

МІНЕСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ технічний університет україни

«кпі ім. Ігоря Сікорського»

**Курсова робота**

**Системне программування**

Виконала:

студентка групи ВА-41

Луценко В.С.

Київ 2018

Зміст

Зміст………………………………………………………………………………………………2

Завдання…………………………………………………………………………………………..3

Лістинги…………………………………………………………………………….................4-41

UML……………………………………………………………………………….................42-46

Скріншоти результатів роботи……………………………………………………………..47-51

Література………………………………………………………………………….....................52

Завдання

Java. Методи програмування 2013 (И.Н. Блінов, В.С. Романчик), Глава 16, Завдання А:

14. Система Житлово-комунальні послуг(ЖЕК). **Орендар** відправляє **Заявку**, в якій вказує рід робіт, масштаб та бажаний час виконання. Диспетчер формує відповідну **Бригаду** та реєструє її в **Плані робіт**.

Лістинги

package ua.hcs.view;

import ua.hcs.controller.LoginButtonController;

import javax.swing.\*;

import java.awt.\*;

/\*\*

\* Фрейм авторизації.

\*/

public class Authorization extends JFrame {

private JLabel headerLabel;

private JLabel nameLabel;

private JTextField nameField;

private JLabel passwordLabel;

private JPasswordField passwordField;

private JButton loginButton;

public Authorization() {

headerLabel = new JLabel();

nameLabel = new JLabel();

nameField = new JTextField();

passwordLabel = new JLabel();

passwordField = new JPasswordField();

loginButton = new JButton();

setDefaultCloseOperation(WindowConstants.EXIT\_ON\_CLOSE);

setTitle("Додаток ЖЕК ~ Авторизація");

headerLabel.setFont(new Font("Tahoma", Font.BOLD, 15));

headerLabel.setHorizontalAlignment(SwingConstants.LEFT);

headerLabel.setText("Додаток ЖЕК");

nameLabel.setText("Введіть ім'я: ");

passwordLabel.setText("Введіть пароль: ");

loginButton.setText("Ввійти");

loginButton.addActionListener(new LoginButtonController(nameField, passwordField, this));

GroupLayout layout = new GroupLayout(getContentPane());

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addContainerGap()

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)

.addComponent(headerLabel)

.addComponent(nameLabel)

.addComponent(nameField)

.addComponent(passwordLabel)

.addComponent(passwordField)

.addComponent(loginButton))

.addContainerGap()));

layout.setVerticalGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addContainerGap()

.addComponent(headerLabel)

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

.addComponent(nameLabel)

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

.addComponent(nameField)

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

.addComponent(passwordLabel)

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

.addComponent(passwordField)

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

.addComponent(loginButton)

.addContainerGap()));

pack();

setLocationRelativeTo(null);

}

}

package ua.hcs.view;

import ua.hcs.controller.ConfirmBCreationButtonController;

import ua.hcs.model.Request;

import javax.swing.\*;

import java.awt.\*;

/\*\*

\* Фрейм формування бригади на запит.

\*/

public class BrigadeCreation extends JFrame {

private JLabel headerLabel;

private JLabel workTypeLabel;

private JLabel workScaleLabel;

private JLabel workPrefTimeLabel;

private JLabel brigadierNameLabel;

private JTextField brigadierNameTextField;

private JLabel workerNumberLabel;

private JTextField workerNumberTextField;

private JButton confirmButton;

private Request request;

private JFrame parentFrame;

public BrigadeCreation(Request request, JFrame parentFrame) {

headerLabel = new JLabel();

workScaleLabel = new JLabel();

workPrefTimeLabel = new JLabel();

workTypeLabel = new JLabel();

brigadierNameLabel = new JLabel();

brigadierNameTextField = new JTextField();

workerNumberLabel = new JLabel();

workerNumberTextField = new JTextField();

confirmButton = new JButton();

this.request = request;

this.parentFrame = parentFrame;

setDefaultCloseOperation(WindowConstants.DISPOSE\_ON\_CLOSE);

setTitle("Додаток ЖЕК ~ Формування бригади");

headerLabel.setFont(new Font("Tahoma", Font.BOLD, 15));

headerLabel.setHorizontalAlignment(SwingConstants.LEFT);

headerLabel.setText("Формування бригади для запиту " + request.getId() + ":\n");

workTypeLabel.setText("Тип робіт: \"" + request.getWorkType() + "\"");

workScaleLabel.setText("Масштаб робіт: \"" + request.getWorkScale() + "\"");

workPrefTimeLabel.setText("Бажаний час виконання: " + request.getWorkPrefferedTime() + " години");

brigadierNameLabel.setText("Введіть ім'я бригадира:");

workerNumberLabel.setText("Введіть необхідну кількість працівників:");

confirmButton.setText("Запланувати роботу");

confirmButton.addActionListener(new ConfirmBCreationButtonController(request,

brigadierNameTextField, workerNumberTextField, this));

GroupLayout layout = new GroupLayout(getContentPane());

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addContainerGap()

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)

.addComponent(headerLabel)

.addComponent(workTypeLabel)

.addComponent(workScaleLabel)

.addComponent(workPrefTimeLabel)

.addComponent(brigadierNameLabel)

.addComponent(brigadierNameTextField)

.addComponent(workerNumberLabel)

.addComponent(workerNumberTextField)

.addComponent(confirmButton))

.addContainerGap()));

layout.setVerticalGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addContainerGap()

.addComponent(headerLabel)

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

.addComponent(workTypeLabel)

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

.addComponent(workScaleLabel)

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

.addComponent(workPrefTimeLabel)

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

.addComponent(brigadierNameLabel)

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

.addComponent(brigadierNameTextField)

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

.addComponent(workerNumberLabel)

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

.addComponent(workerNumberTextField)

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

.addComponent(confirmButton)

.addContainerGap()));

pack();

setLocationRelativeTo(parentFrame);

}

}

package ua.hcs.view;

import ua.hcs.controller.AcceptRequestButtonController;

import ua.hcs.controller.DeleteWPRecordButtonController;

import ua.hcs.model.\*;

import ua.hcs.util.Const;

import javax.swing.\*;

import javax.swing.table.DefaultTableCellRenderer;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.util.ArrayList;

import java.util.List;

import java.util.stream.Collectors;

/\*\*

\* Фрейм диспетчера.

\*/

public class Dispatcher extends JFrame {

private JLabel headerLabel;

private JScrollPane requestsTableScrollPane;

private JTable requestsTable;

private JButton acceptRequestButton;

private JScrollPane workPlanTableScrollPane;

private JTable workPlanTable;

private JButton deleteWorkPlanRecordButton;

private List<Request> lastRequests = new ArrayList<>();

private List<WorkPlanRecord> lastWorkPlan = new ArrayList<>();

public Dispatcher() {

headerLabel = new JLabel();

requestsTableScrollPane = new JScrollPane();

requestsTable = new JTable();

acceptRequestButton = new JButton();

workPlanTableScrollPane = new JScrollPane();

workPlanTable = new JTable();

deleteWorkPlanRecordButton = new JButton();

setDefaultCloseOperation(WindowConstants.EXIT\_ON\_CLOSE);

setTitle("Додаток ЖЕК ~ Диспетчер");

headerLabel.setFont(new Font("Tahoma", Font.BOLD, 15));

headerLabel.setHorizontalAlignment(SwingConstants.LEFT);

headerLabel.setText("Панель диспетчера");

requestsTable.setModel(new RequestsTableModel());

requestsTable.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

((DefaultTableCellRenderer)requestsTable.getTableHeader().getDefaultRenderer())

.setHorizontalAlignment(JLabel.LEFT);

requestsTableScrollPane.setBorder(BorderFactory.createTitledBorder("Заяви орендарів"));

requestsTableScrollPane.setPreferredSize(new Dimension(Const.DISPATCHER\_FRAME\_WIDTH,

Const.DISPATCHER\_FRAME\_HEIGHT));

requestsTableScrollPane.setViewportView(requestsTable);

acceptRequestButton.setText("Прийняти заяву");

acceptRequestButton.addActionListener(new AcceptRequestButtonController(requestsTable, this));

workPlanTable.setModel(new WorkPlanTableModel());

workPlanTable.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

((DefaultTableCellRenderer)workPlanTable.getTableHeader().getDefaultRenderer())

.setHorizontalAlignment(JLabel.LEFT);

workPlanTableScrollPane.setBorder(BorderFactory.createTitledBorder("План робіт"));

workPlanTableScrollPane.setPreferredSize(new Dimension(Const.DISPATCHER\_FRAME\_WIDTH,

Const.DISPATCHER\_FRAME\_HEIGHT));

workPlanTableScrollPane.setViewportView(workPlanTable);

deleteWorkPlanRecordButton.setText("Видалити з плану робіт");

deleteWorkPlanRecordButton.addActionListener(new DeleteWPRecordButtonController(workPlanTable, this));

GroupLayout layout = new GroupLayout(getContentPane());

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addContainerGap()

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)

.addComponent(headerLabel)

.addComponent(requestsTableScrollPane)

.addComponent(acceptRequestButton)

.addComponent(workPlanTableScrollPane)

