**Modul 9**

* Gui Penyedia Travel

package Modul\_9;

import java.awt.BorderLayout;

import java.awt.Dimension;

import java.awt.GridLayout;

import java.awt.Toolkit;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.WindowAdapter;

import java.awt.event.WindowEvent;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JTextField;

/\*\*

\*

\* @author ozora

\*/

public class GuiPenyediaTravel extends JFrame

{

private static final long serialVersionUID = 1L;

private PenyediaTravel myAgent;

private JTextField titleField, jbField, priceField;

GuiPenyediaTravel (PenyediaTravel a) {

super(a.getLocalName());

myAgent = a;

JPanel p = new JPanel();

p.setLayout(new GridLayout(3, 2));

p.add(new JLabel("Kota Tujuan:"));

titleField = new JTextField(15);

p.add(titleField);

p.add(new JLabel("Jadwal Berangkat:"));

jbField = new JTextField(15);

p.add(jbField);

p.add(new JLabel("Biaya:"));

priceField = new JTextField(15);

p.add(priceField);

getContentPane().add(p, BorderLayout.CENTER);

JButton addButton = new JButton("Tambah");

addButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent ev) {

try {

String title = titleField.getText().trim();

String jadwalBerangkat = jbField.getText().trim();

String price = priceField.getText().trim();

String[] hari = {"Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu", "Minggu"};

int jb=0;

for (int i = 0; i < hari.length; i++) {

if (jadwalBerangkat.equals(hari[i])) {

jb = i;

}

}

myAgent.updateCatalogue(title, price, jb);

titleField.setText("");

priceField.setText("");

jbField.setText("");

} catch (Exception e) {

JOptionPane.showMessageDialog(GuiPenyediaTravel.this, "Invalid values. " + e.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

}

}

});

p = new JPanel();

p.add(addButton);

getContentPane().add(p, BorderLayout.SOUTH);

addWindowListener(new WindowAdapter() {

@Override

public void windowClosing(WindowEvent e) {

myAgent.doDelete();

}

});

setResizable(false);

}

public void showGui() {

pack();

Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();

int centerX = (int) screenSize.getWidth() / 2;

int centerY = (int) screenSize.getHeight() / 2;

setLocation(centerX - getWidth() / 2, centerY - getHeight() / 2);

super.setVisible(true);

}

}

* Pemesan

package Modul\_9;

import com.sun.org.apache.xalan.internal.lib.ExsltStrings;

import jade.core.Agent;

import jade.core.AID;

import jade.core.behaviours.\*;

import jade.lang.acl.ACLMessage;

import jade.lang.acl.MessageTemplate;

import jade.domain.DFService;

import jade.domain.FIPAException;

import jade.domain.FIPAAgentManagement.DFAgentDescription;

import jade.domain.FIPAAgentManagement.ServiceDescription;

import java.text.SimpleDateFormat;

import java.util.Calendar;

import javax.swing.JOptionPane;

/\*\*

\*

\* @author ozora

\*/

public class Pemesan extends Agent

{

private static final long serialVersionUID = 1L;

// The title of the book to buy

private String targetNamaTravel;

private String tujuan;

// The list of known seller agents

private AID[] sellerAgents;

// Put agent initializations here

protected void setup() {

// Print a welcome message

System.out.println("Hello! Buyer-agent " + getAID().getName() + " is ready.");

// Get the title of the book to buy as a start-up argument

Object[] args = getArguments();

if (args != null && args.length > 0) {

targetNamaTravel = (String) args[0];

tujuan=JOptionPane.showInputDialog("Tujuan anda");

System.out.println("Target travel adalah " + targetNamaTravel);

System.out.println("Tujuan "+tujuan);

// Add a TickerBehaviour that schedules a request for seller agents every minute

addBehaviour(new TickerBehaviour(this, 1500) {

private static final long serialVersionUID = 1L;

protected void onTick() {

System.out.println("Mencoba untuk order " + targetNamaTravel);

// Update the list of seller agents

DFAgentDescription template = new DFAgentDescription();

ServiceDescription sd = new ServiceDescription();

sd.setType("travel-order");

template.addServices(sd);

try {

DFAgentDescription[] result = DFService.search(myAgent, template);

System.out.println("Found the following seller agents:");

sellerAgents = new AID[result.length];

for (int i = 0; i < result.length; ++i) {

sellerAgents[i] = result[i].getName();

