**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ЛАБОРАТОРНАЯ РАБОТА №3

по дисциплине: **«Визуальные средства разработки программных приложений»**

на тему: **«Потоки ввода-вывода»**

Выполнил: студент группы ИТП-31

Пронуза М.Ю.

Проверил преподаватель:

Ястребов А.А.

Гомель 2024

**Цель работы**: изучить потоки ввода-вывода с использованием сериализации в языке программирования *JAVA*.

**Задание:**

Выполнить задания, сохраняя объекты приложения в одном или нескольких файлах с применением механизма сериализации. Объекты могут содержать поля, помеченные как *static*, а также *transient*.

**Ход выполнения и результаты работы:**

Текстовый вариант задания для варианта 21 представлен на рисунке 1.

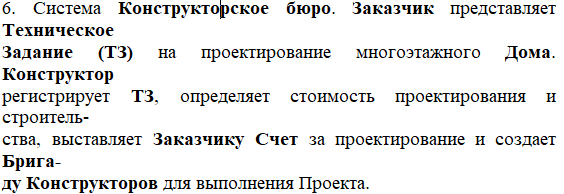


Рисунок 1 – Задание варианта 21

Для выполнения задания использована сериализация между классами. Была получена консольная программа.

На рисунке 2 представлено меню программы.

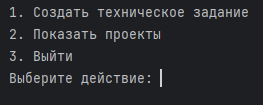


Рисунок 2 – Меню программы

На рисунке 3 представлен результат добавления заявки.

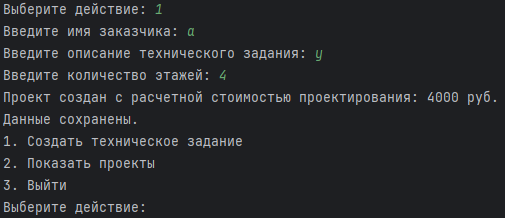


Рисунок 3 – Результат добавления заявки

На рисунке 4 представлен вывод списка заказов.

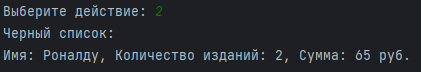


Рисунок 4 – Вывод списка заказов

Текст *Main* представлен в приложении А.

Текст *DataManager* представлен в приложении Б.

Текст *TechnicalTask* представлен в приложении В.

Текст *Client* представлен в приложении Г.

Текст *Bureau* представлен в приложении Д.

Текст *Constructor* представлен в приложении Е.

**Вывод:** в ходе выполнения лабораторной работы изучены потоки ввода-вывода с использованием сериализации в языке программирования *JAVA*. В результате получено консольное приложение, обеспечивающее систему периодических изданий с сериализацией.

**ПРИЛОЖЕНИЕ А**

(обязательное)

**Текст *Main***

import java.io.\*;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

Bureau bureau = new Bureau();

while (true) {

System.out.println("1. Создать техническое задание");

System.out.println("2. Показать проекты");

System.out.println("3. Выйти");

System.out.print("Выберите действие: ");

int choice = scanner.nextInt();

scanner.nextLine();

switch (choice) {

case 1:

createTechnicalTask(scanner, bureau);

break;

case 2:

showProjects(bureau);

break;

case 3:

System.out.println("Завершение программы.");

scanner.close();

return;

default:

System.out.println("Неверный выбор");

}

}

}

private static void createTechnicalTask(Scanner scanner, Bureau bureau) {

System.out.print("Введите имя заказчика: ");

String clientName = scanner.nextLine();

Client client = new Client(clientName);

System.out.print("Введите описание технического задания: ");

String taskDescription = scanner.nextLine();

System.out.print("Введите количество этажей: ");

int numberOfFloors = scanner.nextInt();

TechnicalTask technicalTask = new TechnicalTask(taskDescription, numberOfFloors);

Project project = bureau.createProject(client, technicalTask);

System.out.println("Проект создан с расчетной стоимостью проектирования: " + project.getDesignCost() + " руб.");

try {

DataManager.saveData(bureau, "bureauData.dat");

System.out.println("Данные сохранены.");

} catch (IOException e) {

System.out.println("Ошибка сохранения: " + e.getMessage());

}

}

private static void showProjects(Bureau bureau) {

try {

Bureau loadedBureau = (Bureau) DataManager.loadData("bureauData.dat");

System.out.println("Проекты:");

for (Project project : loadedBureau.getProjects()) {

System.out.println(project);

}

} catch (IOException | ClassNotFoundException e) {

System.out.println("Ошибка загрузки данных: " + e.getMessage());

}

}

}

**ПРИЛОЖЕНИЕ Б**

(обязательное)

**Текст *DataManager***

import java.io.\*;

public class DataManager {

public static void saveData(Object data, String fileName) throws IOException {

try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(fileName))) {

oos.writeObject(data);

}

}

public static Object loadData(String fileName) throws IOException, ClassNotFoundException {

try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(fileName))) {

return ois.readObject();

}

}

}

**ПРИЛОЖЕНИЕ В**

(обязательное)

**Текст *TechnicalTask***

import java.io.Serializable;

class TechnicalTask implements Serializable {

private static final long serialVersionUID = 1L;

private String description;

private int numberOfFloors;

public TechnicalTask(String description, int numberOfFloors) {

this.description = description;

this.numberOfFloors = numberOfFloors;

}

public String getDescription() {

return description;

}

public int getNumberOfFloors() {

return numberOfFloors;

}

@Override

public String toString() {

return "Техническое задание: " + description + ", Количество этажей: " + numberOfFloors;

}

}

**ПРИЛОЖЕНИЕ Г**

(обязательное)

**Текст *Client***

import java.io.Serializable;

class Client implements Serializable {

private static final long serialVersionUID = 1L;

private String name;

public Client(String name) {

this.name = name;

}

public String getName() {

return name;

}

@Override

public String toString() {

return name;

}

}

**ПРИЛОЖЕНИЕ Д**

(обязательное)

**Текст *Bureau***

import java.io.Serializable;

import java.util.ArrayList;

import java.util.List;

class Bureau implements Serializable {

private static final long serialVersionUID = 1L;

private List<Project> projects = new ArrayList<>();

public Project createProject(Client client, TechnicalTask technicalTask) {

Project project = new Project(client, technicalTask);

projects.add(project);

return project;

}

public List<Project> getProjects() {

return projects;

}

}

**ПРИЛОЖЕНИЕ Е**

(обязательное)

**Текст *Constructor***

import java.io.Serializable;

class Constructor implements Serializable {

private static final long serialVersionUID = 1L;

private String name;

public Constructor(String name) {

this.name = name;

}

@Override

public String toString() {

return name;

}

}