Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Кафедра інформатики

Звіт з лабораторної роботи №1

По предмету: «Штучний інтелект для інформаційних технологій»

Виконав:

студент групи ІТІНФ-20-1

Самченко С. О.

Харків 2023

**Мета:**1.1.1 Вивчити основні принципи моделі BDI та основні принципи опису планів, переконань та можливостей агентів.

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

**Вибрана тема завдання:**Емулятор гри у пінгпонг

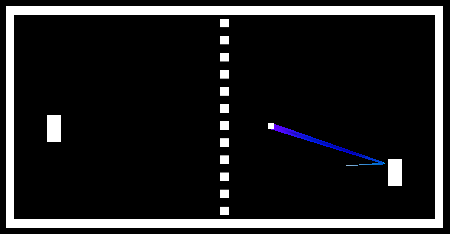


Рис. 1 – результат роботи програми

**Код програми:**

Main.java

package main;  
  
import java.util.HashMap;  
import java.util.Map;  
  
import agents.PingPongAgentBDI;  
import jadex.bridge.IComponentIdentifier;  
import jadex.bridge.IExternalAccess;  
import jadex.bridge.service.RequiredServiceInfo;  
import jadex.bridge.service.search.SServiceProvider;  
import jadex.bridge.service.types.cms.CreationInfo;  
import jadex.bridge.service.types.cms.IComponentManagementService;  
import jadex.commons.future.IFuture;  
import jadex.commons.future.ITuple2Future;  
import jadex.commons.future.ThreadSuspendable;  
import util.AgentDeployer;  
  
public class Main {  
 public static IExternalAccess platform;  
 public static String platformName;  
 private static IComponentManagementService cms;  
  
 public static void main(String[] args) {  
 String[] defargs = new String[] { "-gui", "false", "-welcome", "true", "-cli", "false", "-printpass", "false",  
 "-awareness", "false" };  
 IFuture<IExternalAccess> plataform = jadex.base.Starter.createPlatform(defargs);  
 ThreadSuspendable sus = new ThreadSuspendable();  
 platform = plataform.get(sus);  
 cms = SServiceProvider.getService(platform.getServiceProvider(), IComponentManagementService.class,  
 RequiredServiceInfo.SCOPE\_PLATFORM).get(sus);  
  
 Map<String, Object> agParam = new HashMap<String, Object>();  
 agParam.put("type", PingPongAgentBDI.PONG);  
 agParam.put("index", 0);  
 new AgentDeployer(agParam, "bin/agents/PingPongAgentBDI.class", cms).deploy();  
   
   
 agParam.put("type", PingPongAgentBDI.PING);  
 agParam.put("index", 1);  
 new AgentDeployer(agParam, "bin/agents/PingPongAgentBDI.class", cms).deploy();  
  
 }  
  
}

DirectoryFacilitator.java

package util;  
  
import java.util.HashMap;  
import jadex.bridge.IComponentIdentifier;  
  
public class DirectoryFacilitator {  
 private HashMap<String, IComponentIdentifier> yellow\_pages;  
 private static DirectoryFacilitator this\_instance;  
  
 private DirectoryFacilitator() {  
 yellow\_pages = new HashMap<String, IComponentIdentifier>();  
 }  
  
 public static DirectoryFacilitator getInstance() {  
 if (this\_instance == null) {  
 this\_instance = new DirectoryFacilitator();  
 }  
 return this\_instance;  
 }  
  
 public synchronized void registerAgent(String agentName, IComponentIdentifier AID) {  
 this\_instance.yellow\_pages.put(agentName, AID);  
 System.out.println(String.format("%s just registered itself! Now the directory has %d entries", agentName,  
 this\_instance.yellow\_pages.keySet().size()));  
 }  
  
 public IComponentIdentifier getAgentAID(String agentName) {  
 return this\_instance.yellow\_pages.get(agentName);  
 }  
}

AgentDeployer.java

package util;  
  
import java.util.Map;  
  
import jadex.bridge.IComponentIdentifier;  
import jadex.bridge.service.types.cms.CreationInfo;  
import jadex.bridge.service.types.cms.IComponentManagementService;  
import jadex.commons.future.ITuple2Future;  
  
public class AgentDeployer {  
  
 private Map<String, Object> param;  
 private String agentPath = "";  
 private IComponentManagementService cms;  
  
 public AgentDeployer(Map<String, Object> param, String agentPath, IComponentManagementService cms) {  
 this.param = param;  
 this.agentPath = agentPath;  
 this.cms = cms;  
 }  
  
 public ITuple2Future<IComponentIdentifier, Map<String, Object>> deploy() {  
 ITuple2Future<IComponentIdentifier, Map<String, Object>> agent = cms.createComponent(agentPath,  
 new CreationInfo(param));  
 Exception e = agent.getException();  
 if(e != null) {  
 e.printStackTrace();  
 }  
 return agent;  
 }  
}

PingPongAgentBDI.java

package util;  
  
import java.util.Map;  
  
import jadex.bridge.IComponentIdentifier;  
import jadex.bridge.service.types.cms.CreationInfo;  
import jadex.bridge.service.types.cms.IComponentManagementService;  
import jadex.commons.future.ITuple2Future;  
  
public class AgentDeployer {  
  
 private Map<String, Object> param;  
 private String agentPath = "";  
 private IComponentManagementService cms;  
  
 public AgentDeployer(Map<String, Object> param, String agentPath, IComponentManagementService cms) {  
 this.param = param;  
 this.agentPath = agentPath;  
 this.cms = cms;  
 }  
  
 public ITuple2Future<IComponentIdentifier, Map<String, Object>> deploy() {  
 ITuple2Future<IComponentIdentifier, Map<String, Object>> agent = cms.createComponent(agentPath,  
 new CreationInfo(param));  
 Exception e = agent.getException();  
 if(e != null) {  
 e.printStackTrace();  
 }  
 return agent;  
 }  
}

AgentBDI.java

package agents;  
  
import jadex.micro.annotation.Agent;  
import jadex.micro.annotation.AgentCreated;  
  
@Agent  
public class AgentBDI {  
  
 @AgentCreated  
 public void created() {  
 System.out.println("hello");  
 }  
  
}

**Висновок:**За допомогою бібліотеки JADEX можна реалізувати агентів, що виконуються в агентно-орієнтованій моделі BDI (Belief-Desire-Intention). Особливості створення агентів у JADEX полягають у використанні ADF-файлу для опису агента, його планів, переконань та можливостей.

Плани в агентному моделюванні використовуються для опису поведінки агента в різних ситуаціях, переконання - для опису знань та уявлень агента про світ, а можливості - для опису фізичних можливостей агента.

В JADEX існує декілька видів планів, переконань та можливостей, що відрізняються за своєю природою та функціональністю. Для роботи з планами необхідно підключити бібліотеку PlanAPI, яка містить методи для керування планами агента.

У JADEX можливість запуску агентів забезпечується Java Control Center, який містить засоби для керування та моніторингу агентів. Механізм взаємодії в межах платформи JADEX забезпечується за допомогою передачі повідомлень між агентами.

Отже, вивчення основних принципів моделі BDI, опису планів, переконань та можливостей агентів є важливим для розуміння принципів агентно-орієнтованого програмування. Реалізація агентів за допомогою бібліотеки JADEX дозволяє створювати агентів, що відповідають цим принципам.