Mateusz Adamczyk 235834
Tomasz Murawka 226077

Wrocław, 13.11.2018
Prowadzący: dr inż. Paweł Głuchowski

Inżynieria oprogramowania - sprawozdanie

Etapy 5 - 7 – Opracowanie diagramu klas oraz diagramów sekwencji dla wybranych przypadków użycia.

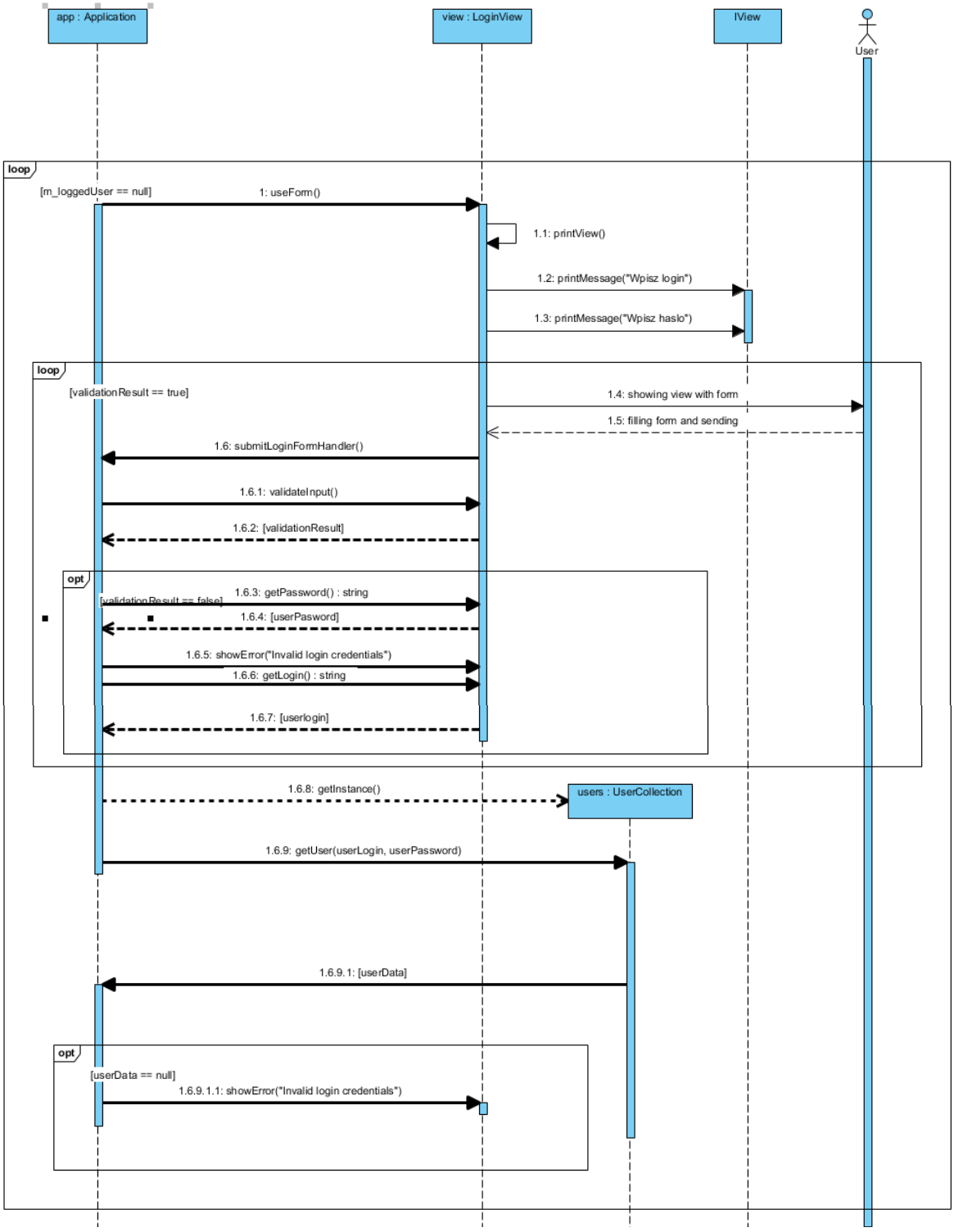
W ramach piątego, szóstego i siódmego etapu laboratorium grupa utworzyła diagram klas występujących w aplikacji, a także diagramy sekwencji dla dwóch wybranych przypadków użycia. Diagramy zostały przygotowane za pomocą narzędzia Visual Paradigm.

1. Diagram klas

Z racji swoich rozmiarów, przygotowany diagram klas załączono jako **dodatek 1** do sprawozdania.

