Mateusz Adamczyk 235834 Wrocław, 13.11.2018

Tomasz Murawka 226077 Prowadzący: dr inż. Paweł Głuchowski

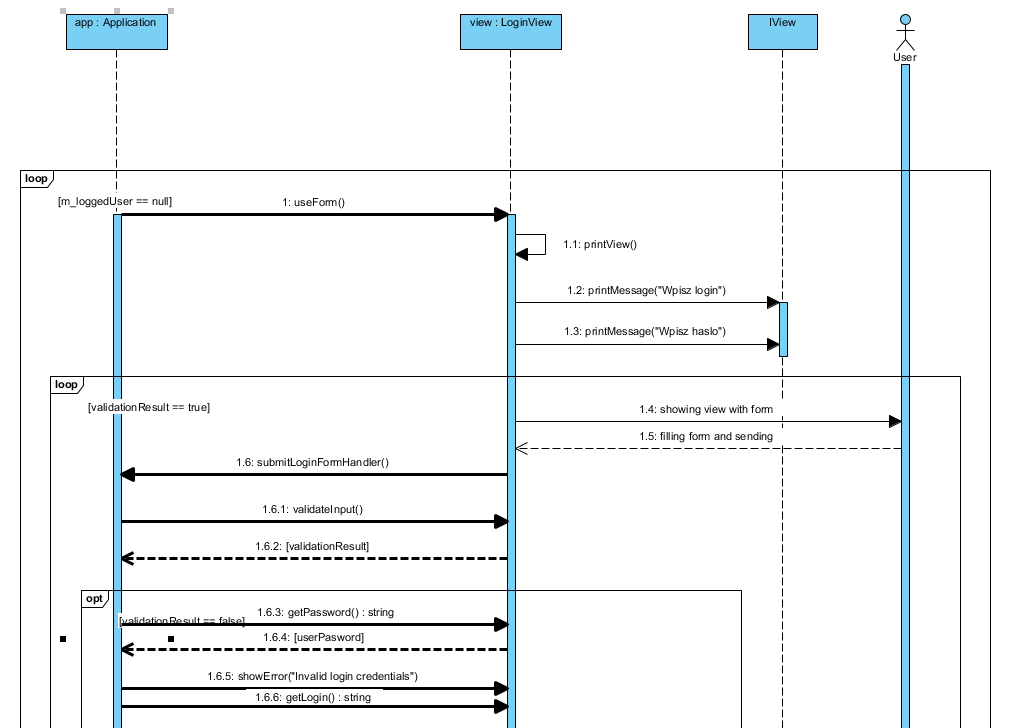
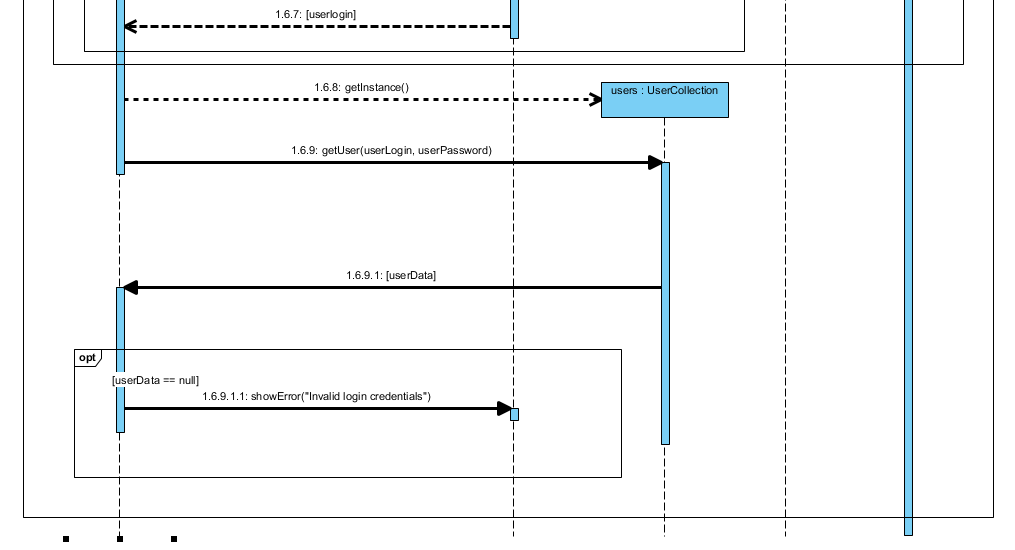
Inżynieria oprogramowania - sprawozdanie