.addComponent(deleteWorkPlanRecordButton))

.addContainerGap()));

layout.setVerticalGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addContainerGap()

.addComponent(headerLabel)

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

.addComponent(requestsTableScrollPane)

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

.addComponent(acceptRequestButton)

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

.addComponent(workPlanTableScrollPane)

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

.addComponent(deleteWorkPlanRecordButton)

.addContainerGap()));

pack();

setLocationRelativeTo(null);

// Запланувати оновлення таблиць

Timer requestsTableTimer = new Timer(0, (ActionEvent e) -> {

List<Request> requests = StorageDAO.getInstance().readRequests()

.stream()

.filter(x -> !x.isAccepted())

.collect(Collectors.toList());

if (!requests.equals(lastRequests)) {

lastRequests = requests;

fillRequestTable(requests);

}

});

requestsTableTimer.setDelay(Const.UPDATE\_DELAY);

requestsTableTimer.start();

Timer workPlanTableTimer = new Timer(0, (ActionEvent e) -> {

List<WorkPlanRecord> workPlan = StorageDAO.getInstance().readWorkPlan();

if (!workPlan.equals(lastWorkPlan)) {

lastWorkPlan = workPlan;

fillWorkPlanTable(workPlan);

}

});

workPlanTableTimer.setDelay(Const.UPDATE\_DELAY);

workPlanTableTimer.start();

}

/\*\*

\* Заповнює таблицю запитів.

\* @param requests нові реквести що будуть відображені

\*/

private void fillRequestTable(List<Request> requests) {

DefaultTableModel requestsTableModel = (DefaultTableModel) requestsTable.getModel();

// Спочатку затерти старі дані

for (int i = requestsTableModel.getRowCount() - 1; i >= 0; i--)

requestsTableModel.removeRow(i);

// Один рядок даних

Object[] rowData = new Object[5];

// Заповнення таблиці

for (Request request : requests) {

rowData[0] = request.getId();

rowData[1] = StorageDAO.getInstance() // Знайти ім'я користувача по Id запиту

.readUsers()

.stream()

.filter(x -> x.getId() == request.getId\_tenant())

.findFirst()

.map(User::getName)

.orElse("Немає такого користувача")

;

rowData[2] = request.getWorkType();

rowData[3] = request.getWorkScale();

rowData[4] = request.getWorkPrefferedTime();

requestsTableModel.addRow(rowData);

}

}

/\*\*

\* Заповнює таблицю плану робіт.

\* @param workPlan новий план що буде відображено

\*/

private void fillWorkPlanTable(List<WorkPlanRecord> workPlan) {

DefaultTableModel workPlanTableModel = (DefaultTableModel) workPlanTable.getModel();

// Спочатку затерти старі дані

for (int i = workPlanTableModel.getRowCount() - 1; i >= 0; i--)

workPlanTableModel.removeRow(i);

// Один рядок даних

Object[] rowData = new Object[6];

// Заповнення таблиці

for (WorkPlanRecord workPlanRecord : workPlan) {

rowData[0] = workPlanRecord.getId();

rowData[1] = workPlanRecord.getIdRequest();

rowData[2] = StorageDAO.getInstance() // Знайти реквест по запису в плані роботи

.readRequests()

.stream()

.filter(x -> x.getId() == workPlanRecord.getIdRequest())

.flatMap(req -> StorageDAO.getInstance() // Знайти користувача по реквесту

.readUsers()

.stream()

.filter(usr -> usr.getId() == req.getId\_tenant()))

.findFirst()

.map(User::getName)

.orElse("Немає такого користувача");

rowData[3] = workPlanRecord.getIdBrigade();

rowData[4] = StorageDAO.getInstance()

.readBrigades()

.stream()

.filter(brigade -> brigade.getId() == workPlanRecord.getIdBrigade())

.findFirst()

.map(Brigade::getBrigadierName)

.orElse("Немає такого бригадира");

rowData[5] = StorageDAO.getInstance()

.readBrigades()

.stream()

.filter(brigade -> brigade.getId() == workPlanRecord.getIdBrigade())

.findFirst()

.map(Brigade::getWorkersNumber)

.orElse(0);

workPlanTableModel.addRow(rowData);

}

}

}

package ua.hcs.view;

import ua.hcs.controller.SendRequestButtonController;

import ua.hcs.model.User;

import javax.swing.\*;

import java.awt.\*;

/\*\*

\* Фрейм орендара.

\*/

public class Tenant extends JFrame {

private JLabel headerLabel;

private JLabel kindOfWorkLabel;

private JTextField kindOfWorkField;

private JLabel scaleOfWorkLabel;

private JTextField scaleOfWorkField;

private JLabel prefTimeOfWorkLabel;

private JTextField prefTimeOfWorkField;

private JButton sendRequestButton;

private User tenant;

public Tenant(User tenant) {

headerLabel = new JLabel();

kindOfWorkLabel = new JLabel();

kindOfWorkField = new JTextField();

scaleOfWorkLabel = new JLabel();

scaleOfWorkField = new JTextField();

prefTimeOfWorkLabel = new JLabel();

prefTimeOfWorkField = new JTextField();

sendRequestButton = new JButton();

this.tenant = tenant;

setDefaultCloseOperation(WindowConstants.EXIT\_ON\_CLOSE);

setTitle("Додаток ЖЕК ~ Заява");

headerLabel.setFont(new Font("Tahoma", Font.BOLD, 15));

headerLabel.setHorizontalAlignment(SwingConstants.LEFT);

headerLabel.setText("Сформувати заяву");

kindOfWorkLabel.setText("Введіть рід робіт:");

scaleOfWorkLabel.setText("Введіть масштаб робіт:");

prefTimeOfWorkLabel.setText("Введіть бажаний час в годинах:");

sendRequestButton.setText("Відправити заяву");

sendRequestButton.addActionListener(new SendRequestButtonController(kindOfWorkField,

scaleOfWorkField, prefTimeOfWorkField, tenant, this));

GroupLayout layout = new GroupLayout(getContentPane());

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addContainerGap()

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)

.addComponent(headerLabel)

.addComponent(kindOfWorkLabel)

.addComponent(kindOfWorkField)

.addComponent(scaleOfWorkLabel)

.addComponent(scaleOfWorkField)

.addComponent(prefTimeOfWorkLabel)

.addComponent(prefTimeOfWorkField)

.addComponent(sendRequestButton))

.addContainerGap()));

layout.setVerticalGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addContainerGap()

.addComponent(headerLabel)

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

.addComponent(kindOfWorkLabel)

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

.addComponent(kindOfWorkField)

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

.addComponent(scaleOfWorkLabel)

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

.addComponent(scaleOfWorkField)

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

.addComponent(prefTimeOfWorkLabel)

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

.addComponent(prefTimeOfWorkField)

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

.addComponent(sendRequestButton)

.addContainerGap()));

pack();

setLocationRelativeTo(null);

}

}

package ua.hcs.model;

import org.jetbrains.annotations.NotNull;

import java.io.Serializable;

import java.util.Objects;

/\*\*

\* Зображує модель робочої Бригади.

\*/

public class Brigade implements Serializable, Cloneable, Comparable<Brigade> {

private long id;

private String brigadierName;

private int workersNumber;

public Brigade(long id, String brigadierName, int workersNumber) {

this.id = id;

this.brigadierName = brigadierName;

this.workersNumber = workersNumber;

}

public Brigade(String[] fields) {

this.id = Long.valueOf(fields[0]);

this.brigadierName = fields[1];

this.workersNumber = Integer.valueOf(fields[2]);

}

public long getId() {

return id;

}

public String getBrigadierName() {

return brigadierName;

}

public int getWorkersNumber() {

return workersNumber;

}

@Override

public Brigade clone() throws CloneNotSupportedException {

return (Brigade) super.clone();

}

@Override

public int compareTo(@NotNull Brigade o) {

return Long.valueOf(this.id - o.id).intValue();

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

Brigade brigade = (Brigade) o;

return id == brigade.id &&

workersNumber == brigade.workersNumber &&

Objects.equals(brigadierName, brigade.brigadierName);

}

@Override

public int hashCode() {

return Objects.hash(id, brigadierName, workersNumber);

}

@Override

public String toString() {

return "Brigade{" +

"id=" + id +

", brigadierName='" + brigadierName + '\'' +

", workersNumber=" + workersNumber +

'}';

}

}

package ua.hcs.model;

import org.jetbrains.annotations.NotNull;

import java.io.Serializable;

import java.util.Objects;

/\*\*

\* Зображує модель Запиту користувача.