System.out.println(sellerAgents[i].getName());

}

} catch (FIPAException fe) {

fe.printStackTrace();

}

// Perform the request

myAgent.addBehaviour(new RequestPerformer());

}

});

} else {

// Make the agent terminate

System.out.println("Spesifikasi travel yang dituju tidak ada");

// doDelete();

}

}

// Put agent clean-up operations here

protected void takeDown() {

// Printout a dismissal message

System.out.println("Travel-agent " + getAID().getName() + " berhenti.");

}

/\*\*

\* Inner class RequestPerformer. This is the behaviour used by Book-buyer

\* agents to request seller agents the target book.

\*/

private class RequestPerformer extends Behaviour {

private static final long serialVersionUID = 1L;

private AID bestSeller; // The agent who provides the best offer

private int bestPrice; // The best offered price

private int bestHari;

private AID bestSeller2; // The agent who provides the best offer

private int bestPrice2; // The best offered price

private int bestHari2;

private int paramHariKe;

private int repliesCnt = 0; // The counter of replies from seller agents

private MessageTemplate mt; // The template to receive replies

private int step = 0;

public void action() {

String[] hari = {"Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu", "Minggu"};

SimpleDateFormat ambilHari = new SimpleDateFormat("EEEE");

Calendar cal = Calendar.getInstance();

String aH = ambilHari.format(cal.getTime());

for (int a = 0; a < hari.length; a++) {

if (aH.equals(hari[a])) {

paramHariKe = a;

}

}

switch (step) {

case 0:

// Send the cfp to all sellers

ACLMessage cfp = new ACLMessage(ACLMessage.CFP);

for (int i = 0; i < sellerAgents.length; ++i) {

cfp.addReceiver(sellerAgents[i]);

}

cfp.setContent(targetNamaTravel);

cfp.setConversationId("travel-trade");

cfp.setReplyWith("cfp" + System.currentTimeMillis()); // Unique value

myAgent.send(cfp);

// Prepare the template to get proposals

mt = MessageTemplate.and(MessageTemplate.MatchConversationId("travel-trade"),

MessageTemplate.MatchInReplyTo(cfp.getReplyWith()));

step = 1;

break;

case 1:

// Receive all proposals/refusals from seller agents

ACLMessage reply = myAgent.receive(mt);

if (reply != null) {

// Reply received

if (reply.getPerformative() == ACLMessage.PROPOSE) {

// This is an offer

// int price = Integer.parseInt(reply.getContent());

String a = reply.getContent();

String b[] = a.split("/");

int harga = Integer.parseInt(b[0]);

int harike = Integer.parseInt(b[1]);

if (bestSeller == null || harike < bestHari2) {

bestPrice2 = harga;

bestSeller2 = reply.getSender();

bestHari2 = harike;

if (harike < bestHari2 && harga < bestPrice2) {

// This is the best offer at present

bestPrice2 = harga;

bestSeller2 = reply.getSender();

bestHari2 = harike;

}

}

if (bestSeller == null || (harike > paramHariKe && harike < bestHari)) {

bestPrice = harga;

bestSeller = reply.getSender();

bestHari = harike;

if (harike < bestHari && harga < bestPrice) {

// This is the best offer at present

bestPrice2 = harga;

bestSeller2 = reply.getSender();

bestHari2 = harike;

}

}

}

repliesCnt++;

if (repliesCnt >= sellerAgents.length) {

if (bestSeller == null) {

bestPrice = bestPrice2;

bestSeller = bestSeller2;

bestHari = bestHari2;

}

// We received all replies

step = 2;

}

} else {

block();

}

break;

case 2:

// Send the purchase order to the seller that provided the best offer

ACLMessage order = new ACLMessage(ACLMessage.ACCEPT\_PROPOSAL);

order.addReceiver(bestSeller);

order.setContent(targetNamaTravel);

order.setConversationId("travel-trade");

order.setReplyWith("order" + System.currentTimeMillis());

myAgent.send(order);

// Prepare the template to get the purchase order reply

mt = MessageTemplate.and(

MessageTemplate.MatchConversationId("travel-trade"),

MessageTemplate.MatchInReplyTo(order.getReplyWith()));

step = 3;

break;

case 3:

// Receive the purchase order reply

reply = myAgent.receive(mt);

if (reply != null) {

// Purchase order reply received

if (reply.getPerformative() == ACLMessage.INFORM) {

// Purchase successful. We can terminate

System.out.println(targetNamaTravel

+ " order berhasil dari agent "

+ reply.getSender().getName());

System.out.println("Biaya = " + bestPrice + " Hari " + hari[bestHari]);

// myAgent.doDelete();

} else {

System.out.println("Order gagal: travel yang dituju sudah habis diorder.");

}

step = 4;

} else {

block();

}

break;

}

}

public boolean done() {

if (step == 2 && bestSeller == null) {

System.out.println("Attempt failed: " + targetNamaTravel + " not available for sale");

}

return ((step == 2 && bestSeller == null) || step == 4);

}

} // End of inner class RequestPerformer

}

* Penyedia Travel

package Modul\_9;

import jade.core.Agent;

import jade.core.Agent;

import jade.core.behaviours.CyclicBehaviour;

import jade.core.behaviours.OneShotBehaviour;

import jade.domain.DFService;

import jade.domain.FIPAException;

import jade.domain.FIPAAgentManagement.DFAgentDescription;

import jade.domain.FIPAAgentManagement.ServiceDescription;

import jade.lang.acl.ACLMessage;

import jade.lang.acl.MessageTemplate;

import java.util.Hashtable;

/\*\*

\*

\* @author ozora

\*/

public class PenyediaTravel extends Agent

{

private static final long serialVersionUID = 1L;

// The catalogue of books for sale (maps the title of a book to its price)

private Hashtable<String, String> catalogue;

// The GUI by means of which the user can add books in the catalogue

private GuiPenyediaTravel myGui;

// Put agent initializations here

protected void setup() {

// Create the catalogue

catalogue = new Hashtable<String, String>();

// Create and show the GUI

myGui = new GuiPenyediaTravel(this);

myGui.showGui();

// Register the travel-order service in the yellow pages

DFAgentDescription dfd = new DFAgentDescription();

dfd.setName(getAID());

ServiceDescription sd = new ServiceDescription();

sd.setType("travel-order");

sd.setName("JADE-book-trading");

dfd.addServices(sd);

try {

DFService.register(this, dfd);

} catch (FIPAException fe) {

fe.printStackTrace();

}

// Add the behaviour serving queries from buyer agents

addBehaviour(new OfferRequestsServer());

// Add the behaviour serving purchase orders from buyer agents

addBehaviour(new PurchaseOrdersServer());

}

// Put agent clean-up operations here

@Override

protected void takeDown() {

// Deregister from the yellow pages

try {

DFService.deregister(this);

} catch (FIPAException fe) {

fe.printStackTrace();

}

// Close the GUI

myGui.dispose();

// Printout a dismissal message

System.out.println("Penyedia " + getAID().getName() + " berhenti.");

}

/\*\*

\* This is invoked by the GUI when the user adds a new book for sale

\*/

public void updateCatalogue(final String tittle, final String price, final int jb) {

addBehaviour(new OneShotBehaviour() {

/\*\*

\*

\*/

private static final long serialVersionUID = 1L;

@Override

public void action() {

final String gb=price+"/"+jb;

catalogue.put(tittle, gb);

String[] hari = {"Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu", "Minggu"};

System.out.println(tittle + " keberangkatan "+hari[jb]+" dimasukkan kedalam catalogue dengan biaya = " + price);

}

});

}

private class OfferRequestsServer extends CyclicBehaviour {

private static final long serialVersionUID = 1L;

@Override

public void action() {

MessageTemplate mt = MessageTemplate.MatchPerformative(ACLMessage.CFP);

ACLMessage msg = myAgent.receive(mt);

if (msg != null) {

// CFP Message received. Process it

String title = msg.getContent();

ACLMessage reply = msg.createReply();

String price = catalogue.get(title);

if (price != null) {

// The requested book is available for sale. Reply with the price

reply.setPerformative(ACLMessage.PROPOSE);

reply.setContent(price);

} else {

// The requested book is NOT available for sale.

reply.setPerformative(ACLMessage.REFUSE);

reply.setContent("not-available");

}

myAgent.send(reply);

} else {

block();

}

}

} // End of inner class OfferRequestsServer

private class PurchaseOrdersServer extends CyclicBehaviour {

private static final long serialVersionUID = 1L;