**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.

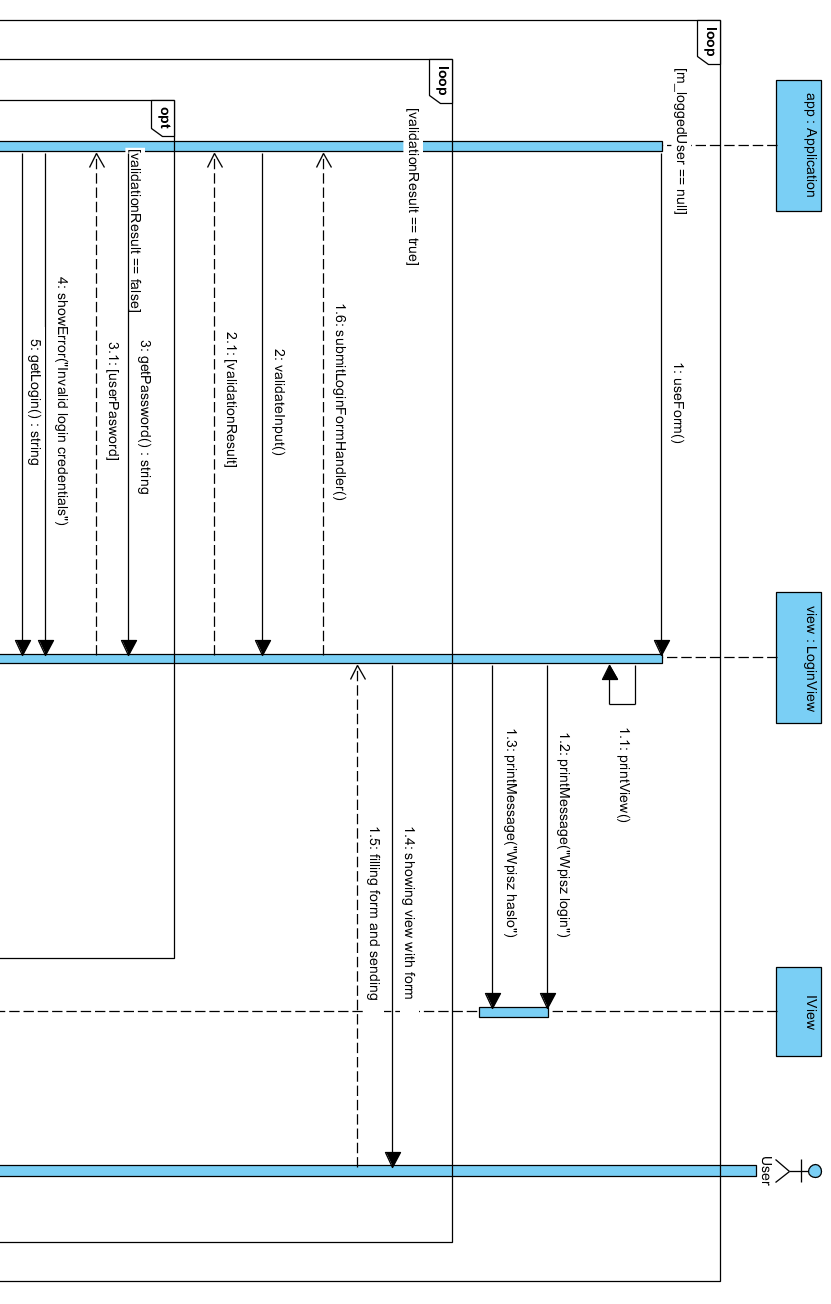
1. **Diagram klas**

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