\*/

public class Request implements Serializable, Cloneable, Comparable<Request> {

private long id;

private long id\_tenant;

private String workType;

private String workScale;

private double workPrefferedTime;

private boolean accepted;

public Request(long id, long id\_tenant, String workType, String workScale,

double workPrefferedTime, boolean accepted) {

this.id = id;

this.id\_tenant = id\_tenant;

this.workType = workType;

this.workScale = workScale;

this.workPrefferedTime = workPrefferedTime;

this.accepted = accepted;

}

public Request(String[] fields) {

this.id = Long.valueOf(fields[0]);

this.id\_tenant = Long.valueOf(fields[1]);

this.workType = fields[2];

this.workScale = fields[3];

this.workPrefferedTime = Double.valueOf(fields[4]);

this.accepted = Boolean.valueOf(fields[5]);

}

public long getId() {

return id;

}

public long getId\_tenant() {

return id\_tenant;

}

public String getWorkType() {

return workType;

}

public String getWorkScale() {

return workScale;

}

public double getWorkPrefferedTime() {

return workPrefferedTime;

}

public boolean isAccepted() {

return accepted;

}

@Override

public Request clone() throws CloneNotSupportedException {

return (Request) super.clone();

}

@Override

public int compareTo(@NotNull Request o) {

return Long.valueOf(this.id - o.id).intValue();

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

Request request = (Request) o;

return id == request.id &&

id\_tenant == request.id\_tenant &&

Double.compare(request.workPrefferedTime, workPrefferedTime) == 0 &&

accepted == request.accepted &&

Objects.equals(workType, request.workType) &&

Objects.equals(workScale, request.workScale);

}

@Override

public int hashCode() {

return Objects.hash(id, id\_tenant, workType, workScale, workPrefferedTime, accepted);

}

@Override

public String toString() {

return "Request{" +

"id=" + id +

", id\_tenant=" + id\_tenant +

", workType='" + workType + '\'' +

", workScale='" + workScale + '\'' +

", workPrefferedTime=" + workPrefferedTime +

", accepted=" + accepted +

'}';

}

}

package ua.hcs.model;

import org.jetbrains.annotations.NotNull;

import ua.hcs.util.Const;

import javax.swing.table.DefaultTableModel;

import java.io.Serializable;

import java.util.Arrays;

/\*\*

\* Модель таблиці запитів орендаторів.

\*/

public class RequestsTableModel extends DefaultTableModel implements Serializable,

Cloneable, Comparable<RequestsTableModel> {

private Object[] header;

private Object[][] content;

private Class[] types = new Class[]{

Long.class, String.class, String.class, String.class, String.class

};

private boolean[] canEdit = new boolean[]{

false, false, false, false, false

};

public RequestsTableModel() {

header = Const.REQUESTS\_TABLE\_HEADER;

content = new Object[][]{{1.0, "2", "3", "4", "5"}, {1.0, "2", "3", "4", "5"}};

setDataVector(content, header);

}

public Object[] getHeader() {

return header;

}

public Object[][] getContent() {

return content;

}

public Class[] getTypes() {

return types;

}

@Override

public Class getColumnClass(int columnIndex) {

return types[columnIndex];

}

@Override

public boolean isCellEditable(int rowIndex, int columnIndex) {

return canEdit[columnIndex];

}

@Override

public RequestsTableModel clone() throws CloneNotSupportedException {

return (RequestsTableModel) super.clone();

}

@Override

public int compareTo(@NotNull RequestsTableModel o) {

return Long.valueOf(this.header.length - o.header.length).intValue();

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

RequestsTableModel that = (RequestsTableModel) o;

return Arrays.equals(header, that.header) &&

Arrays.equals(content, that.content) &&

Arrays.equals(types, that.types) &&

Arrays.equals(canEdit, that.canEdit);

}

@Override

public int hashCode() {

int result = Arrays.hashCode(header);

result = 31 \* result + Arrays.hashCode(content);

result = 31 \* result + Arrays.hashCode(types);

result = 31 \* result + Arrays.hashCode(canEdit);

return result;

}

@Override

public String toString() {

return "RequestsTableModel{" +

"header=" + (header == null ? null : Arrays.asList(header)) +

", content=" + (content == null ? null : Arrays.asList(content)) +

'}';

}

}

package ua.hcs.model;

import ua.hcs.util.Const;

import java.io.\*;

import java.nio.file.Files;

import java.nio.file.Paths;

import java.util.ArrayList;

import java.util.List;

import java.util.concurrent.locks.ReadWriteLock;

import java.util.concurrent.locks.ReentrantReadWriteLock;

/\*\*

\* Забезпечує синхронізований доступ до сховища даних.

\* Використано паттерн Singleton.

\* Багато процесів можуть одночасно читати зі сховища,

\* проте коли хтось починає писати в сховище всі інші чекають.

\*/

public class StorageDAO {

private static final StorageDAO \_instance = new StorageDAO();

private ReadWriteLock brigadesLock;

private ReadWriteLock requestsLock;

private ReadWriteLock usersLock;

private ReadWriteLock workPlanLock;

private StorageDAO() {

brigadesLock = new ReentrantReadWriteLock();

requestsLock = new ReentrantReadWriteLock();

usersLock = new ReentrantReadWriteLock();

workPlanLock = new ReentrantReadWriteLock();

}

/\*\*

\* Читає TSV файл

\* @param filePath шлях до файлу

\* @param columns кількість стовбців

\* @param lock лок на окремий файл

\* @return Лист рядків, що складаються з масиву стовбців

\*/

private List<String[]> readFileSplittedByTab(String filePath,

int columns, ReadWriteLock lock) {

lock.readLock().lock();

File file = new File(filePath);

// if file is absent or empty return empty list

if (!file.exists() || file.length() == 0) {

lock.readLock().unlock();

return new ArrayList<String[]>();

}

List<String> lines = null;

try {

lines = Files.readAllLines(Paths.get(file.getAbsolutePath()));

} catch (IOException e) {

e.printStackTrace();

} finally {

lock.readLock().unlock();

}

List<String[]> splittedFile = new ArrayList<>();

String[] parts = null;

for (String line : lines) {

parts = line.split("\t");

if (parts.length != columns)

continue;

splittedFile.add(parts);

}

return splittedFile;

}

/\*\*

\* Повертає об'єкт сховища

\* @return об'єкт сховища

\*/

public static synchronized StorageDAO getInstance() {

return \_instance;

}

/\*\*

\* Повертає актуальний список бригад.

\* @return актуальний лист бригад

\*/

public List<Brigade> readBrigades() {

List<String[]> splittedFile = readFileSplittedByTab(Const.BRIGADES\_FILEPATH,

3, brigadesLock);

List<Brigade> brigades = new ArrayList<>();

for (String[] line : splittedFile)

brigades.add(new Brigade(line));

return brigades;

}

/\*\*

\* Записуэ бригади до сховища.

\* @param brigades бригади, що будуть записані

\* @param appendData чи дописувати дані

\*/

public void writeBrigades(List<Brigade> brigades, boolean appendData) {

PrintWriter printWriter = null;

try {

brigadesLock.writeLock().lock();

printWriter = new PrintWriter(

new BufferedWriter(

new FileWriter(Const.BRIGADES\_FILEPATH, appendData)));

for (Brigade b : brigades) {

printWriter.println(b.getId() + "\t" +

b.getBrigadierName() + "\t" +

b.getWorkersNumber());

}

} catch (IOException e) {

e.printStackTrace();

} finally {

printWriter.flush();

printWriter.close();

brigadesLock.writeLock().unlock();

}

}

/\*\*

\* Повертає актуальний список запитів користувачів.

\* @return актуальний список запитів користувачів

\*/

public List<Request> readRequests() {

List<String[]> splittedFile = readFileSplittedByTab(Const.REQUESTS\_FILEPATH,

6, requestsLock);

List<Request> request = new ArrayList<>();

for (String[] line : splittedFile)

request.add(new Request(line));

return request;

}

/\*\*

\* Записуэ запити користувачів до сховища.

\* @param requests запити, що будуть записані

\* @param appendData чи дописувати дані

\*/

public void writeRequests(List<Request> requests, boolean appendData) {

PrintWriter printWriter = null;

try {

requestsLock.writeLock().lock();

printWriter = new PrintWriter(

new BufferedWriter(

new FileWriter(Const.REQUESTS\_FILEPATH, appendData)));

for (Request r : requests) {

printWriter.println(r.getId() + "\t" +

r.getId\_tenant() + "\t" +

r.getWorkType() + "\t" +

r.getWorkScale() + "\t" +

r.getWorkPrefferedTime() + "\t" +

r.isAccepted());

}

} catch (IOException e) {

e.printStackTrace();

} finally {

printWriter.flush();

printWriter.close();

requestsLock.writeLock().unlock();

}

}

/\*\*

\* Повертає актуальний список користувачів.

\* @return актуальний лист користувачів

\*/

public List<User> readUsers() {

List<String[]> splittedFile = readFileSplittedByTab(Const.USERS\_FILEPATH,

4, usersLock);

List<User> users = new ArrayList<>();

for (String[] line : splittedFile)

users.add(new User(line));

return users;

}

/\*\*

\* Записуэ користувачів до сховища.

\* @param users користувачі, що будуть записані

\* @param appendData чи дописувати дані

\*/

public void writeUsers(List<User> users, boolean appendData) {

PrintWriter printWriter = null;

try {

usersLock.writeLock().lock();

printWriter = new PrintWriter(

new BufferedWriter(

new FileWriter(Const.USERS\_FILEPATH, appendData)));

for (User u : users) {

printWriter.println(u.getId() + "\t" +

u.getName() + "\t" +

u.getPassword() + "\t" +

u.getRole());

}

} catch (IOException e) {

e.printStackTrace();

} finally {

printWriter.flush();

printWriter.close();

usersLock.writeLock().unlock();

}

}

/\*\*

\* Повертає актуальний план робіт.

