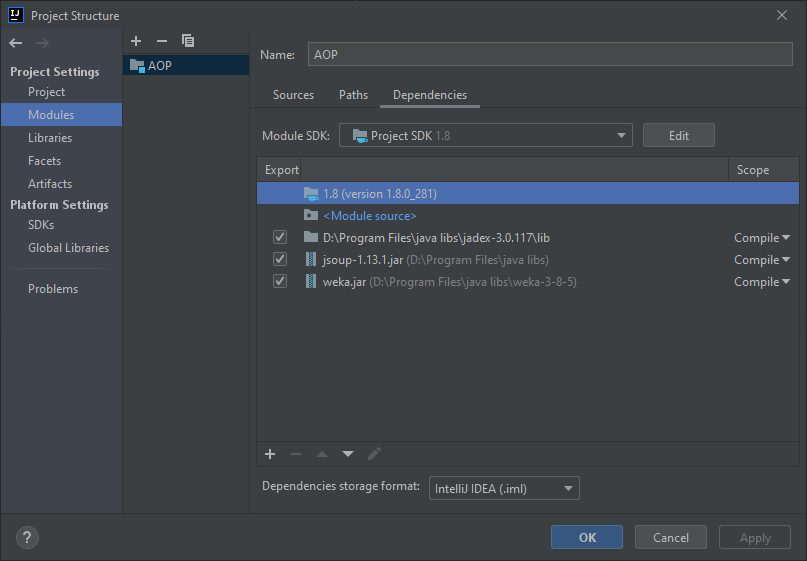


Рисунок 3.1 – Підключення бібліотек до проєкту

* 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();  
  
 // Lab1  
 config.addComponent("Lab1.JobAgentBDI.class");  
  
 Starter.createPlatform(config).get();  
 }  
}

* + 1. Algorithms.java

package Lab1;  
  
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);  
 }  
}

* + 1. FindJobRequest.java

package Lab1;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class FindJobRequest {  
 public String jobName;  
 public List<String> sitesForSearch = new ArrayList<String>();  
  
 FindJobRequest() { ; }  
}

* + 1. Job.java

package Lab1;  
  
public class Job {  
 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 Lab1;  
  
import jadex.bdiv3.annotation.\*;  
import jadex.bdiv3.features.IBDIAgentFeature;  
import jadex.micro.annotation.\*;  
  
import java.util.ArrayList;  
import java.util.List;  
  
@Agent  
@Plans(@Plan(body=@Body(JobAgentCapability.openJobSite.class)))  
public class JobAgentBDI {  
 protected JobFinderForm window;  
  
 @Belief  
 protected List<Job> jobList = new ArrayList<Job>();  
  
 @AgentFeature  
 protected IBDIAgentFeature bdiFeature;  
  
 @Capability(beliefmapping = @Mapping(value = "jobList"))  
 protected JobAgentCapability capability = new JobAgentCapability();  
  
 @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);  
  
 // Print to console  
 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. JobAgentCapability.java

package Lab1;  
  
import jadex.bdiv3.annotation.\*;  
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.awt.\*;  
import java.io.IOException;  
import java.net.URL;  
import java.util.ArrayList;  
import java.util.List;  
  
@Capability  
public class JobAgentCapability {  
 @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() {  
 // Print to console  
 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;  
 }  
 }  
  
 @Plan(trigger = @Trigger(goals = FindJob.class))  
 protected void findJob(FindJobRequest request)  
 {  
 // Print to console  
 System.out.println("Request: \"" + request.jobName + "\"");  
 System.out.print("Resources: ");  
 System.out.println(request.sitesForSearch.toString());  
 System.out.println();  
  
 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));  
  
 this.setJobList(jobList);  
 }  
  
 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. JobFinderForm.java

package Lab1;  
  
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;  
  
 protected JobAgentBDI parentBDI;  
  
 public JobFinderForm(JobAgentBDI 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();  
  
 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));  
 }  
 }  
 });  
 }  
  
 public void show() {  
 JFrame mainFrame = new JFrame("JobsAgent");  
 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);  
 }  
  
 // 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="Lab1.JobFinderForm">

<grid id="27dc6" binding="mainPanel" layout-manager="GridLayoutManager" row-count="4" 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>

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

<constraints>

<grid row="2" 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="3" 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. Результати виконання програми

На рисунках 3.2 – 3.3 наведено скріншоти графічного інтерфейсу агента при роботі.