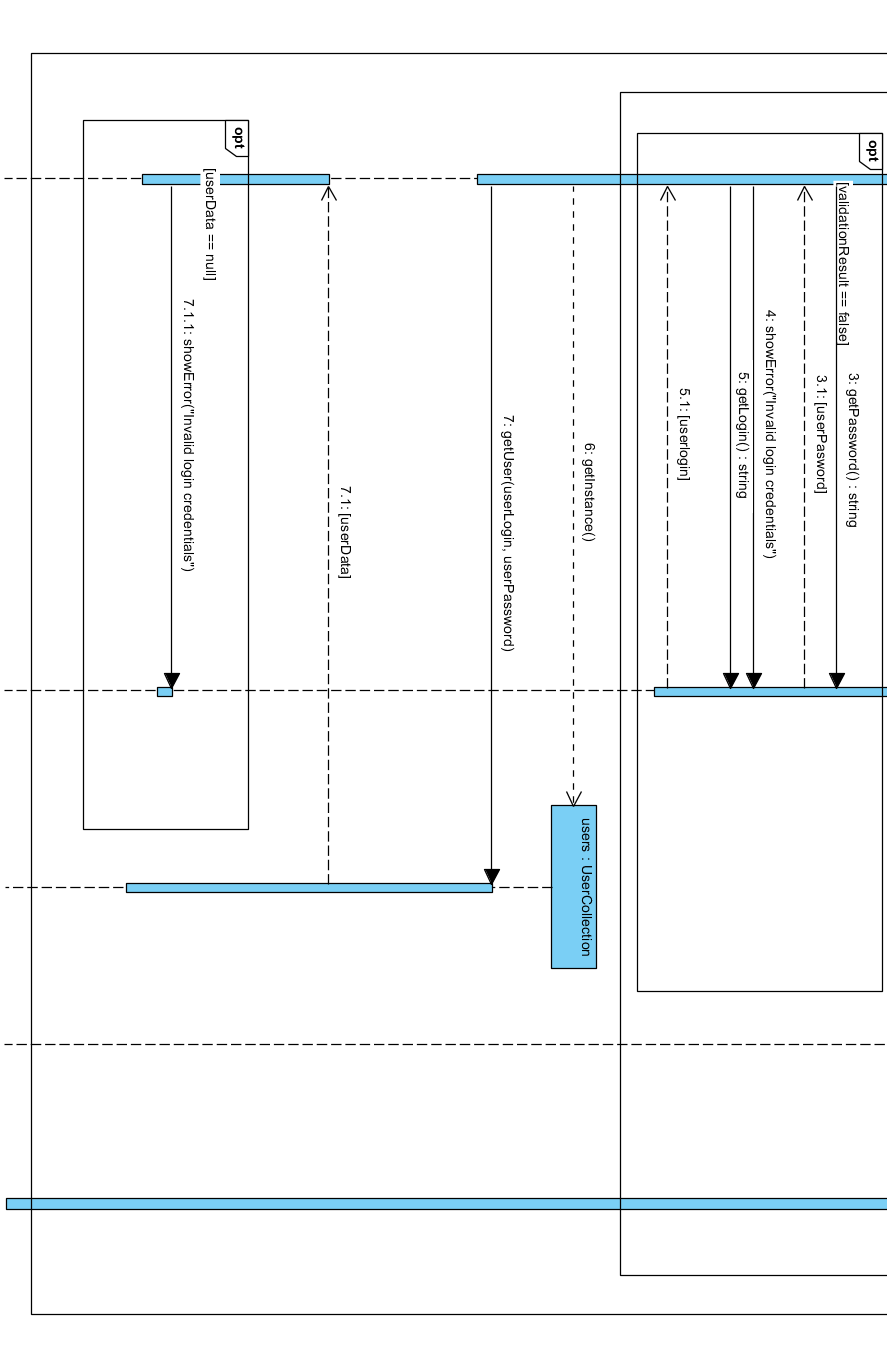
1. **Diagramy sekwencji**
2. **Diagram przedstawiający logowanie do systemu:**