\* @return актуальний план робіт

\*/

public List<WorkPlanRecord> readWorkPlan() {

List<String[]> splittedFile = readFileSplittedByTab(Const.WORKPLAN\_FILEPATH,

3, workPlanLock);

List<WorkPlanRecord> workPlan = new ArrayList<>();

for (String[] line : splittedFile)

workPlan.add(new WorkPlanRecord(line));

return workPlan;

}

/\*\*

\* Записуэ план робіт до сховища.

\* @param workPlan план робіт, що буде записано

\* @param appendData чи дописувати дані

\*/

public void writeWorkPlan(List<WorkPlanRecord> workPlan, boolean appendData) {

PrintWriter printWriter = null;

try {

workPlanLock.writeLock().lock();

printWriter = new PrintWriter(

new BufferedWriter(

new FileWriter(Const.WORKPLAN\_FILEPATH, appendData)));

for (WorkPlanRecord wpr : workPlan) {

printWriter.println(wpr.getId() + "\t" +

wpr.getIdRequest() + "\t" +

wpr.getIdBrigade());

}

} catch (IOException e) {

e.printStackTrace();

} finally {

printWriter.flush();

printWriter.close();

workPlanLock.writeLock().unlock();

}

}

}

package ua.hcs.model;

import org.jetbrains.annotations.NotNull;

import java.io.Serializable;

import java.util.Objects;

/\*\*

\* Зображуэ модель Користувача системи. Це може бути квартиронаймач або диспетчер.

\*/

public class User implements Serializable, Cloneable, Comparable<User> {

private long id;

private String name;

private String password;

private String role;

public User(long id, String name, String password, String role) {

this.id = id;

this.name = name;

this.password = password;

this.role = role;

}

public User(String[] fields) {

this.id = Long.valueOf(fields[0]);

this.name = fields[1];

this.password = fields[2];

this.role = fields[3];

}

public long getId() {

return id;

}

public String getName() {

return name;

}

public String getPassword() {

return password;

}

public String getRole() {

return role;

}

@Override

public User clone() throws CloneNotSupportedException {

return (User) super.clone();

}

@Override

public int compareTo(@NotNull User o) {

return Long.valueOf(this.id - o.id).intValue();

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

User user = (User) o;

return id == user.id &&

Objects.equals(name, user.name) &&

Objects.equals(password, user.password) &&

Objects.equals(role, user.role);

}

@Override

public int hashCode() {

return Objects.hash(id, name, password, role);

}

@Override

public String toString() {

return "User{" +

"id=" + id +

", name='" + name + '\'' +

", role='" + role + '\'' +

'}';

}

}

package ua.hcs.model;

import org.jetbrains.annotations.NotNull;

import java.io.Serializable;

import java.util.Objects;

/\*\*

\* Зображує один запис Робочого плану.

\*/

public class WorkPlanRecord implements Serializable, Cloneable, Comparable<WorkPlanRecord> {

private long id;

private long idRequest;

private long idBrigade;

public WorkPlanRecord(long id, long idRequest, long idBrigade) {

this.id = id;

this.idRequest = idRequest;

this.idBrigade = idBrigade;

}

public WorkPlanRecord(String[] fields) {

this.id = Long.valueOf(fields[0]);

this.idRequest = Long.valueOf(fields[1]);

this.idBrigade = Long.valueOf(fields[2]);

}

public long getId() {

return id;

}

public long getIdRequest() {

return idRequest;

}

public long getIdBrigade() {

return idBrigade;

}

@Override

public WorkPlanRecord clone() throws CloneNotSupportedException {

return (WorkPlanRecord) super.clone();

}

@Override

public int compareTo(@NotNull WorkPlanRecord o) {

return Long.valueOf(this.id - o.id).intValue();

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

WorkPlanRecord that = (WorkPlanRecord) o;

return id == that.id &&

idRequest == that.idRequest &&

idBrigade == that.idBrigade;

}

@Override

public int hashCode() {

return Objects.hash(id, idRequest, idBrigade);

}

@Override

public String toString() {

return "WorkPlanRecord{" +

"id=" + id +

", idRequest=" + idRequest +

", idBrigade=" + idBrigade +

'}';

}

}

package ua.hcs.model;

import org.jetbrains.annotations.NotNull;

import ua.hcs.util.Const;

import javax.swing.table.DefaultTableModel;

import java.io.Serializable;

import java.util.Arrays;

/\*\*

\* Модель таблиці плану робіт.

\*/

public class WorkPlanTableModel extends DefaultTableModel implements Serializable,

Cloneable, Comparable<WorkPlanTableModel> {

private Object[] header;

private Object[][] content;

private Class[] types = new Class[]{

Long.class, Long.class, String.class, Long.class, String.class, Integer.class

};

private boolean[] canEdit = new boolean[]{

false, false, false, false, false, false

};

public WorkPlanTableModel() {

header = Const.WORK\_PLAN\_TABLE\_HEADER;

content = new Object[][]{{1.0, 2.1, "3", 4.0, "5", 6}, {1.0, 2.1, "3", 4.0, "5", 67}};

setDataVector(content, header);

}

public Object[] getHeader() {

return header;

}

public Object[][] getContent() {

return content;

}

public Class[] getTypes() {

return types;

}

@Override

public Class getColumnClass(int columnIndex) {

return types[columnIndex];

}

@Override

public boolean isCellEditable(int rowIndex, int columnIndex) {

return canEdit[columnIndex];

}

@Override

public WorkPlanTableModel clone() throws CloneNotSupportedException {

return (WorkPlanTableModel) super.clone();

}

@Override

public int compareTo(@NotNull WorkPlanTableModel o) {

return Long.valueOf(this.header.length - o.header.length).intValue();

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

WorkPlanTableModel that = (WorkPlanTableModel) o;

return Arrays.equals(header, that.header) &&

Arrays.equals(content, that.content) &&

Arrays.equals(types, that.types) &&

Arrays.equals(canEdit, that.canEdit);

}

@Override

public int hashCode() {

int result = Arrays.hashCode(header);

result = 31 \* result + Arrays.hashCode(content);

result = 31 \* result + Arrays.hashCode(types);

result = 31 \* result + Arrays.hashCode(canEdit);

return result;

}

@Override

public String toString() {

return "WorkPlanTableModel{" +

"header=" + (header == null ? null : Arrays.asList(header)) +

", content=" + (content == null ? null : Arrays.asList(content)) +

'}';

}

}

package ua.hcs.controller;

import ua.hcs.model.Request;

import ua.hcs.model.StorageDAO;

import ua.hcs.model.User;

import ua.hcs.view.BrigadeCreation;

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

/\*\*

\* Обробляє натиснення на кнопку прийняття запиту.

\*/

public class AcceptRequestButtonController implements ActionListener {

private JTable requestsTable;

private JFrame frame;

public AcceptRequestButtonController(JTable requestsTable, JFrame frame) {

this.requestsTable = requestsTable;

this.frame = frame;

}

@Override

public void actionPerformed(ActionEvent e) {

int selectedRowIndex = requestsTable.getSelectedRow();

if (selectedRowIndex < 0) {

JOptionPane.showMessageDialog(frame, "Будь ласка, виберіть з таблиці заяву!");

return;

}

// Отримати дані з таблиці

DefaultTableModel requestsTableModel = (DefaultTableModel) requestsTable.getModel();

Object[] record = new Object[5];

for (int i = 0; i < 5; i++)

record[i] = requestsTableModel.getValueAt(selectedRowIndex, i);

// Знайти користувача за ім'ям

String username = String.valueOf(record[1]);

User tenant = StorageDAO.getInstance().readUsers()

.stream()

.filter(x -> x.getName().equals(username))

.findFirst()

.orElse(null);

if (tenant == null) {

JOptionPane.showMessageDialog(frame, "Не знайдено користувача з ім'ям \"" + username + "\"." +

" Будь ласка, зверніться до адміністратора!");

return;

}

// Позначити запит, як прийнятий

Request request = new Request(

Long.valueOf(String.valueOf(record[0])),

tenant.getId(),

String.valueOf(record[2]),

String.valueOf(record[3]),

Double.valueOf(String.valueOf(record[4])),

true);

// Показати фрейм формування бригади під запит

BrigadeCreation brigadeCreation = new BrigadeCreation(request, frame);

brigadeCreation.setVisible(true);

}

}

package ua.hcs.controller;

import ua.hcs.model.Brigade;

import ua.hcs.model.Request;

import ua.hcs.model.StorageDAO;

import ua.hcs.model.WorkPlanRecord;

import ua.hcs.util.Util;

import javax.swing.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.Arrays;

import java.util.List;

import java.util.stream.Collectors;

/\*\*

\* Обробляє натиснення на кнопку формування бригади на запит.