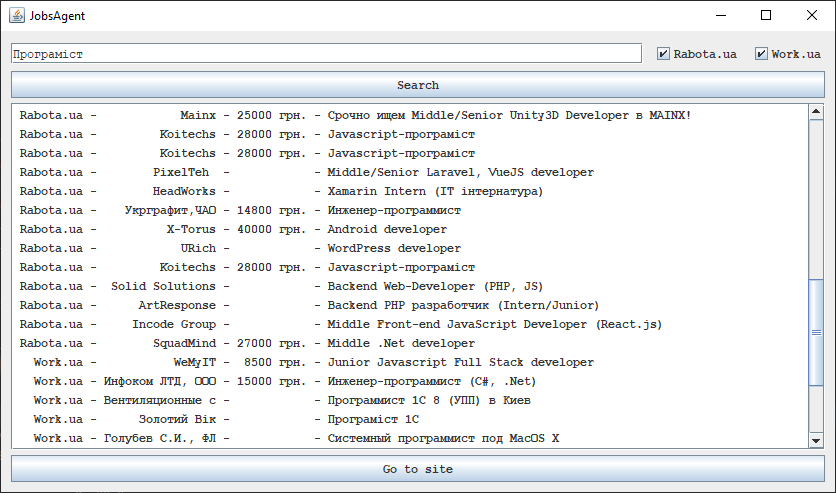


Рисунок 3.2 – Пошук по запиту «Програміст» на сайтах Rabota.ua та Work.ua

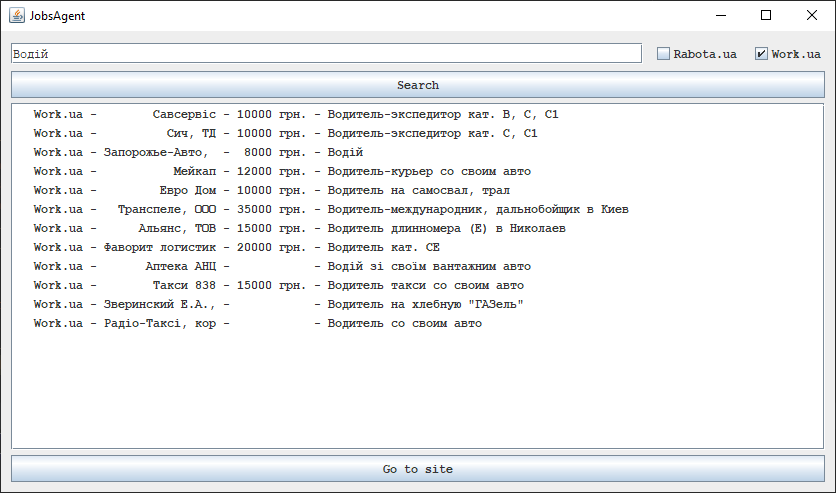


Рисунок 3.3 – Пошук по запиту «Водій» на сайті Work.ua

Додатково всі дії, які виконує агент відображено в консолі (рис. 3.4).

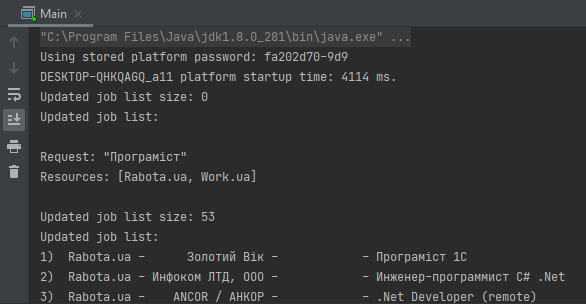


Рисунок 3.4 – Відображення дій агента в консолі

Висновки

Протягом виконання цієї лабораторної роботи вивчив основні принципи опису планів, переконань та можливостей агентів. Навчився реалізовувати агентів за допомогою засобів JADEX на основі опису їх планів, переконань та можливостей.

Завантажив та встановив Java SE Development Kit 8, Інтегроване середовище розробки IntelliJ IDEA, бібліотеки jadex, weka та jsoup.

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.