

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №4

Специальность ПОЗ

Выполнила Р.И.
Гаврилюк, студентка
группы ПОЗ

Проверил А.А.
Крощенко,
ст. преп. кафедры ИИТ,
«—» ————— 2020 г.

Брест 2020

Вариант 6

Цель работы: приобрести практические навыки в области объектно-ориентированного проектирования.

Задание 1.

Реализовать указанный класс, включив в него вспомогательный внутренний класс или классы. Реализовать 2-3 метода (на выбор). Продемонстрировать использование реализованных классов.

6) Создать класс Catalog (каталог) с внутренним классом, с помощью объектов которого можно хранить информацию об истории выдач книги читателям.

Код программы

Catalog.java

```
import java.util.ArrayList;

class Catalog{
    public class Book{
        private String mNameBook;
        private String mDate;
        private String mReader;

        public Book(String nameBook, String date, String reader){
            this.mNameBook = nameBook;
            this.mDate = date;
            this.mReader = reader;
        }

        public String getNameBook(){
            return this.mNameBook;
        }

        public String getDate(){
            return this.mDate;
        }

        public String setReader(){
            return this.mReader;
        }

        public void setNameBook(String nameBook){
            this.mNameBook = nameBook;
        }

        public void getDate(String date){
            this.mDate = date;
        }

        public void getReader(String reader){
            this.mReader = reader;
        }

        public String toString(){
```

```

        return "Book: " + mNameBook +
            "\nDate: " + mDate +
            "\nReader: " + mReader;
    }
}

private ArrayList<Catalog.Book> books;

public Catalog(){
    books = new ArrayList<>();
}

public void addBook(String nameBook, String date, String reader){
    Book book = new Book(nameBook, date, reader);
    books.add(book);
}

public void addBook(Book book){
    books.add(book);
}

public void showBooks(){
    for(Book book : books){
        System.out.println(book.toString());
    }
}
}

```

Main.java

```

class Laba4{
    public static void main(String[] args) {
        Catalog catalog = new Catalog();
        catalog.addBook("book 1", "12.03.2000", "reader 1");
        catalog.addBook("book 2", "13.04.2001", "reader 2");
        catalog.addBook("book 3", "14.05.2002", "reader 3");
        catalog.addBook("book 4", "15.06.2003", "reader 4");
        catalog.addBook("book 5", "16.07.2004", "reader 5");
        catalog.showBooks();
    }
}

```

Рисунки с результатами работы программы

```

Book: book 1
Date: 12.03.2000
Reader: reader 1
Book: book 2
Date: 13.04.2001
Reader: reader 2
Book: book 3
Date: 14.05.2002
Reader: reader 3
Book: book 4
Date: 15.06.2003
Reader: reader 4
Book: book 5
Date: 16.07.2004
Reader: reader 5

```

Задание 2.

Реализовать агрегирование. При создании класса агрегируемый класс объявляется как атрибут (локальная переменная, параметр метода). Включить в каждый класс 2-3 метода на выбор. Продемонстрировать использование разработанных классов.

6) Создать класс Страница, используя класс Слово.

Код программы

Page.java

```
import java.util.Vector;

class Page{
    private Vector<Word> mPage;

    public Page(){
        mPage = new Vector<>();
    }

    public void addWord(Word word){
        mPage.add(word);
    }

    public String toString(){
        String str = new String();
        for (Word word : mPage) {
            str += word.getWord();
            if(!word.getWord().equals("\n")){
                str += " ";
            }
        }
        return str;
    }
}
```

Word.java

```
class Word{
    private String mWord;

    public Word(String word){
        this.mWord = word;
    }

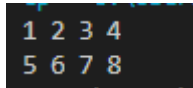
    public void setWord(String word){
        this.mWord = word;
    }

    public String getWord(){
        return this.mWord;
    }
}
```

Main.java

```
class Laba4{
    public static void main(String[] args) {
        //task 2
        Page page = new Page();
        page.addWord(new Word("1"));
        page.addWord(new Word("2"));
        page.addWord(new Word("3"));
        page.addWord(new Word("4"));
        page.addWord(new Word("\n"));
        page.addWord(new Word("5"));
        page.addWord(new Word("6"));
        page.addWord(new Word("7"));
        page.addWord(new Word("8"));
        System.out.println(page.toString());
    }
}
```

Рисунки с результатами работы программы



```
1 2 3 4
5 6 7 8
```

Задание 3.