\*/

public class ConfirmBCreationButtonController implements ActionListener {

private Request request;

private JTextField brigadierName;

private JTextField workersNumber;

private JFrame frame;

public ConfirmBCreationButtonController(Request request, JTextField brigadierName,

JTextField workersNumber, JFrame frame) {

this.request = request;

this.brigadierName = brigadierName;

this.workersNumber = workersNumber;

this.frame = frame;

}

@Override

public void actionPerformed(ActionEvent e) {

// Сформувати бригаду і занесена до робочого плану

String enteredBrigadierName = brigadierName.getText();

String enteredWorkersNumberStr = workersNumber.getText();

int enteredWorkersNumber = 0;

if (enteredBrigadierName.isEmpty() || enteredWorkersNumberStr.isEmpty()) {

JOptionPane.showMessageDialog(frame, "Будь ласка, введіть ім'я бригадира та кількість працівників!");

return;

} else {

try {

enteredWorkersNumber = Integer.valueOf(enteredWorkersNumberStr);

} catch (NumberFormatException nfe) {

JOptionPane.showMessageDialog(frame,

"Будь ласка, введіть кількість працівників цілими цифрами! Приклад: 6, 5, 12");

return;

}

}

// Сперше, позначимо запит як прийнятий і запишемо зміни

List<Request> filteredRequests = StorageDAO.getInstance()

.readRequests()

.stream()

.filter(x -> x.getId() != request.getId())

.collect(Collectors.toList());

filteredRequests.add(request); // додати змінений

// Записати зміни

StorageDAO.getInstance().writeRequests(filteredRequests, false);

Brigade brigade = new Brigade(Util.generateId(), enteredBrigadierName, enteredWorkersNumber);

StorageDAO.getInstance().writeBrigades(Arrays.asList(brigade), true);

WorkPlanRecord workPlanRecord = new WorkPlanRecord(Util.generateId(), request.getId(), brigade.getId());

StorageDAO.getInstance().writeWorkPlan(Arrays.asList(workPlanRecord), true);

JOptionPane.showMessageDialog(frame, "Бригада сформована та занесена до робочого плану!");

frame.dispose();

}

}

package ua.hcs.controller;

import ua.hcs.model.StorageDAO;

import ua.hcs.model.WorkPlanRecord;

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.List;

import java.util.stream.Collectors;

/\*\*

\* Обробляє натиснення на кнопку видалення запису з плану робіт.

\*/

public class DeleteWPRecordButtonController implements ActionListener {

private JTable workPlanTable;

private JFrame frame;

public DeleteWPRecordButtonController(JTable workPlanTable, JFrame frame) {

this.workPlanTable = workPlanTable;

this.frame = frame;

}

@Override

public void actionPerformed(ActionEvent e) {

int selectedRowIndex = workPlanTable.getSelectedRow();

if (selectedRowIndex < 0) {

JOptionPane.showMessageDialog(frame, "Будь ласка, виберіть один із записів робочого плану з таблиці!");

return;

}

DefaultTableModel requestsTableModel = (DefaultTableModel) workPlanTable.getModel();

Object[] record = new Object[6];

for (int i = 0; i < 5; i++)

record[i] = requestsTableModel.getValueAt(selectedRowIndex, i);

long workPlanId = Long.valueOf(record[0].toString());

List<WorkPlanRecord> filteredWorkPlan = StorageDAO.getInstance()

.readWorkPlan()

.stream()

.filter(x -> x.getId() != workPlanId)

.collect(Collectors.toList());

StorageDAO.getInstance().writeWorkPlan(filteredWorkPlan, false);

JOptionPane.showMessageDialog(frame, "Запис " + workPlanId +

" успішно видалено з робочого плану!");

}

}

package ua.hcs.controller;

import ua.hcs.model.StorageDAO;

import ua.hcs.model.User;

import ua.hcs.util.Const;

import ua.hcs.view.Dispatcher;

import ua.hcs.view.Tenant;

import javax.swing.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.WindowEvent;

import static java.awt.event.WindowEvent.WINDOW\_CLOSING;

/\*\*

\* Обробляє натиснення на кнопку ввійти.

\*/

public class LoginButtonController implements ActionListener {

private JTextField nameField;

private JPasswordField passwordField;

private JFrame frame;

public LoginButtonController(JTextField nameField, JPasswordField passwordField, JFrame frame) {

this.nameField = nameField;

this.passwordField = passwordField;

this.frame = frame;

}

@Override

public void actionPerformed(ActionEvent e) {

String enteredName = nameField.getText();

String enteredPass = String.valueOf(passwordField.getPassword());

if (enteredName.isEmpty() || enteredPass.isEmpty()) {

JOptionPane.showMessageDialog(frame, "Будь ласка, введіть ваше ім'я та пароль!");

return;

}

User foundUser = StorageDAO.getInstance().readUsers()

.stream()

.filter(x -> x.getName().equals(enteredName))

.findFirst()

.orElse(null);

if (foundUser == null)

JOptionPane.showMessageDialog(frame, "Користувача \"" + enteredName + "\" не знайдено!");

else if (!foundUser.getPassword().equals(enteredPass))

JOptionPane.showMessageDialog(frame, "Пароль введено не вірно!");

else if (foundUser.getRole().equals(Const.ROLE\_TENANT)) {

Tenant tenantGUI = new Tenant(foundUser);

frame.setVisible(false);

tenantGUI.setVisible(true);

frame.dispose();

} else if (foundUser.getRole().equals(Const.ROLE\_DISPATCHER)) {

Dispatcher dispatcher = new Dispatcher();

frame.setVisible(false);

dispatcher.setVisible(true);

frame.dispose();

} else {

JOptionPane.showMessageDialog(frame,

"В вашого користувача помилкова роль \"" + foundUser.getRole() +

"\"! Будь ласка, зверніться до адміністратора!");

frame.dispatchEvent(new WindowEvent(frame, WINDOW\_CLOSING));

}

}

}

package ua.hcs.controller;

import ua.hcs.model.Request;

import ua.hcs.model.StorageDAO;

import ua.hcs.model.User;

import ua.hcs.util.Util;

import javax.swing.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.Arrays;

/\*\*

\* Обробляє натиснення на кнопку відправлення запиту.

\*/

public class SendRequestButtonController implements ActionListener {

private JTextField kindOfWorkField;

private JTextField scaleOfWorkField;

private JTextField prefTimeOfWorkField;

private User tenant;

private JFrame frame;

public SendRequestButtonController(JTextField kindOfWorkField, JTextField scaleOfWorkField,

JTextField prefTimeOfWorkField, User tenant, JFrame frame) {

this.kindOfWorkField = kindOfWorkField;

this.scaleOfWorkField = scaleOfWorkField;

this.prefTimeOfWorkField = prefTimeOfWorkField;

this.tenant = tenant;

this.frame = frame;

}

@Override

public void actionPerformed(ActionEvent e) {

String kindOfWork = kindOfWorkField.getText();

String scaleOfWork = scaleOfWorkField.getText();

String prefTimeOfWorkStr = prefTimeOfWorkField.getText();

Double prefTimeOfWork = 0.0;

if (kindOfWork.isEmpty() || scaleOfWork.isEmpty() || prefTimeOfWorkStr.isEmpty()) {

JOptionPane.showMessageDialog(frame, "Будь ласка, заповніть всі поля!");

return;

}

else {

try {

prefTimeOfWork = Double.valueOf(prefTimeOfWorkStr);

} catch (NumberFormatException nfe) {

JOptionPane.showMessageDialog(frame, "Будь ласка, введіть години цифрами! Приклад: 6, 1.5, 3.0");

return;

}

}

Request request = new Request(Util.generateId(), tenant.getId(), kindOfWork, scaleOfWork, prefTimeOfWork, false);

StorageDAO.getInstance()

.writeRequests(Arrays.asList(request), true);

JOptionPane.showMessageDialog(frame, "Ваш запит успішно відправлено!");

}

}

package ua.hcs;

import ua.hcs.view.Authorization;

/\*\*

\* Запускає додаток ЖЕК.