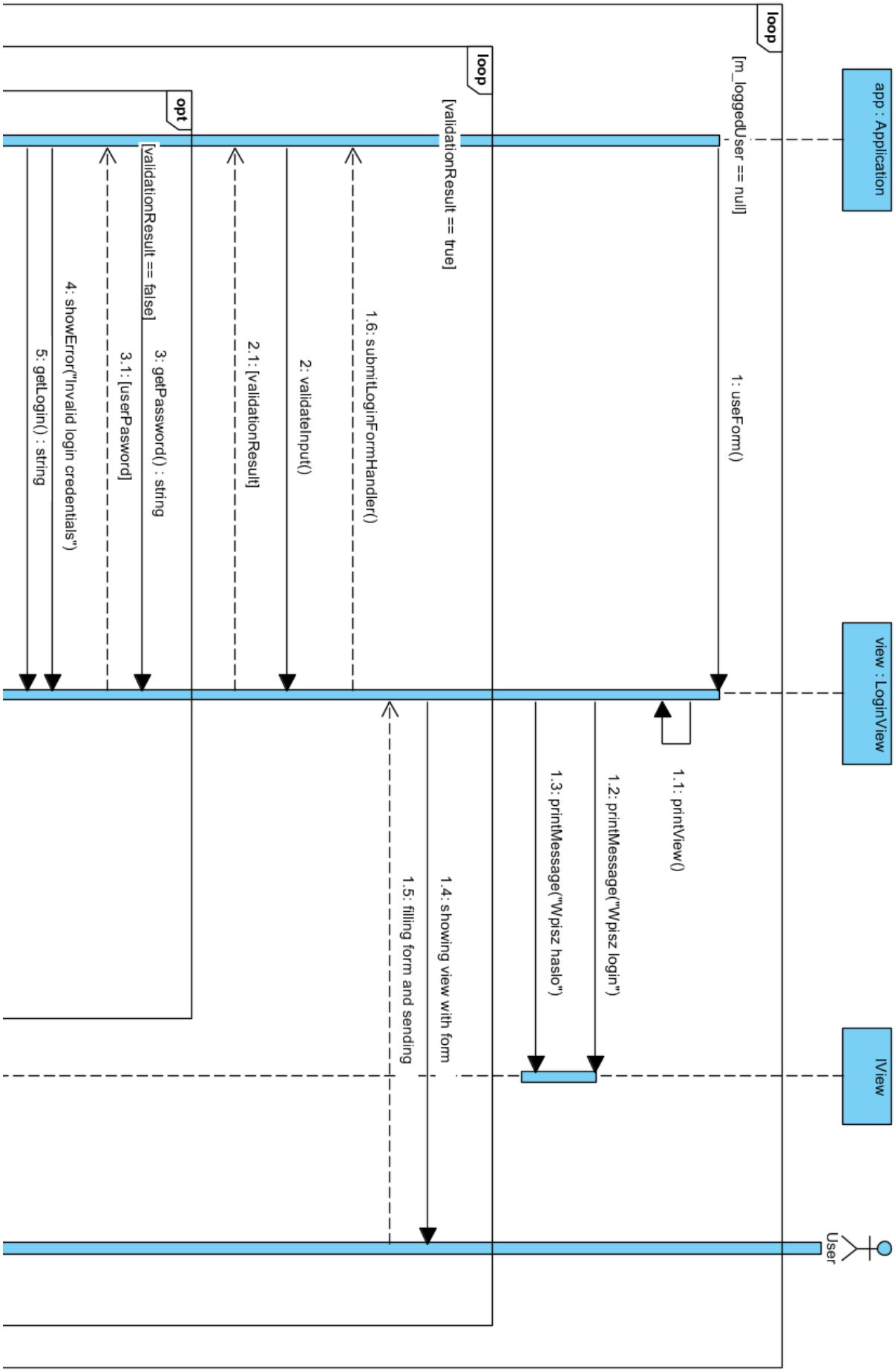
2. Diagramy sekwencji

a) Diagram przedstawiający logowanie do systemu:

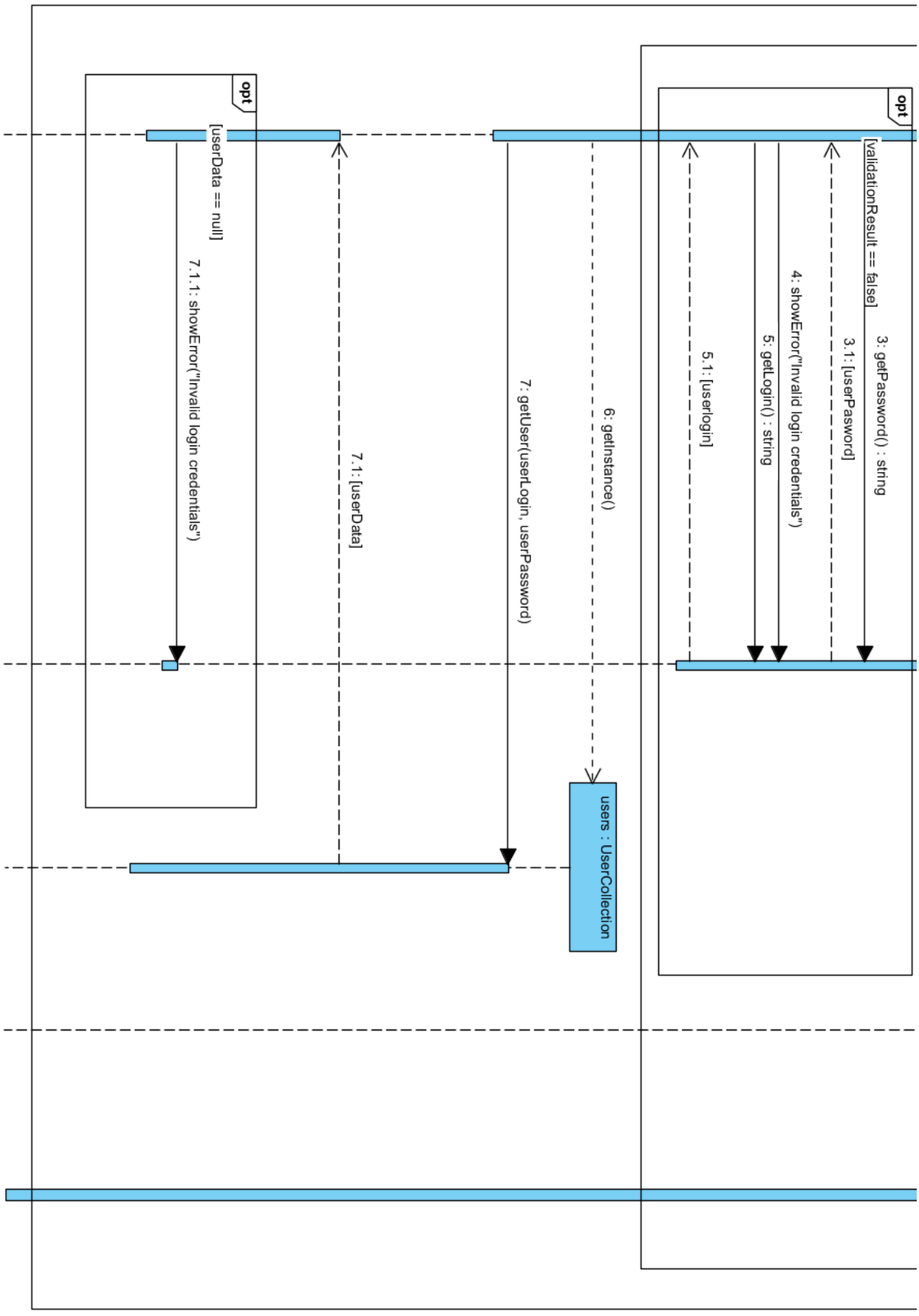


Rysunek 1 - Diagram sekwencji logowania do systemu

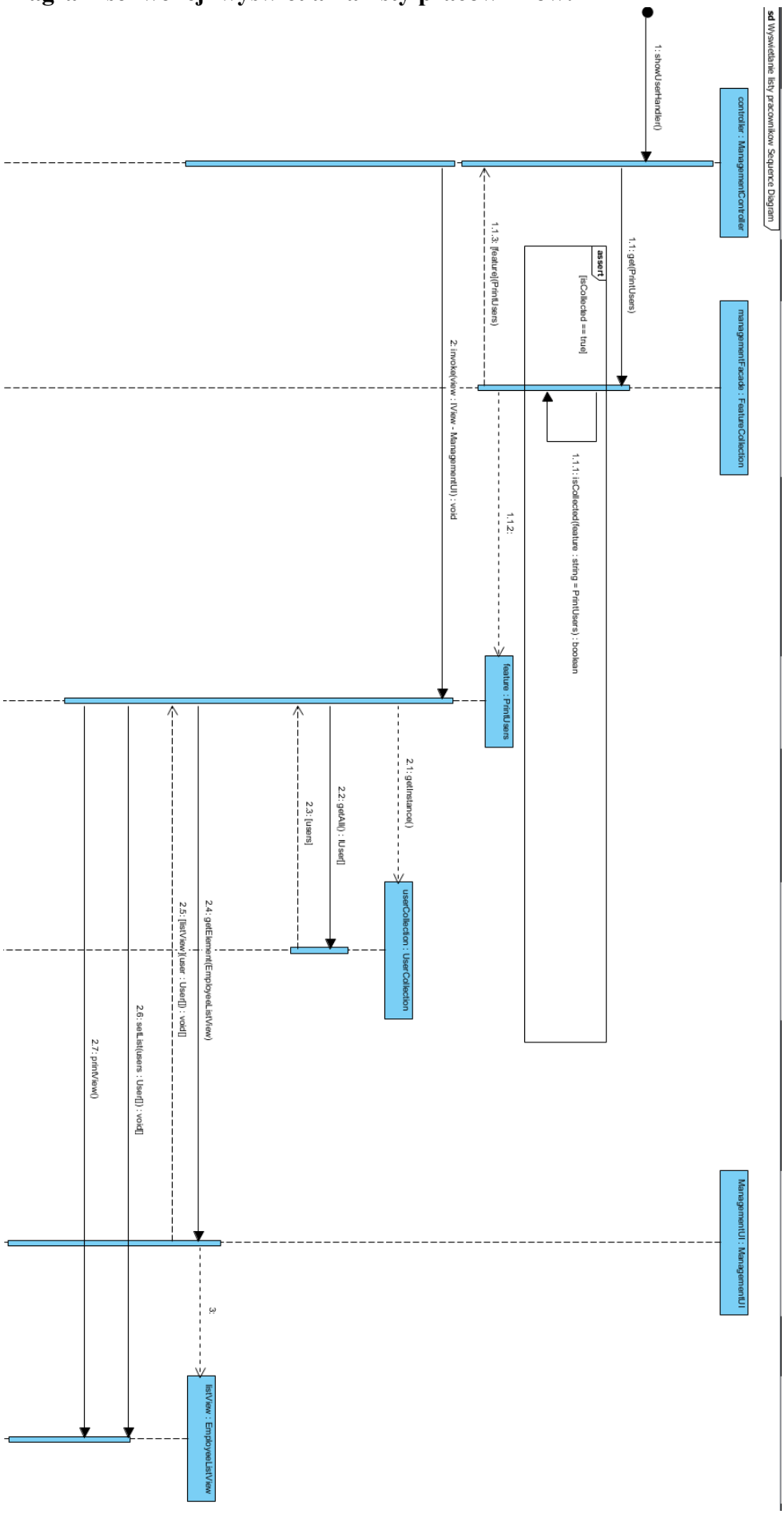
Przybliżenie 1:



Przybliżenie 2:

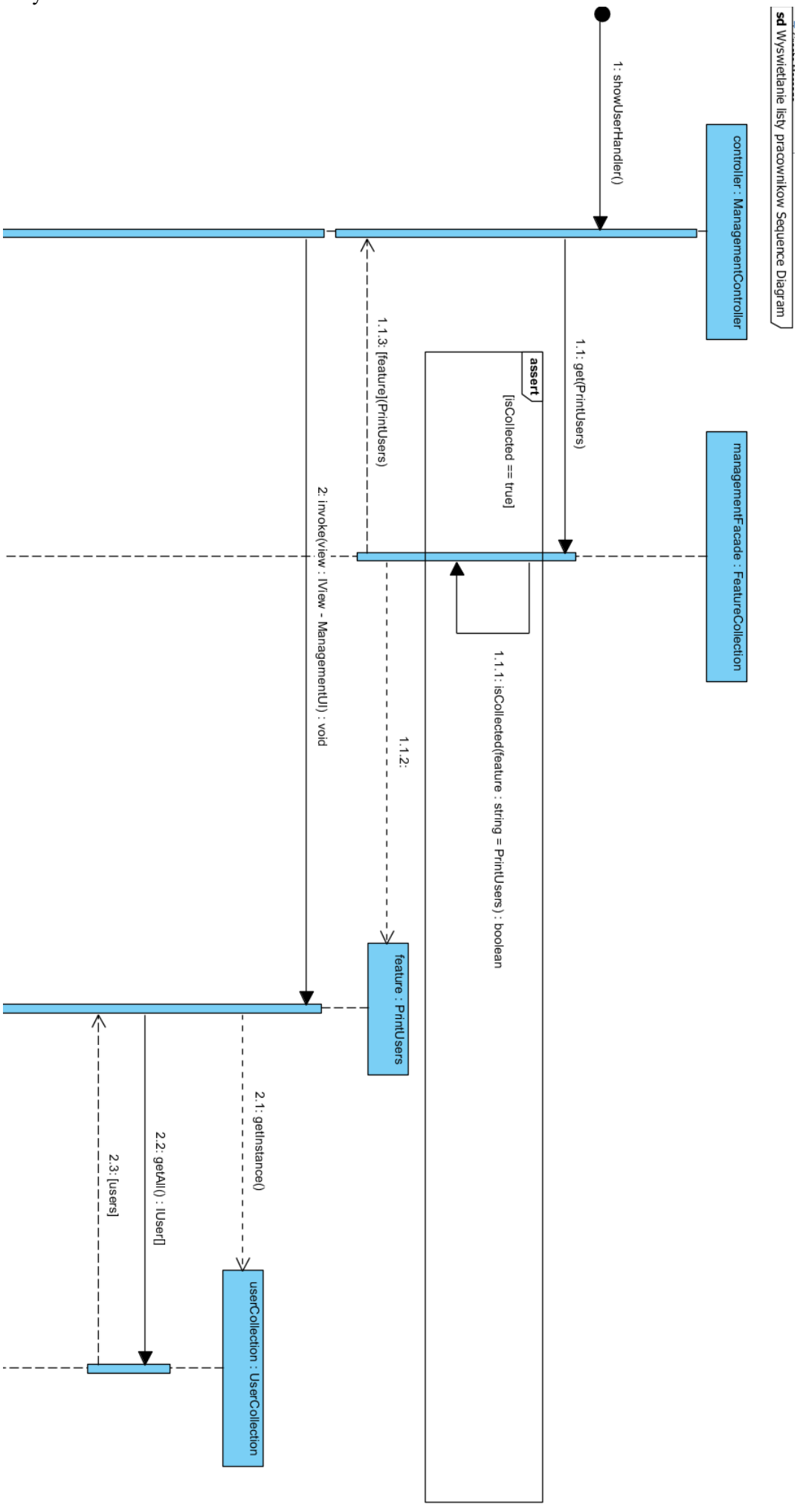


b) Diagram sekwencji wyświetlania listy pracowników:

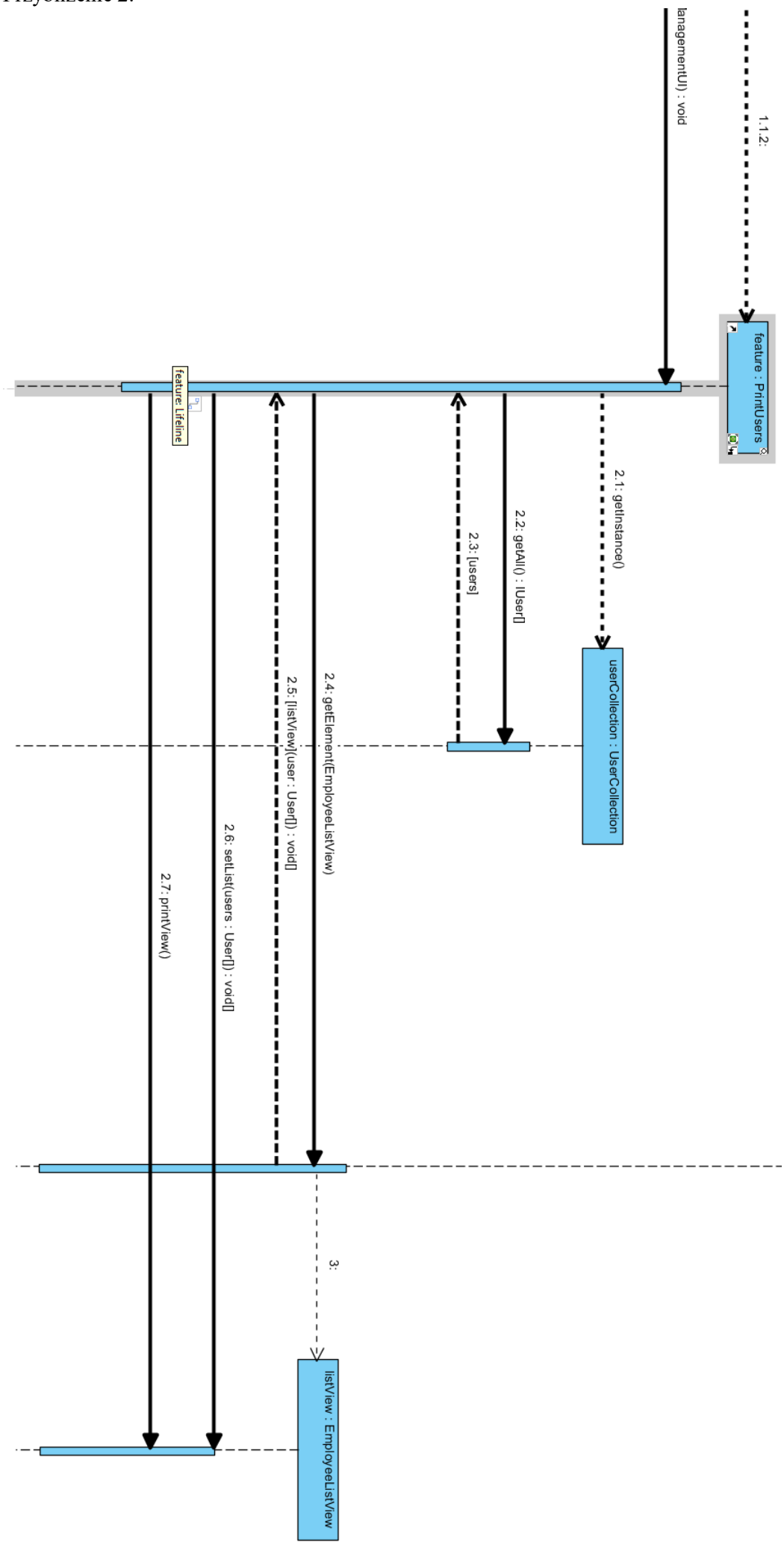


Rysunek 2 - Diagram sekwencji wyświetlania listy pracowników

Przybliżenie 1:



Przybliżenie 2:



3. Wykonanie kodu źródłowego wynikającego z diagramu klas [dodatek 1]

```
package Views;
import java.util.AbstractList;
public interface IListView extends IView {
	public void setList(AbstractList<Object> list);
}
```

```
package Views;
import java.util.AbstractList;
import java.util.ArrayList;
import Models.User;
public class EmployeeListView implements IListView {
	private ArrayList<User> m_list;
	public void printView() {
		for(User user : m_list) {
			IView.printMessage(user.getLogin());
		}
	}
	public IView getElement(String p_parameter) {
		throw new UnsupportedOperationException();
	}
	@SuppressWarnings("unchecked")
	@Override
	public void setList(AbstractList<Object> p_list) {
		m_list = (ArrayList<User>)(ArrayList<?>)p_list;
	}
}
```

```
package Views;
public class LoginView implements IView, IForm {
	String m_password;
	String m_login;
	ILoginHandler m_loginSubmitHandler;
	public void setLoginSubmitHandler(ILoginHandler p_handler) {
		m_loginSubmitHandler = p_handler;
	}
	public String getPassword() {
		return m_password;
	}
	public String getLogin() {
		return m_login;
	}
	public void printView() {
		IView.printMessage("System zarzadzania MPK");
		IView.printMessage("Zaloguj sie by kontynuowac");
	}
	public IView getElement(String p_parameter) {
		return null;
	}
	@Override
	public boolean validateInput() {
		return !m_password.isEmpty() && !m_login.isEmpty();
	}
	@Override
	public void showError(String p_errorMessage) {
		IView.printMessage(p_errorMessage);
	}
```

```
	@Override
	public void useForm() {
		printView();
		IView.printMessage("Wpisz login");
		m_login = IForm.readConsole();
		IView.printMessage("Wpisz haslo");
		m_password = IForm.readPassword();
		m_loginSubmitHandler.submitLoginFormHandler();
	}
}

package Views;
public interface ILoginHandler {
	public void submitLoginFormHandler();
}

package Views;
import java.io.BufferedReader;
import java.io.Console;
import java.io.IOException;
import java.io.InputStreamReader;
public interface IForm {
	public boolean validateInput();
	public void showError(String p_errorMessage);
	public void useForm();
	public static void tryClear() {
		try {
			Runtime.getRuntime().exec("cls");
		} catch (IOException e) {
		}
	}
	public static String readConsole() {
		Console console = System.console();
		String input = "";
		if(console == null) {
			BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
			try {
				input = br.readLine();
			} catch (IOException e) {
			}
		}
		else {
			input  = console.readLine().toString();
		}
		return input;
	}
	public static String readPassword() {
		Console console = System.console();

		String password = "";
		if(console == null) {
			BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
			try {
				password = br.readLine();
			} catch (IOException e) {
			}
		}
		else {
			password = console.readPassword().toString();
		}
		return password;
	}
}

package Views;
public interface IManagementMenuHandler {
	public void showUsersHandler();
}
```

```
package Views;
public interface IView {
	public void printView();
	public IView getElement(String p_parameter);
	public static void printMessage(String p_message) {
		System.out.println(p_message);}
}

package ManagementSystem;
import java.util.HashMap;
import Core.Controller;
import Views.EmployeeListView;
import Views.IForm;
import Views.IManagementMenuHandler;
import Views.IView;
public class ManagementUI implements IView, IForm {
	private IManagementMenuHandler m_menuHandler;
	private Controller m_controller;
	private HashMap<String, IView> m_nestedViews;
	public ManagementUI() {
		m_nestedViews = new HashMap<String, IView>();
		m_nestedViews.put("EmployeeListView", new EmployeeListView());}
	@Override
	public void printView() {
		IView.printMessage("0 : Wyloguj");
		IView.printMessage("1 : Wyswietl pracownikow");}
	@Override
	public IView getElement(String p_parameter) {
		return m_nestedViews.get(p_parameter);}
	@Override
	public boolean validateInput() {
		return true;}
	@Override
	public void showError(String p_errorMessage) {
		IView.printMessage(p_errorMessage);}
	@Override
	public void useForm() {
		IView.printMessage("Wybrana opcja:");
		String userInput = IForm.readConsole();
		switch(userInput){
		case "0":
			m_controller.Logout();
			break;
		case "1":
			m_menuHandler.showUsersHandler();
			break;}
	}
	public void setMenuHandler(IManagementMenuHandler p_menuHandler) {
		m_menuHandler = p_menuHandler;}
	public void setController(Controller p_controller) {
		m_controller = p_controller;}
}
```

```
package ManagementSystem;
import Core.Controller;
import Core.ControllerStatus;
import Core.FeatureCollectionBuilder;
import Features.FeatureCollection;
import Features.FeatureEntry;
import Views.IManagementMenuHandler;
public class ManagementController implements Controller, IManagementMenuHandler {
	private ManagementUI m_UI;
	private FeatureCollection features;
	private ControllerStatus m_returnStatus = ControllerStatus.Running;
	public ManagementController() {
		m_UI = new ManagementUI();
		m_UI.setMenuHandler(this);
		m_UI.setController(this);
		FeatureCollectionBuilder builder = new FeatureCollectionBuilder();
		features = builder.build(new ManagementFeatureStrategy()); }
	public void Logout() {
		m_returnStatus = ControllerStatus.Logout;}
	@Override
	public ControllerStatus Run() {
		while(m_returnStatus == ControllerStatus.Running) {
			m_UI.printView();
			m_UI.useForm();}
		return m_returnStatus;}
	@Override
	public void Exit() {
		m_returnStatus = ControllerStatus.Exit;}
	@Override
	public void showUsersHandler() {
	features.get(FeatureEntry.PrintUsers).invoke(m_UI.getElement("EmployeeListView"));}
}

package ManagementSystem;
import Core.BuilderStrategy;
import Features.FeatureEntry;
public class ManagementFeatureStrategy extends BuilderStrategy {
	public ManagementFeatureStrategy() {
		this.FeatureList.add(FeatureEntry.PrintUsers);}
}

package Features;
public enum FeatureEntry {
	AddMandate, ChangeMandateState,
	ChangeBusStatus, PrintTimetable,
	PrintBuses, AddInspectionScheaduleEntry,
	DeleteInspectionScheaduleEntry, ModifyInspectionScheadule,
	AddDriverScheaduleEntry, DeleteDriverScheaduleEntry,
	ModifyDriverScheadule, AddUser,
	PrintUsers, ModifyUserData,
	ModifyUserRole, DeleteUser,
	PrintDriverScheadule, PrintInspectionScheadule
