Федеральное государственное автономное

образовательное учреждение

Высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

|  |
| --- |
| Институт космических и информационных технологий |
| институт |
| Программная инженерия |
| кафедра |

**ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ**

|  |
| --- |
| Реализация паттернов проектирования |
| тема |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Преподаватель | |  |  |  | Д. В. Грузенкин |
|  | |  | подпись, дата |  | инициалы, фамилия |
| Студент | КИ21-17/1Б, 032156940 |  |  |  | Н. А. Самарин |
|  | номер группы, зачётной книжки |  | подпись, дата |  | инициалы, фамилия |

Красноярск 2024

**СОДЕРЖАНИЕ**

1 Задание............................................................................................................... 3

2 Ход работы........................................................................................................ 3

3 Вывод................................................................................................................. 6

**1 Задание**

Реализовать в том же проекте, в котором выполнялась работа 5, минимум  
три паттерна проектирования (по меньшей мере по одному для каждого вида).  
Можно в рамках одного класса.

**2 Ход работы**

На языке программирования Java были реализованы паттерны проектирования: Фабричный метод, Фасад, Шаблонный метод. Листинги представлены ниже.

Листинг 1 – RecordFactory

package org.example.model;  
public abstract class RecordFactory {  
 public Record createRecord (Object nonSpecificData, Object specificData){  
 Record record = getRecordWithNonSpecificData(nonSpecificData);  
 record = getRecordWithSpecificData(record, specificData);  
 return record;  
 }  
 public abstract Record getRecordWithNonSpecificData(Object nonSpecificData);  
 public abstract Record getRecordWithSpecificData(Record record, Object Specifi  
 cData);  
}

Листинг 2 – VacancyFactoryImpl

package org.example.model;  
public class VacancyFactoryImpl extends RecordFactory{  
 @Override  
 public Record getRecordWithNonSpecificData(Object nonSpecificData) {  
 VacancyImpl vacancy = new VacancyImpl();  
 vacancy.setNonSpecificData(nonSpecificData);  
 return vacancy;  
 }  
 @Override  
 public Record getRecordWithSpecificData(Record record, Object vacancySpecificD  
 ata) {  
 VacancyImpl vacancy = (VacancyImpl) record;  
 vacancy.setVacancySpecificData(vacancySpecificData);  
 return vacancy;  
 }  
}

Листинг 3 – ResumeFactoryImpl

package org.example.model;  
public class ResumeFactoryImpl extends RecordFactory{  
 @Override  
 public Record getRecordWithNonSpecificData(Object nonSpecificData) {  
 ResumeImpl resume = new ResumeImpl();  
 resume.setNonSpecificData(nonSpecificData);  
 return resume;  
 }  
 @Override  
 public Record getRecordWithSpecificData(Record record, Object resumeSpecificDa  
 ta) {  
 ResumeImpl resume = (ResumeImpl) record;  
 resume.setResumeSpecificData(resumeSpecificData);

Окончание листинга 3  
 return resume;  
 }

Листинг 4 – RecordController

package org.example.controller;  
import org.example.database.RecordDataRepository;  
import org.example.model.\*;  
import org.example.model.Record;  
import org.example.service.PaymentSystem;  
public class RecordController {  
 RecordFactory recordFactory;  
 PaymentSystem paymentSystem;  
 RecordDataRepository recordDataRepository;  
 public RecordController(RecordDataRepository recordDataRepository, RecordType   
 type) {  
 switch (type) {  
 case VACANCY -> recordFactory = new VacancyFactoryImpl();  
 case RESUME -> recordFactory = new ResumeFactoryImpl();  
 }  
 this.recordDataRepository = recordDataRepository;  
 this.paymentSystem = PaymentSystem.getInstance();  
 }  
 public void registerRecord(Object nonSpecificRecordData, Object specificRecord  
 Data) {  
 Record record = recordFactory.createRecord(nonSpecificRecordData, specificRe  
 cordData);  
 // Работа с recordDataRepository  
 }  
 public void promoteRecord(Record record){  
 // Работа с paymentSystem  
 }  
}

Листинг 5 – EmailSystem

package org.example.service;

import java.util.Properties;

import javax.mail.Message;

import javax.mail.MessagingException;

import javax.mail.Session;

import javax.mail.Transport;

import javax.mail.internet.InternetAddress;

import javax.mail.internet.MimeMessage;

public class EmailSystem {

public static void sendEmail(String to, String text, String subject){

String from = "sender@abc.com"; // sender email

String host = "127.0.0.1"; // mail server host

Properties properties = System.getProperties();

properties.setProperty("mail.smtp.host", host);

Session session = Session.getDefaultInstance(properties); // default session

try {

MimeMessage message = new MimeMessage(session); // email message

message.setFrom(new InternetAddress(from)); // setting header fields

message.addRecipient(Message.RecipientType.TO, new InternetAddress(to));

message.setSubject(subject); // subject line

Окончание листинга 5

message.setText(text);

Transport.send(message);

} catch (MessagingException mex){ mex.printStackTrace(); }

}

}

**3 Вывод**

В ходе работы на языке программирования Java были реализованы  
паттерны проектирования.