Построить модель программной системы с применением отношений (обобщения, агрегации, ассоциации, реализации) между классами. Задать атрибуты и методы классов. Реализовать (если необходимо) дополнительные классы. Продемонстрировать работу разработанной системы.

6) Система **Телефонная станция**. **Абонент** оплачивает **Счет** за разговоры и **Услуги**, может попросить **Администратора** сменить **номер** и отказаться от **услуг**. **Администратор** изменяет **номер**, **Услуги** и временно отключает **Абонента** за неуплату.

Код программы

main.java

```
import java.util.Vector;

import Service.mService;

class Laba4{
    public static void main(String[] args) {
        //task 3
        Administrator admin = new Administrator("ADMIN");

        Vector<Service> services = new Vector<>();
        Service serviceSMS = new ServiceSMS();
        Service serviceMMS = new ServiceMMS();
        Service serviceInternet = new ServiceInternet();
        services.add(serviceSMS);
        services.add(serviceMMS);
        services.add(serviceInternet);
    }
}
```

```

    Abonent abonent_1 = new Abonent("KATE", "+375-29-111-99-55", services, admin);
    System.out.println(abonent_1.toString());

    admin.addAbonent(abonent_1);

    abonent_1.useService(serviceMMS);
    abonent_1.turnOnService(serviceMMS);
    abonent_1.useService(serviceMMS);
    abonent_1.ring();
    abonent_1.ring();

    Bill bill = new Bill();
    abonent_1.payBill(bill);

    abonent_1.turnOffService(serviceSMS);
    abonent_1.changeNumber("+375-33-000-11-87");
    System.out.println(abonent_1.toString());

    admin.blockAbonent(abonent_1);
    abonent_1.ring();
    abonent_1.useService(serviceInternet);
    admin.addAbonent(abonent_1);
    admin.blockAbonent(abonent_1);
    abonent_1.payBill(bill);
    admin.unblockAbonent(abonent_1);
}
}

```

service.java

```

class Service{

    public enum mService {
        SMS,
        MMS,
        INTERNET
    }

    private mService mServiceName;
    private boolean mIsTurnedOn;
    private boolean mIsPaid;

    public Service(mService serviceName){
        this.mServiceName = serviceName;
        this.mIsTurnedOn = false;
        this.mIsPaid = true;
    }

    public mService getServiceName(){
        return this.mServiceName;
    }

    public void setServiceName(mService serviceName){
        this.mServiceName = serviceName;
    }
}