}

package Features;
import Views.IView;
public interface Feature {
	public void invoke(IView p_view);
}
```

```
package Features;
import Views.IListView;
import Views.IView;
import java.util.ArrayList;
import Models.UserCollection;
public class PrintUsers implements Feature {
	@SuppressWarnings("unchecked")
	public void invoke(IView p_view) {
		UserCollection users = UserCollection.getInstance();
		IListView listView = (IListView)p_view;
		listView.setList((ArrayList<Object>)(ArrayList<?>)users.getAll();
		listView.printView();}
}

package Features;
import java.util.HashMap;
public class FeatureCollection {
	private HashMap<FeatureEntry, Feature> m_features = new HashMap<FeatureEntry, Feature>();
	public Feature get(FeatureEntry p_command) {
		return m_features.get(p_command);}
	public boolean isCollected(FeatureEntry p_feature) {
		return m_features.containsKey(p_feature);}
	public void put(Feature p_feature, FeatureEntry p_key) {
		m_features.put(p_key, p_feature);}
}

package Models;
import java.util.ArrayList;
import java.util.HashMap;
import Models.User;
public class UserCollection {
	private static UserCollection instance = null;
	protected UserCollection() {
		m_data = new HashMap<String, User>();
		User admin = new User();
		admin.setLogin("admin");
		admin.setPassword("admin");
		admin.setRole(UserRole.Administrator);
		addUser(admin);};
	public static UserCollection getInstance() {
		if(instance == null) {
			instance = new UserCollection();}
		return instance;}
	HashMap<String,User> m_data;
	public User getUser(String p_login) {
		return m_data.get(p_login);}
	public boolean exists(User p_user) {
		return m_data.containsKey(p_user.getLogin());}
	public boolean addUser(User p_newUser) {
		if(exists(p_newUser)) {
			return false;}
		else {	m_data.put(p_newUser.getLogin(), p_newUser);	return true;}}
	public ArrayList<User> getAll() {
		return new ArrayList<User>(m_data.values());}
}
```

```
package Models;
import Core.IHasher;
import Models.UserRole;
public class User {
	private String m_login;
	private UserRole m_role;
	private String m_password;
	public String getStorageKey() {
		return getLogin();}
	public String getLogin() {
		return m_login;}
	public void setLogin(String p_newLogin) {
		m_login = p_newLogin;}
	public UserRole getRole() {
		return m_role;}
	public void setRole(UserRole m_role) {
		this.m_role = m_role;}
	public String getPassword() {
		return m_password;}
	public void setPassword(String m_password) {
		IHasher hasher = IHasher.getCurrentHasher();
		this.m_password = hasher.hash(m_password);}
}

package Models;
import Features.PrintTimetable;
public class Timetable {
	public PrintTimetable m_unnamed_PrintTimetable_;
	public TimetableEntry m_unnamed_TimetableEntry_;
	public void getTimetable() {
		throw new UnsupportedOperationException();}
	public void addToTimetable(Object p_timeTable) {
		throw new UnsupportedOperationException();}
}

package Models;
public class TimetableEntry {
	public Timetable m_unnamed_Timetable_;
}

package Models;
public enum UserRole {
	Client
	,Management
	,Driver
	,Inspector
	,Administrator
}

package Models;
import Features.ChangeMandateState;
import Features.AddMandate;
public class MandateCollection {
	public ChangeMandateState m_unnamed_ChangeMandateState_;
	public AddMandate m_unnamed_AddMandate_;
	public Mandate m_unnamed_Mandate_;
}

package Models;
public class Mandate {
	public MandateCollection m_unnamed_MandateCollection_;
}
```

```
package Models;
import Features.PrintBuses;
import Features.ChangeBusStatus;
public class BusCollection {
	public PrintBuses m_unnamed_PrintBuses_;
	public ChangeBusStatus m_unnamed_ChangeBusStatus_;
	public Bus m_unnamed_Bus_;
}

package Models;
public class Bus {
	public BusCollection m_unnamed_BusCollection_;
}

package Models;
import Features.DeleteDriverScheaduleEntry;
import Features.AddDriverScheaduleEntry;
import Features.PrintDriverScheadule;
import Features.ModifyDriverScheadule;
public class DriverScheadule {
	public DeleteDriverScheaduleEntry m_unnamed_DeleteDriverScheaduleEntry_;
	public AddDriverScheaduleEntry m_unnamed_AddDriverScheaduleEntry_;
	public PrintDriverScheadule m_unnamed_PrintDriverScheadule_;
	public ModifyDriverScheadule m_unnamed_ModifyDriverScheadule_;
	public DriverScheaduleEntry m_unnamed_DriverScheaduleEntry_;
}

package Models;
public class DriverScheaduleEntry implements ScheaduleEntry {
	public DriverScheadule m_unnamed_DriverScheadule_;
}

package Models;
import Features.DeleteInspectionScheaduleEntry;
import Features.AddInspectionScheaduleEntry;
import Features.PrintInspectionScheadule;
import Features.ModifyInspectionScheadule;
public class InspectionScheadule {
	public DeleteInspectionScheaduleEntry m_unnamed_DeleteInspectionScheaduleEntry_;
	public AddInspectionScheaduleEntry m_unnamed_AddInspectionScheaduleEntry_;
	public PrintInspectionScheadule m_unnamed_PrintInspectionScheadule_;
	public ModifyInspectionScheadule m_unnamed_ModifyInspectionScheadule_;
	public InspectorScheaduleEntry m_unnamed_InspectorScheaduleEntry_;
}


package Models;
public class InspectorScheaduleEntry implements ScheaduleEntry {
	public InspectionScheadule m_unnamed_InspectionScheadule_;
}

package Models;
public interface ScheaduleEntry {
}

package Core;
import Core.ControllerStatus;
public interface Controller {
	public ControllerStatus Run();
	public void Logout();
	public void Exit();
}

package Core;
import Models.UserCollection;
import Views.LoginView;
import Views.ILoginHandler;
```

```
import Models.User;
import Core.ControllerFactory;
public class Application implements ILoginHandler {
	private LoginView m_view;
	private User m_loggedUser;
	public static void main(String[] args) {
		Application app = new Application();
		app.Run();}
	public Application() {
		m_view = new LoginView();
		m_view.setLoginSubmitHandler(this);}
	public void Login() {
		while(m_loggedUser == null) {
			m_view.useForm();}}
	public void Run() {
		Login();
		Controller controller = ControllerFactory.getController(m_loggedUser.getRole());
		ControllerStatus status = controller.Run();
		m_loggedUser = null;
		if(status == ControllerStatus.Logout) {Run();}
		else if(status == ControllerStatus.Exit) {return;}
	}
	public void submitLoginFormHandler() {
		if(!m_view.validateInput()) {