\*/

public class HousingAndCommunalServicesApp extends Authorization {

public static void main(String[] args) {

Authorization authorization = new Authorization();

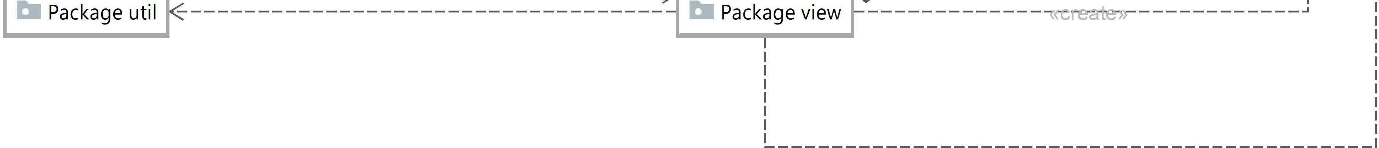
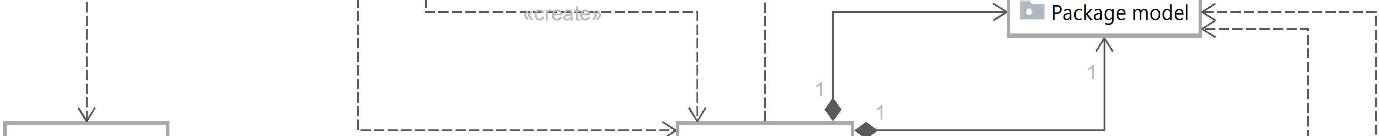
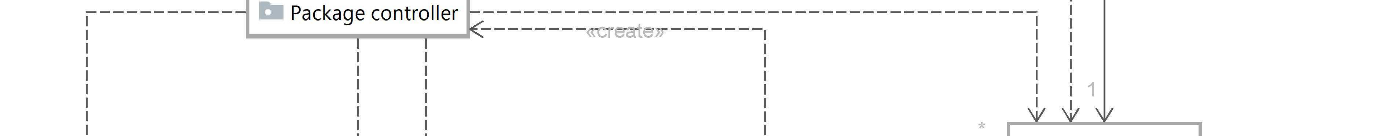
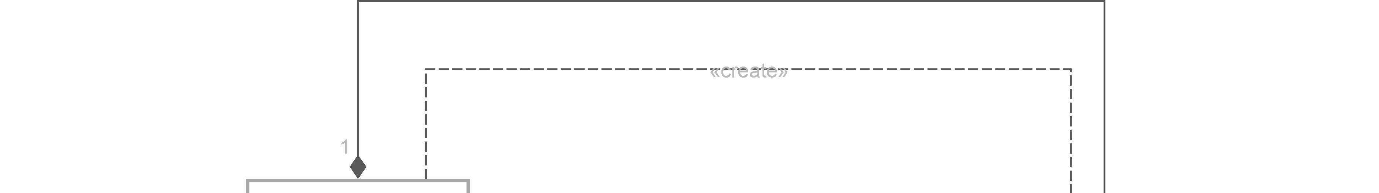
authorization.setVisible(true);

}

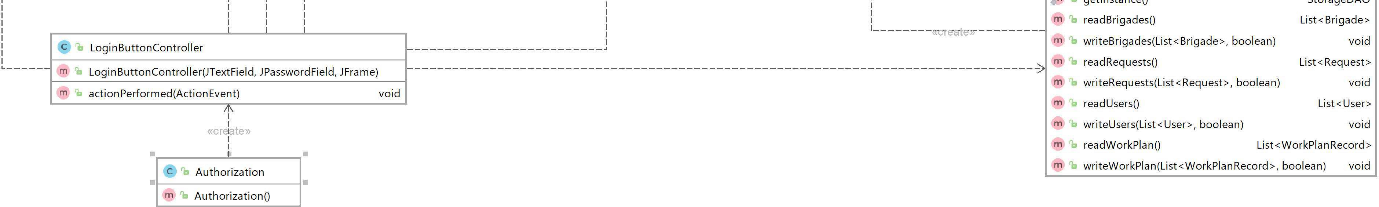
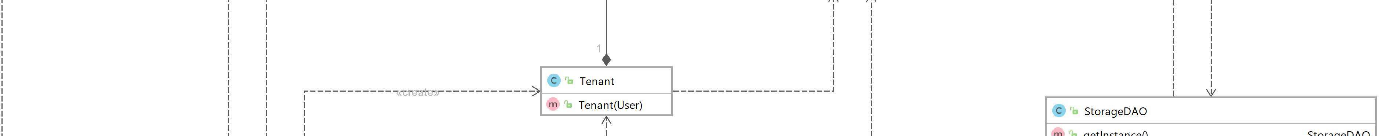
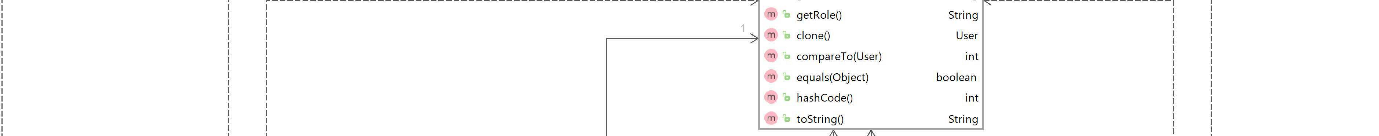
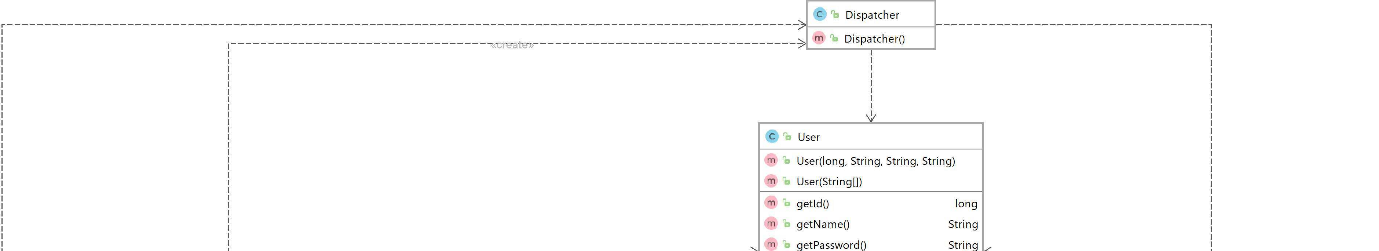
}

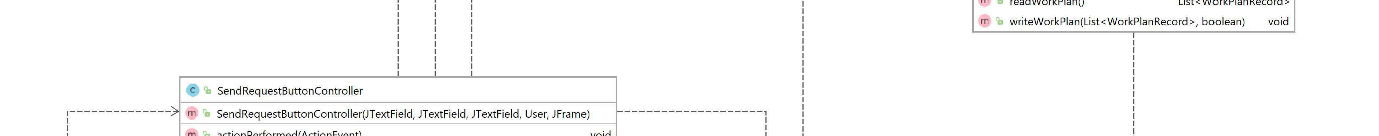
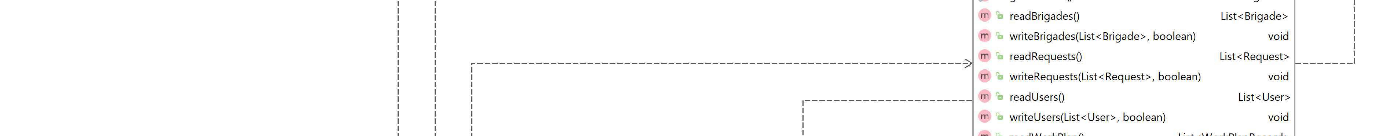
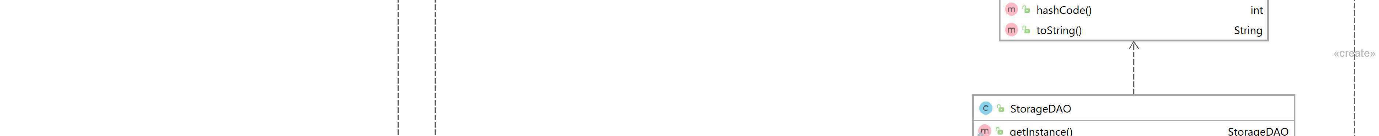
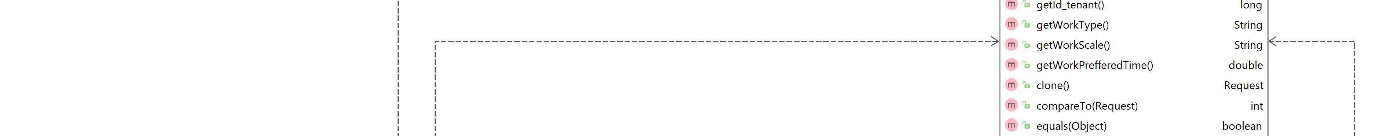
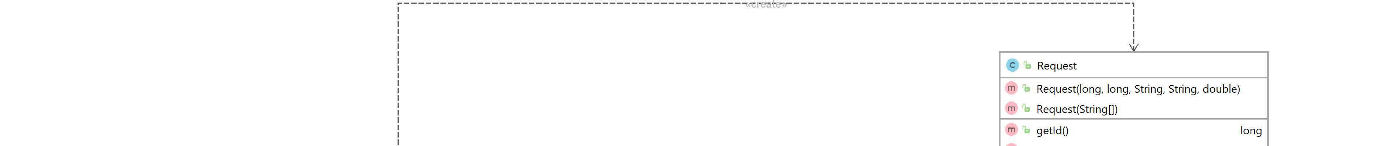
UML

