Федеральное государственное автономное

образовательное учреждение

Высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

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

**ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ**

|  |
| --- |
| Реализация принципов SOLID |
| 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................................................................................................................. 7

**1 Задание**

Реализовать на любом из знакомых вам языков программирования не  
менее трёх принципов SOLID для модулей внутри одного из выбранных вами  
компонентов, которые вы описывали в предыдущей работе (в терминах  
нотации C4 для компонентов внутри одного из контейнеров). Код не должен  
содержать какую-либо логику и реализованные бизнес-функции - необходима  
только сама структура связей.

**2 Ход работы**

На языке программирования Java были реализованы принципы SOLID на  
примере компонентов из диаграммы из предыдущей работы. Реализованы  
принципы: Принцип единственной ответственности, принцип инверсии  
зависимостей, принцип разделения интерфейсов, принцип подстановки  
Барбары Лисков. Листинги представлены ниже.

Листинг 1 – User

package org.example.model;  
public class User {  
 Object userData;  
 public Object getUserData() {  
 return userData;  
 }  
 public void setUserData(Object userData) {  
 this.userData = userData;  
 }  
}

Листинг 2 – UserImpl

package org.example.model;  
public class UserImpl extends User{  
}

Листинг 3 – UserFactory

package org.example.model;  
public abstract class UserFactory {  
 public abstract User createUser ();  
}

Листинг 4 – UserFactoryImpl

public class UserFactoryImpl extends UserFactory {  
 @Override  
 public User createUser() {  
 User user = new UserImpl();  
 return user;  
 }  
}

Листинг 5 – Record

package org.example.model;  
public abstract class Record {  
 Object nonSpecificData;  
 public Object getNonSpecificData() {

Окончание листинга 5

return nonSpecificData;  
 }  
 public void setNonSpecificData(Object nonSpecificData) {  
 this.nonSpecificData = nonSpecificData;  
 }  
}

Листинг 6 – IVacancy

package org.example.model;  
public interface IVacancy{  
 public Object getVacancySpecificData();  
 public void setVacancySpecificData(Object vacancySpecificData);  
}

Листинг 7 – IResume

package org.example.model;  
public interface IResume {  
 public Object getResumeSpecificData();  
 public void setResumeSpecificData(Object resumeSpecificData);  
}

Листинг 8 – ResumeImpl

package org.example.model;  
public class ResumeImpl extends Record implements IResume{  
 Object resumeSpecificData;  
 @Override  
 public Object getResumeSpecificData() {  
 return resumeSpecificData;  
 }  
 @Override  
 public void setResumeSpecificData(Object resumeSpecificData) {  
 this.resumeSpecificData = resumeSpecificData;  
 }  
}

Листинг 9 – VacancyImpl

package org.example.model;  
public class VacancyImpl extends Record implements IVacancy{  
 Object vacancySpecificData;  
 @Override  
 public Object getVacancySpecificData() {  
 return vacancySpecificData;  
 }  
 @Override  
 public void setVacancySpecificData(Object vacancySpecificData) {  
 this.vacancySpecificData = vacancySpecificData;  
 }  
}

Листинг 10 – 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;

Окончание листинга 10

}  
 public abstract Record getRecordWithNonSpecificData(Object nonSpecificData);  
 public abstract Record getRecordWithSpecificData(Record record, Object Specifi  
 cData);  
}

Листинг 11 – 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 12 – 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);  
 return resume;  
 }  
}

Листинг 13 – UserController

package org.example.controller;  
import org.example.database.UserDataRepository;  
import org.example.model.User;  
import org.example.model.UserFactory;  
import org.example.service.EmailSystem;  
import org.example.service.PaymentSystem;  
public class UserController {  
 UserFactory userFactory;  
 EmailSystem emailSystem;  
 UserDataRepository userDataRepository;  
 public UserController(UserFactory userFactory, UserDataRepository userDataRepo  
 sitory) {  
 this.userFactory = userFactory;  
 this.userDataRepository = userDataRepository;

Окончание листинга 13

this.emailSystem = EmailSystem.getInstance();  
 }  
 public User createUser() {  
 User user = userFactory.createUser();  
 return user;  
 }  
 public void sendConfirmationEmail(User user){}  
}

Листинг 14 – 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 15 – AdministratorController

package org.example.controller;  
import org.example.database.RecordDataRepository;  
import org.example.database.UserDataRepository;  
import org.example.model.User;  
public class AdministratorController {  
 UserDataRepository userDataRepository;  
 RecordDataRepository recordDataRepository;  
 public AdministratorController(UserDataRepository userDataRepository, RecordDa  
 taRepository recordDataRepository) {  
 this.userDataRepository = userDataRepository;  
 this.recordDataRepository = recordDataRepository;  
 }  
 void deleteUser(User user){}  
 void deleteRecord(User user){}  
}

Листинг 16 – RecordDataRepository

package org.example.database;  
import org.example.model.IResume;  
import org.example.model.IVacancy;  
public class RecordDataRepository {  
 public void insertVacancy(){}  
 public void insertResume(){}  
 public IVacancy getVacancy(){  
 IVacancy vacancy = null;  
 return vacancy;  
 }  
 public IResume getResume(){  
 IResume resume = null;  
 return resume;  
 }  
}

Листинг 17 – UserDataRepository

package org.example.database;  
import org.example.model.User;  
public class UserDataRepository {  
 public void insertUser(){

//работа с SQL

}

//public User getUser(){

//работа с SQL

//}  
}

Листинг 18 – EmailSystem

package org.example.service;  
public final class EmailSystem {  
 private static EmailSystem instance;  
 void sendEmail(){}  
 public static EmailSystem getInstance() {  
 if (instance == null) {  
 instance = new EmailSystem();  
 }  
 return instance;  
 }  
}

Листинг 19 – PaymentSystem

package org.example.service;  
public final class PaymentSystem {  
 private static PaymentSystem instance;  
 void makePayment(){}  
 public static PaymentSystem getInstance() {  
 if (instance == null) {  
 instance = new PaymentSystem();  
 }  
 return instance;  
 }  
}

**3 Вывод**

В ходе работы на языке программирования Java были реализованы  
принципы SOLID.