```

```
}

public boolean getIsTurnedOn(){
    return this.mIsTurnedOn;
}

public boolean getIsPaid(){
    return this.mIsPaid;
}

public void addService(){
    System.out.print("Service ");
    System.out.print(this.toString());
    System.out.println(" added");
    this.mIsTurnedOn = true;
    this.mIsPaid = true;
}

public boolean turnOffService(){
    //if(something.wrong())
    //return false;
    this.mIsTurnedOn = false;
    return true;
}

public boolean turnOnService(){
    //if(something.wrong())
    //return false;
    this.mIsTurnedOn = true;
    return true;
}

public boolean payForService(){
    if(!this.mIsPaid){
        this.mIsPaid = true;
        System.out.print("Service ");
        System.out.print(this.toString());
        System.out.println(" was paid");
        return true;
    }
    else{
        System.out.print("Service ");
        System.out.print(this.toString());
        System.out.println(" has been already paid");
        return false;
    }
}

public void useService(){
    if(this.mIsPaid && this.mIsTurnedOn){
        System.out.print("Service ");
        System.out.print(this.toString());
        System.out.println(" is used. Money is disappearing...");
        this.mIsPaid = false;
    }
}
```

```

        else if(!this.mIsPaid && this.mIsTurnedOn
            || !this.mIsPaid && !this.mIsTurnedOn){
            System.out.print("Service ");
            System.out.print(this.toString());
            System.out.println(" isn't paid. Service is turned off...");
            this.mIsTurnedOn = false;
        }
        else if(this.mIsPaid && !this.mIsTurnedOn){
            System.out.print("Service ");
            System.out.print(this.toString());
            System.out.println(" isn't turned on");
        }
    }

    public String toString(){
        String serviceString = new String();
        if(this.mServiceName == this.mServiceName.SMS){
            serviceString = "SMS";
        }
        else if(this.mServiceName == this.mServiceName.MMS){
            serviceString = "MMS";
        }
        else if(this.mServiceName == this.mServiceName.INTERNET){
            serviceString = "INTERNET";
        }
        return serviceString;
    }
}

```

serviceMMS.java

```

class ServiceMMS extends Service{
    public ServiceMMS() {
        super(Service.mService.MMS);
    }
}

```

serviceSMS.java

```

class ServiceSMS extends Service{
    public ServiceSMS() {
        super(Service.mService.SMS);
    }
}

```

serviceInternet.java

```

class ServiceInternet extends Service{
    public ServiceInternet() {
        super(Service.mService.INTERNET);
    }
}

```


number.java

```
class Number{
    private String mNumber;
    private boolean mIsPaid;

    public Number(String number){
        this.mNumber = number;
        this.mIsPaid = true;
    }

    public void setNumber(String number){
        this.mNumber = number;
    }

    public String getNumber(){
        return this.mNumber;
    }

    public boolean getIsPaid(){
        return this.mIsPaid;
    }

    public boolean changeNumber(String number){
        //if(smith.wrong())
        //return false;

        System.out.print("Number ");
        System.out.print(this.mNumber);
        System.out.print(" was changed to ");
        System.out.println(number);
        this.setNumber(number);
        return true;
    }

    public boolean payForTalk(){
        if(!this.mIsPaid){
            this.mIsPaid = true;
            System.out.print("Number ");
            System.out.print(this.mNumber);
            System.out.println(" was paid");
            return true;
        }
        else {
            System.out.print("Number ");
            System.out.print(this.mNumber);
            System.out.println(" has been already paid");
            return false;
        }
    }

    public void ring(){
        if(this.mIsPaid){
            System.out.print("Number ");
            System.out.print(this.mNumber);
        }
    }
}
```

```

        System.out.println(" is ringing. Money is disappearing...");
        this.mIsPaid = false;
    }
    else{

        System.out.print("Number ");
        System.out.print(this.mNumber);
        System.out.println(" isn't paid");
    }
}
}
}

```

bill.java

```

import java.util.Vector;

class Bill{

    public boolean pay(Number number, Vector<Service> services){
        //if(smth.wrong())
        //return false;
        number.payForTalk();
        for(Service service : services){
            if(!service.getIsPaid()){
                service.payForService();
            }
            else {
                System.out.print("Service ");
                System.out.print(service.toString());
                System.out.println(" has been already paid");
            }
        }

        System.out.println("Bill is paid");
        return true;
    }

    public boolean payForNumber(Number number){
        if(!number.getIsPaid()){
            number.payForTalk();
            return true;
        }
        else{
            System.out.print("Number ");
            System.out.print(number);
            System.out.println(" has been already paid");
            return false;
        }
    }

    public boolean payForService(Service service){
        if(!service.getIsPaid()){
            service.payForService();
            return true;
        }
    }
}

```

```

        else{
            System.out.print("Service ");
            System.out.print(service.toString());
            System.out.println(" has been already paid");
            return false;
        }
    }
}