Огляд пакетів додатку

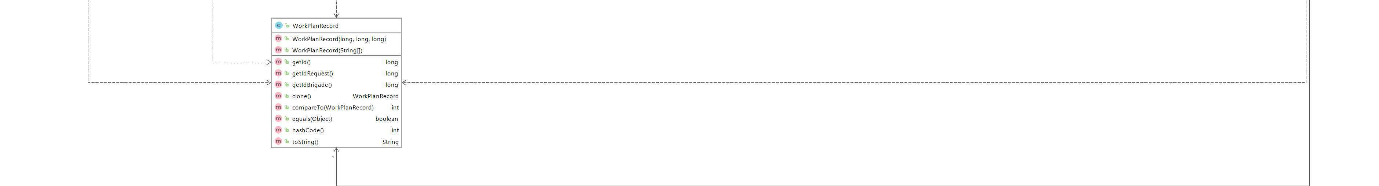
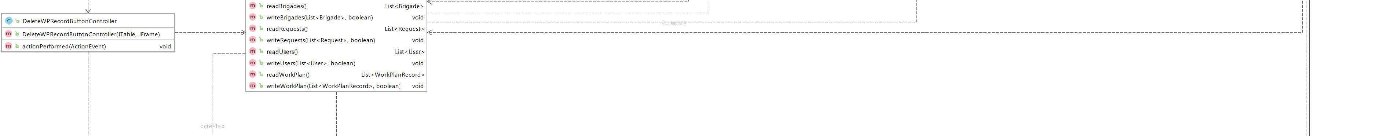
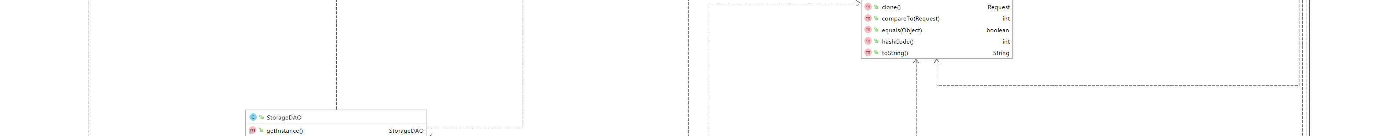
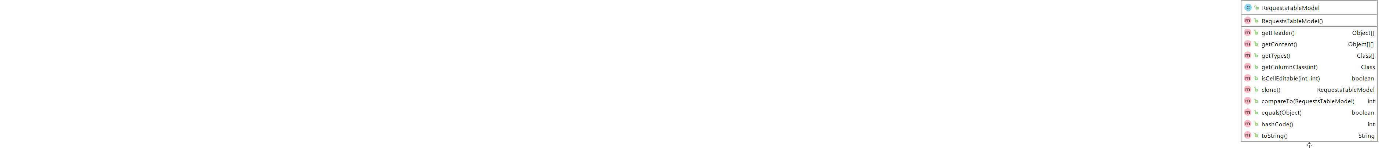


Авторизація

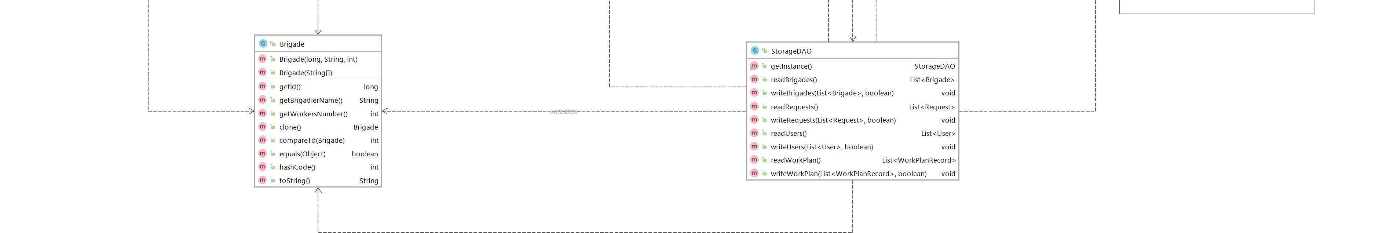
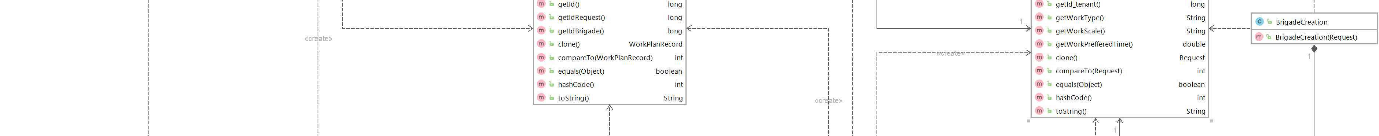
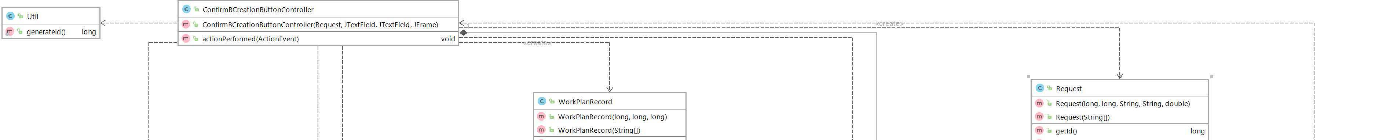