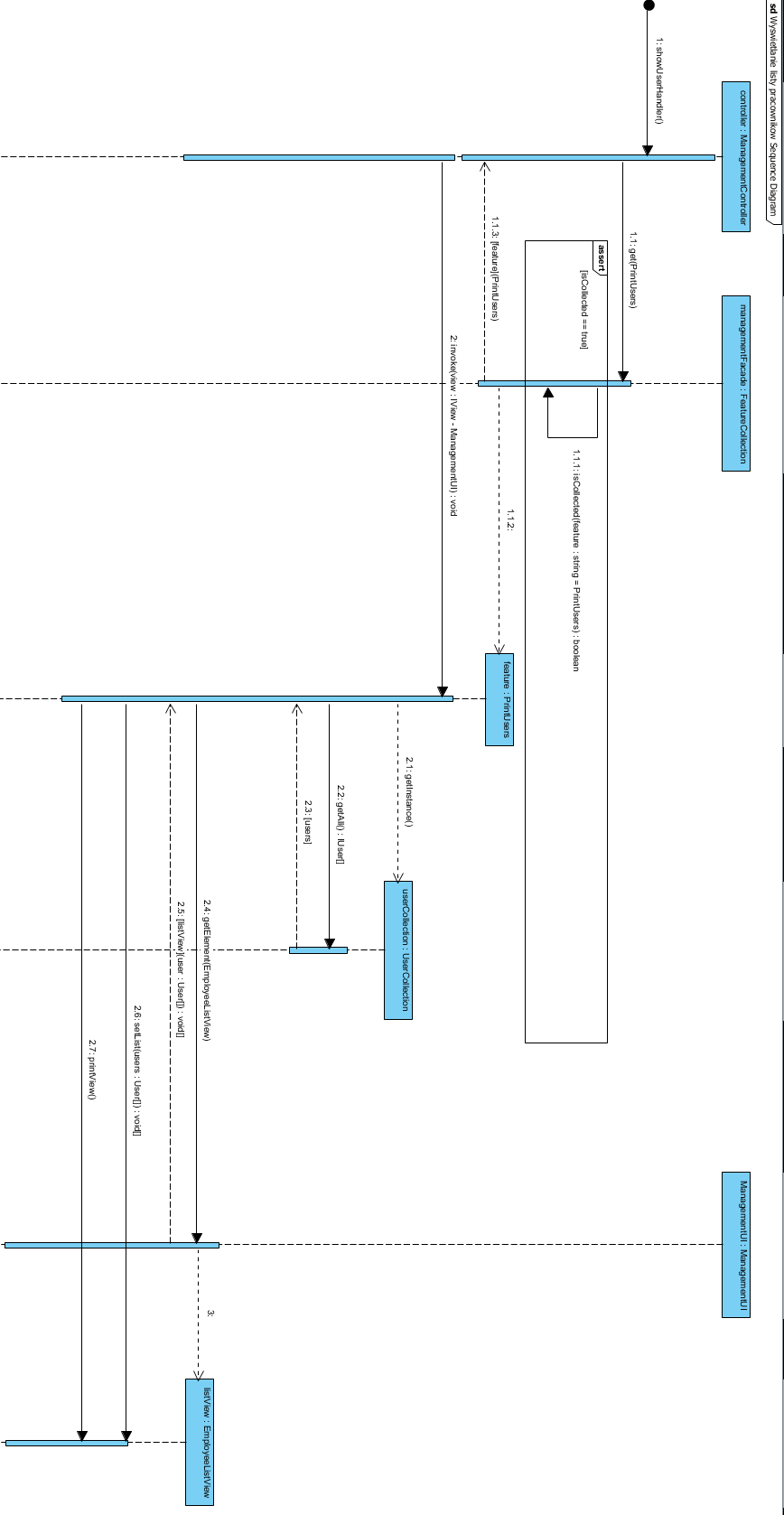
Rysunek 1 - Diagram sekwencji logowania do systemu

Przybliżenie 1:



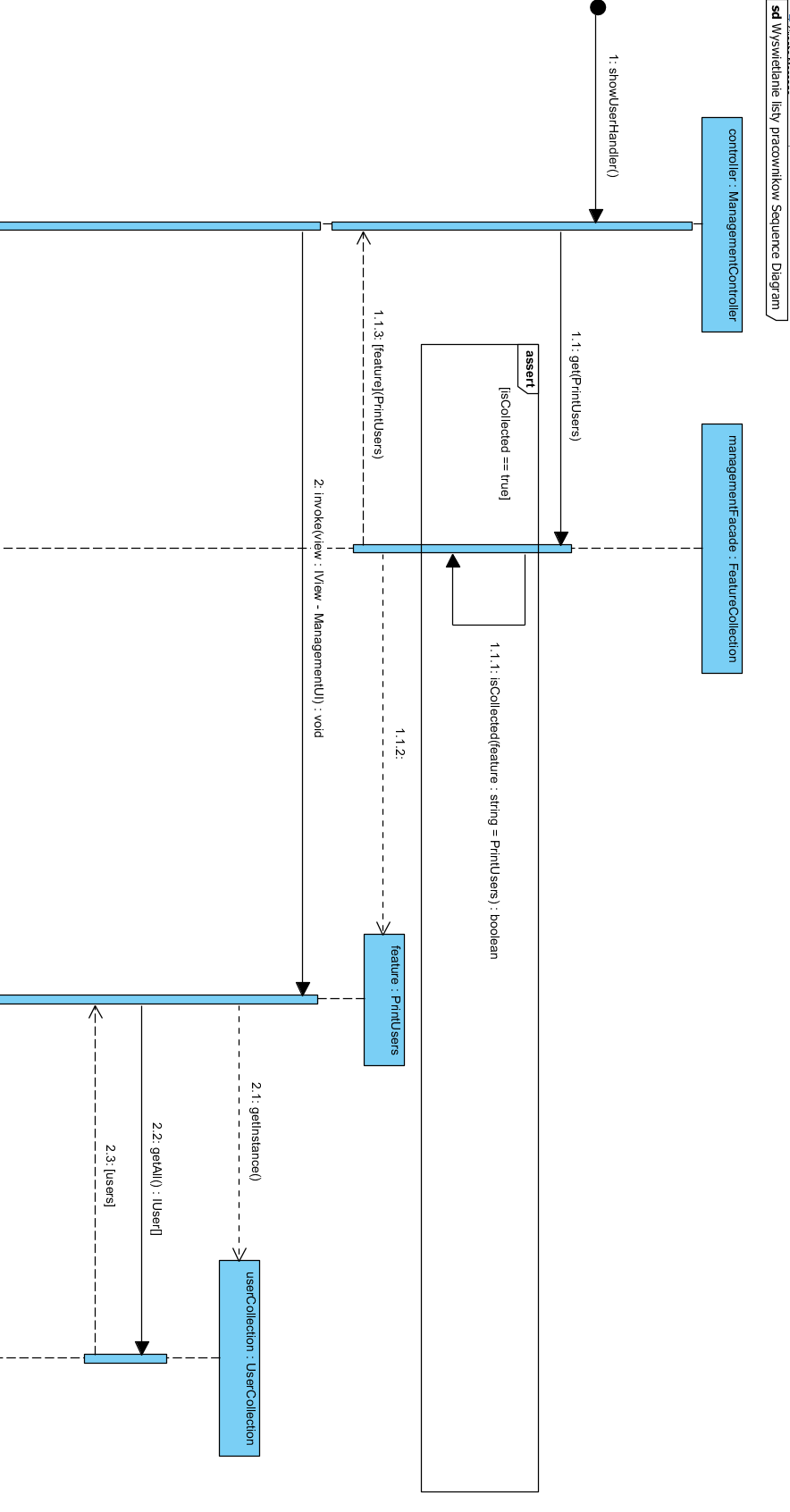
Przybliżenie 2:



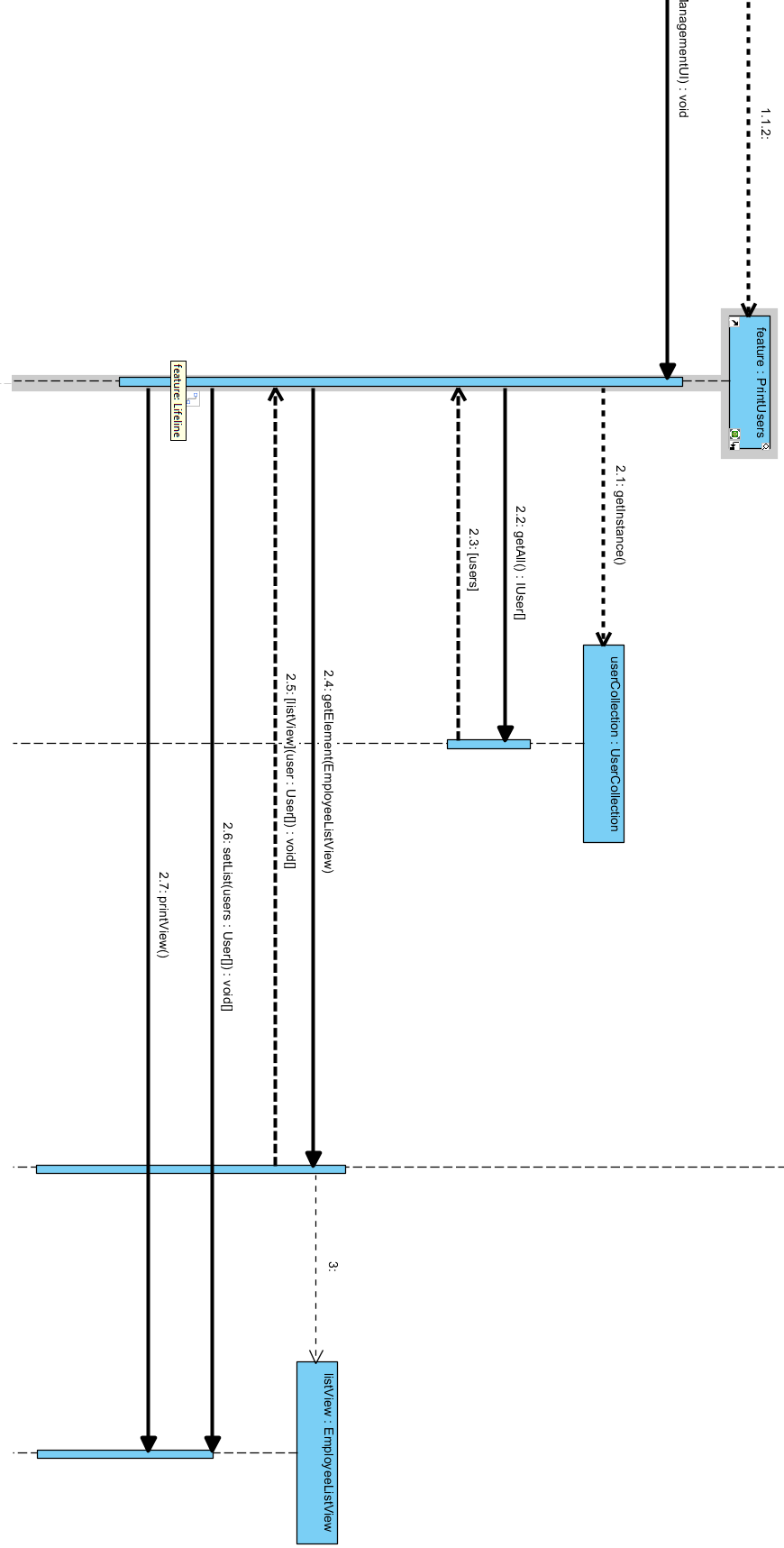
1. **Diagram sekwencji wyświetlania listy pracowników:**

Rysunek 2 - Diagram sekwencji wyświetlania listy pracowników

Przybliżenie 1:



Przybliżenie 2:



1. **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\_;

}