@Override

public void action() {

MessageTemplate mt = MessageTemplate.MatchPerformative(ACLMessage.ACCEPT\_PROPOSAL);

ACLMessage msg = myAgent.receive(mt);

if (msg != null) {

// ACCEPT\_PROPOSAL Message received. Process it

String title = msg.getContent();

ACLMessage reply = msg.createReply();

String price = catalogue.remove(title);

if (price != null) {

reply.setPerformative(ACLMessage.INFORM);

System.out.println(title + " terpesan dari pelanggan " + msg.getSender().getName());

} else {

// The requested book has been sold to another buyer in the meanwhile .

reply.setPerformative(ACLMessage.FAILURE);

reply.setContent("not-available");

}

myAgent.send(reply);

} else {

block();

}

}

} // End of inner class OfferRequestsServer

}

**Modul 10**

* Client

package Modul10;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.rmi.Naming;

import java.rmi.NotBoundException;

public class client {

int a, b;

String tanya;

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

public void menu() throws IOException, NotBoundException {

fungsi f = (fungsi) Naming.lookup("rmi://localhost:1099/data");

System.out.println("Kalkulator");

System.out.println("1. Penjumlahan");

System.out.println("2. Pengurangan");

System.out.println("3. Perkalian");

System.out.println("4. Pembagian");

System.out.print("Pilih aksi : ");

int pilih = Integer.parseInt(br.readLine());

switch (pilih) {

case 1:

System.out.print("Masukkan Angka Pertama\t:");

a = Integer.parseInt(br.readLine());

System.out.print("Masukkan Angka kedua\t:");

b = Integer.parseInt(br.readLine());

System.out.println("Hasilnya adalah : " + f.tambah(a, b));

new client().keluar();

break;

case 2:

System.out.print("Masukkan Angka Pertama\t:");

a = Integer.parseInt(br.readLine());

System.out.print("Masukkan Angka kedua\t:");

b = Integer.parseInt(br.readLine());

System.out.println("Hasilnya adalah " + f.kurang(a, b));

new client().keluar();

break;

case 3:

System.out.print("Masukkan Angka Pertama\t:");

a = Integer.parseInt(br.readLine());

System.out.print("Masukkan Angka kedua\t:");

b = Integer.parseInt(br.readLine());

System.out.println("Hasilnya adalah " + f.kali(a, b));

new client().keluar();

break;

case 4:

System.out.print("Masukkan Angka Pertama\t:");

a = Integer.parseInt(br.readLine());

System.out.print("Masukkan Angka kedua\t:");

b = Integer.parseInt(br.readLine());

System.out.println("Hasilnya adalah " + f.bagi(a, b));

new client().keluar();

break;

}

if (pilih>4){

System.out.println("Pilihan tidak ada");

menu();

}

}

public void keluar() throws IOException, NotBoundException {

System.out.println("Apakah Anda ingin melakukan operasi lagi? ya atau tidak");

tanya = br.readLine();

if (tanya.equals("ya")) {

new client().menu();

} else if (tanya.equals("tidak")) {

System.exit(0);

}

}

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

client c = new client();

c.menu();

}

}

* Fungsi

package Modul10;

import java.rmi.\*;

public interface fungsi extends Remote {

public int tambah(int a, int b) throws RemoteException;

public int kurang(int a, int b) throws RemoteException;

public int kali(int a, int b) throws RemoteException;

public int bagi(int a, int b) throws RemoteException;

}

* Operasi

package Modul10;

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

public class operasi extends UnicastRemoteObject implements fungsi {

int hasil;

public operasi() throws RemoteException {

}

@Override

public int tambah(int a, int b) throws RemoteException {

hasil = a + b;

return hasil;

}

@Override

public int kurang(int a, int b) throws RemoteException {

hasil = a - b;

return hasil;

}

@Override

public int kali(int a, int b) throws RemoteException {

hasil = a \* b;

return hasil;

}

@Override

public int bagi(int a, int b) throws RemoteException {

hasil = a / b;

return hasil;

}

}

* Serverkal

package Modul10;

import java.rmi.RemoteException;

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

public class serverkal {

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

Registry registry = LocateRegistry.createRegistry(1099);

operasi ops = new operasi();

registry.rebind("data", ops);

System.out.println("Server is Running");

}

}