Панель орендара

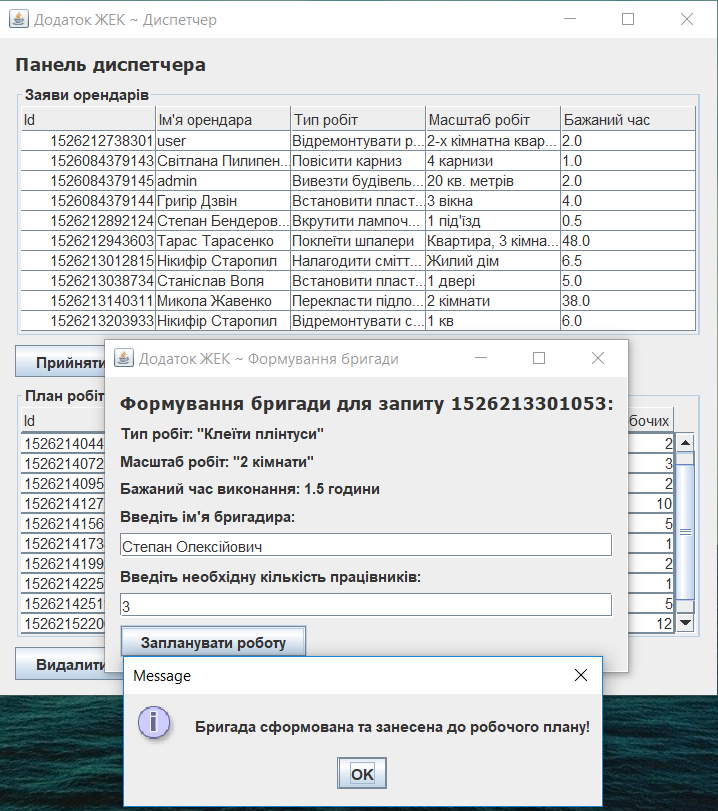
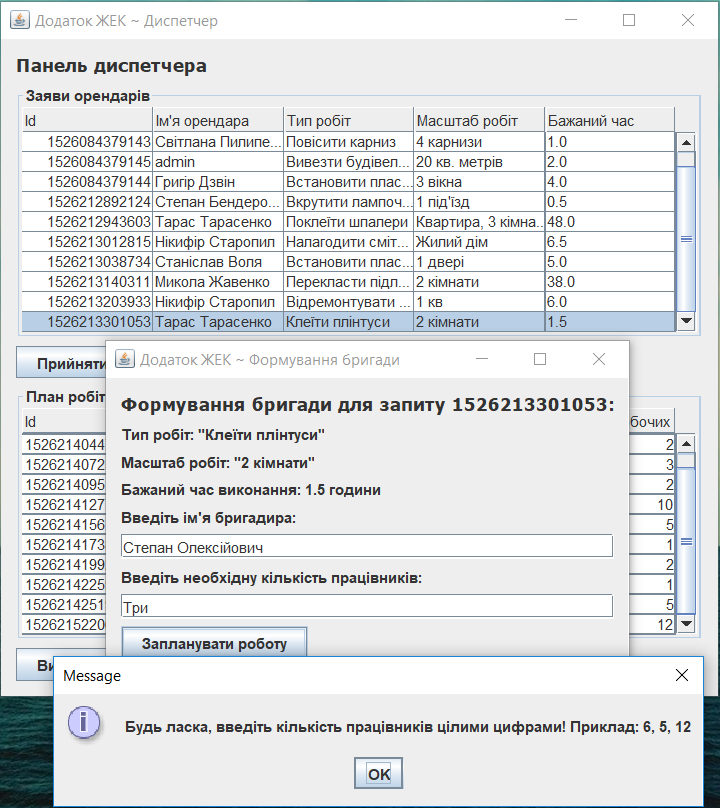
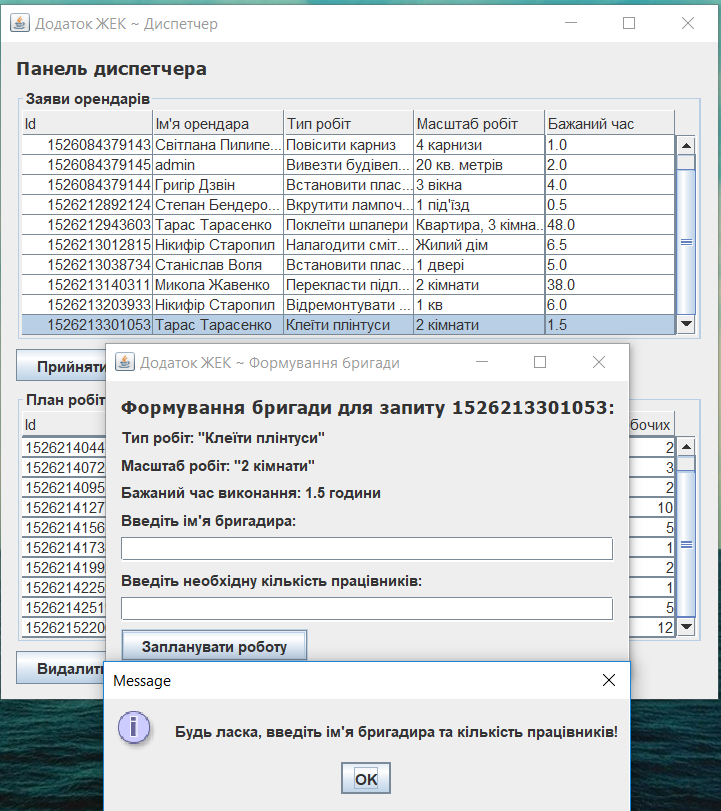
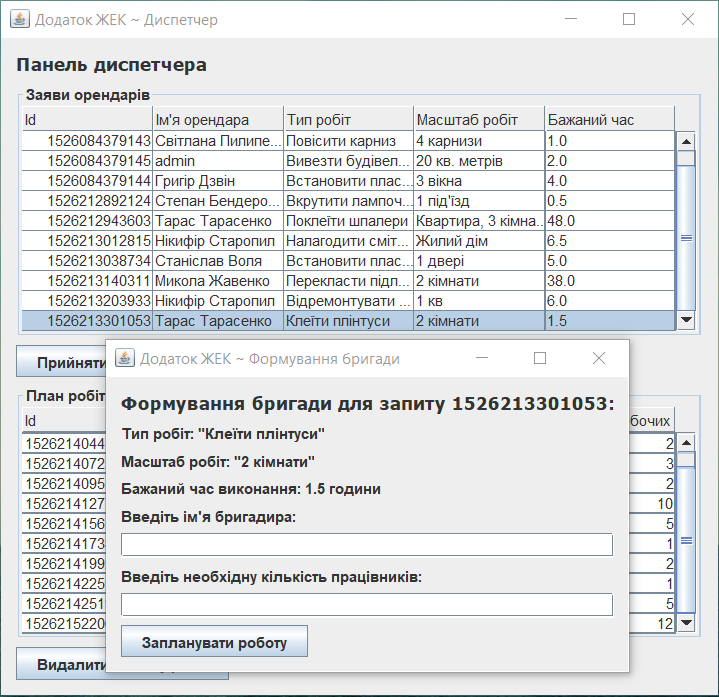
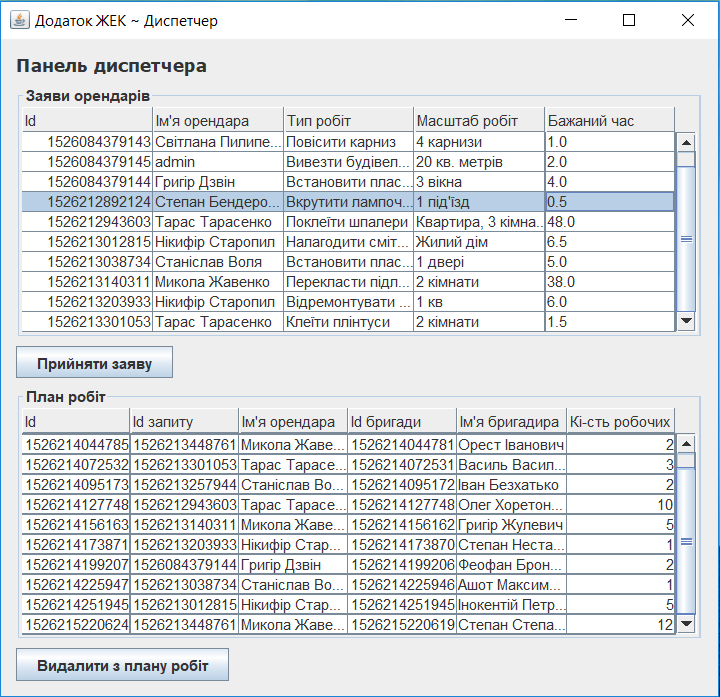
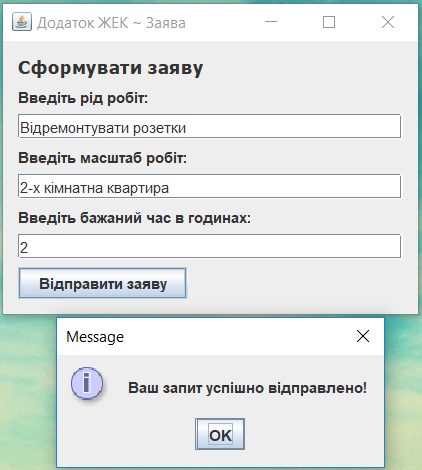
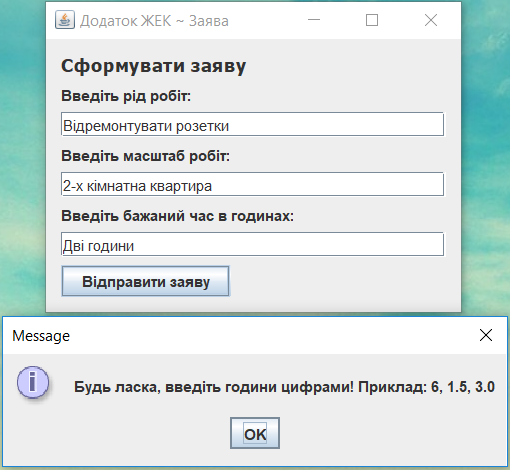
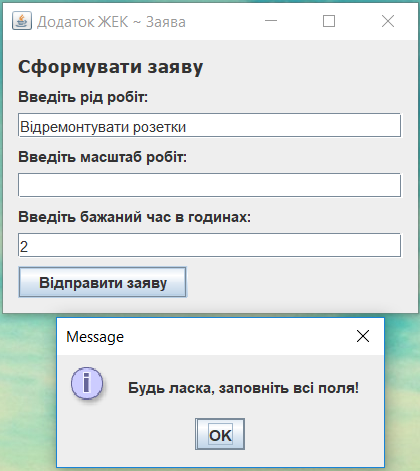
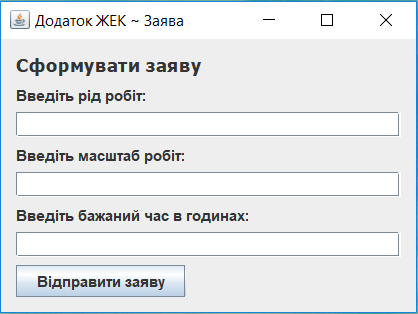
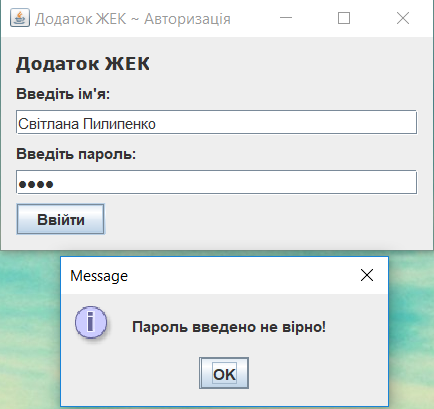
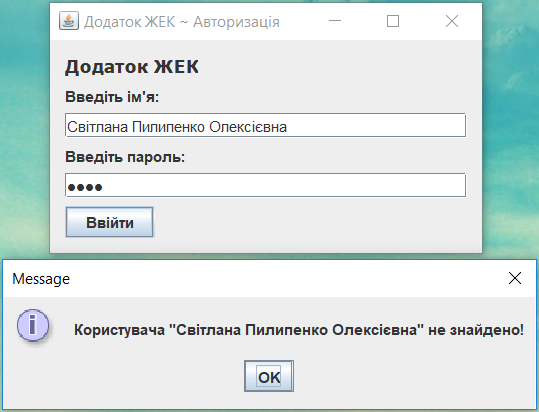
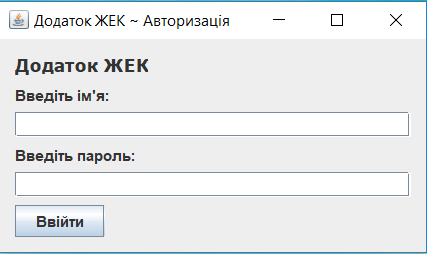
Панель диспетчера



Формування бригади



Скріншоти результатів роботи



Література

[1] Блінов І.Н., Романчик В.С. Java. Методи программування : навч.-мет. видан. /

И.Н. Блінов, В.С. Романчик. — Мінск : видавництво «Четыре четверти», 2013. — 896 с

[2] Скотт К. Java для студента.- СПб.: БХВ- Петербург, 2007.- 448 с.