			m_view.showError("Invalid data");
			return;}
		UserCollection users = UserCollection.getInstance();
		String login = m_view.getLogin();
		String password = m_view.getPassword();
		User user = users.getUser(login);
		if(user == null) {
			m_view.showError("Nieprawidlowy login lub haslo");
			return;}
		IHasher hasher = new DummyHasher();
		if(user.getPassword().compareTo(hasher.hash(password)) != 0) {
			m_view.showError("Nieprawidlowy login lub haslo");
			return;  }
		m_loggedUser = user;}
}

package Core;
public interface IMergable {
	public void Marge(IMergable toMarge);
}

package Core;
import java.util.ArrayList;
import Core.IMergable;
import Features.FeatureEntry;
public class BuilderStrategy implements IMergable {
	public ArrayList<FeatureEntry> FeatureList = new ArrayList<FeatureEntry>();
	public void Marge(IMergable p_toMarge) {
		BuilderStrategy otherStrategy = (BuilderStrategy)p_toMarge;
		for(FeatureEntry feature : otherStrategy.FeatureList) {
			if(!FeatureList.contains(feature)) {
				FeatureList.add(feature);}}}
}
```

```
package Core;
import Models.UserRole;
import AdministratorSystem.AdminController;
import ClientSystem.ClientController;
import DriverSystem.DriverController;
import InspectorSystem.InspectorController;
import ManagementSystem.ManagementController;
public class ControllerFactory {
	public static Controller getController(UserRole p_currentRole) {
		Controller toReturn;
		switch(p_currentRole) {
		case Administrator:
			toReturn = new AdminController();
			break;
		case Client:
			toReturn =  new ClientController();
			break;
		case Driver:
			toReturn =  new DriverController();
			break;
		case Inspector:
			toReturn =  new InspectorController();
			break;
		case Management:
			toReturn =  new ManagementController();
			break;
		default:
			toReturn = new ClientController();
			break;}
		return toReturn;}
}

package Core;
public interface IHasher {
	public String hash(String p_toHash);
	public static IHasher getCurrentHasher() {
		return new DummyHasher();}
}

package Core;
import Core.IHasher;
public class DummyHasher implements IHasher {
	public String hash(String p_toHash) {
		return p_toHash;}
}

package Core;
public enum ControllerStatus {
	Logout, Exit, Running
}

package Core;
import java.util.HashMap;
import Features.AddDriverScheaduleEntry;
import Features.AddInspectionScheaduleEntry;
import Features.AddMandate;
import Features.AddUser;
import Features.ChangeBusStatus;
import Features.ChangeMandateState;
import Features.DeleteDriverScheaduleEntry;
import Features.DeleteInspectionScheaduleEntry;
import Features.DeleteUser;
import Features.Feature;
import Features.FeatureCollection;
import Features.FeatureEntry;
import Features.ModifyDriverScheadule;
import Features.ModifyInspectionScheadule;
import Features.ModifyUserData;
```

```
import Features.ModifyUserRole;
import Features.PrintBuses;
import Features.PrintDriverScheadule;
import Features.PrintInspectionScheadule;
import Features.PrintTimetable;
import Features.PrintUsers;

public class FeatureCollectionBuilder {
	HashMap<FeatureEntry, Feature> m_featureMap;
	public FeatureCollectionBuilder() {
		m_featureMap = new HashMap<FeatureEntry, Feature>();
		mapFeatures();}
	public FeatureCollection build(BuilderStrategy p_strategy) {
		FeatureCollection collection = new FeatureCollection();

		for(FeatureEntry feature : p_strategy.FeatureList) {
			collection.put(m_featureMap.get(feature), feature);}
		return collection;}

	private void mapFeatures() {
		mapUserFeatures();
		mapDriverScheaduleFeatures();
		mapInspectorScheaduleFeatures();
		mapTimetableFeatures();
		mapBusesFeatures();
		mapMandateFeatures();}

	private void mapMandateFeatures() {
		m_featureMap.put(FeatureEntry.AddMandate, new AddMandate());
		m_featureMap.put(FeatureEntry.ChangeMandateState, new ChangeMandateState());}

	private void mapBusesFeatures() {
		m_featureMap.put(FeatureEntry.ChangeBusStatus, new ChangeBusStatus());
		m_featureMap.put(FeatureEntry.PrintBuses, new PrintBuses());}

	private void mapTimetableFeatures() {
		m_featureMap.put(FeatureEntry.PrintTimetable, new PrintTimetable());}

	private void mapInspectorScheaduleFeatures() {
		m_featureMap.put(FeatureEntry.AddInspectionScheaduleEntry, new
AddInspectionScheaduleEntry());
		m_featureMap.put(FeatureEntry.DeleteInspectionScheaduleEntry, new
DeleteInspectionScheaduleEntry());
		m_featureMap.put(FeatureEntry.ModifyInspectionScheadule, new
ModifyInspectionScheadule());
		m_featureMap.put(FeatureEntry.PrintInspectionScheadule, new PrintInspectionScheadule());}
	private void mapDriverScheaduleFeatures() {
		m_featureMap.put(FeatureEntry.AddDriverScheaduleEntry, new
AddDriverScheaduleEntry());
		m_featureMap.put(FeatureEntry.DeleteDriverScheaduleEntry, new
DeleteDriverScheaduleEntry());
		m_featureMap.put(FeatureEntry.ModifyDriverScheadule, new ModifyDriverScheadule());
		m_featureMap.put(FeatureEntry.PrintDriverScheadule, new PrintDriverScheadule());    }

	private void mapUserFeatures() {
		m_featureMap.put(FeatureEntry.AddUser, new AddUser());
		m_featureMap.put(FeatureEntry.PrintUsers, new PrintUsers());
		m_featureMap.put(FeatureEntry.ModifyUserData, new ModifyUserData());
		m_featureMap.put(FeatureEntry.ModifyUserRole, new ModifyUserRole());
		m_featureMap.put(FeatureEntry.DeleteUser, new DeleteUser());}
}

package ClientSystem;
import Core.Controller;
import Core.ControllerStatus;
public class ClientController implements Controller {
	public ClientUI m_unnamed_ClientUI_;
	public ClientFeatureStrategy m_unnamed_ClientFeatureStrategy_;
```

```
		@Override
		public ControllerStatus Run() {
			Views.IView.printMessage("User logged in");
			Views.IForm.readConsole();
			return ControllerStatus.Logout;}
		@Override
		public void Logout() {
			// TODO Auto-generated method stub		}
		@Override
		public void Exit() {
			// TODO Auto-generated method stub		}
}

package ClientSystem;
import Core.FeatureCollectionBuilder;
import Core.BuilderStrategy;
public class ClientFeatureStrategy extends BuilderStrategy {
		public ClientController m_unnamed_ClientController_;
		public FeatureCollectionBuilder m_unnamed_FeatureCollectionBuilder_;
}

package ClientSystem;
import Views.IView;
public class ClientUI {
		public ClientController m_unnamed_ClientController_;
		public IView m_unnamed_IView_;
}
```