```

user.java

```

class User{

    private String mName;
    private mPosition mPos;

    public enum mPosition{
        ABONENT,
        ADMINISTRATOR
    }

    public User(String name, mPosition position){
        this.mName = name;
        this.mPos = position;
    }

    public String getName(){
        return this.mName;
    }

    public void setName(String name){
        this.mName = name;
    }

    public mPosition getPosition(){
        return this.mPos;
    }

    public void setPosition(mPosition position){
        this.mPos = position;
    }

    public String toString(){
        String user = "Name : " + this.mName;
        if(this.mPos == this.mPos.ABONENT){
            user += "\nPosition : abonent";
        }
        else if(this.mPos == this.mPos.ADMINISTRATOR){
            user += "\nPosition : administrator";
        }
        return user;
    }
}

```

abonent.java

```
import java.util.Vector;

class Abonent extends User{

    private Number mNumber;
    private Vector<Service> mServices;
    private boolean mIsBlocked;
    private Administrator mAdministrator;

    public Abonent(String name, String number, Vector<Service> services,
                    Administrator administrator) {
        super(name, User.mPosition.ABONENT);
        this.mNumber = new Number(number);
        this.mServices = services;
        this.mAdministrator = administrator;
        this.mIsBlocked = false;
    }

    public boolean isAbonentBlocked(){
        return this.mIsBlocked;
    }

    public boolean isNumberPaid(){
        return this.mNumber.getIsPaid();
    }

    public void blockAbonent(){
        this.mIsBlocked = true;
        System.out.print("Abonent ");
        System.out.print(super.getName());
        System.out.println(" is blocked");
    }

    public void unblockAbonent(){
        this.mIsBlocked = false;
        System.out.print("Abonent ");
        System.out.print(super.getName());
        System.out.println(" is unblocked");
    }

    public boolean payBill(Bill bill){
        System.out.print("Abonent ");
        System.out.print(super.getName());
        System.out.println(" pays for bill");

        return bill.pay(mNumber, mServices);
    }

    public boolean payForNumber(Bill bill){
        System.out.print("Abonent ");
        System.out.print(super.getName());
        System.out.println(" pays for telephone");
    }
}
```

```

        return bill.payForNumber(mNumber);
    }

    public boolean payForService(Bill bill, int idService){
        System.out.print("Abonent ");
        System.out.print(super.getName());
        System.out.println(" pays for service");

        return bill.payForService(mServices.elementAt(idService));
    }

    public boolean addService(Service service){
        if(!mServices.contains(service)){
            mServices.add(service);
            return true;
        }
        return false;
    }

    public int findService(Service findedService){
        int i = 0;
        for(Service service : this.mServices){
            if(service.getServiceName().equals(findedService.getServiceName())){
                return i;
            }
            ++i;
        }
        return -1;
    }

    public void useService(Service service){
        if(!this.mIsBlocked){
            int id = findService(service);
            if(id != -1){
                this.mServices.elementAt(id).useService();
            }
            else{
                System.out.print("Abonent ");
                System.out.print(super.getName());
                System.out.println(" can't use service. Service isn't added");
            }
        }
        else{
            System.out.print("Abonent ");
            System.out.print(super.getName());
            System.out.println(" can't use service, because he is blocked");
        }
    }

    public void ring(){
        if(!this.mIsBlocked){
            this.mNumber.ring();
        }
        else{
            System.out.print("Abonent ");

```

```

        System.out.print(super.getName());
        System.out.println(" can't ring, because he is blocked");
    }
}

public boolean changeNumber(String number){
    return this.mAdministrator.changeNumber(this, mNumber, number);
}

public String getNumber(){
    return this.getNumber();
}

public boolean turnOffService(Service service){
    return this.mAdministrator.turnOffService(this, service);
}

public boolean turnOnService(Service service){
    return this.mAdministrator.turnOnService(this, service);
}

public String toString(){
    String user = super.toString();
    user += "\nNumber : " + mNumber.getNumber();
    return user;
}
}

```

administrator.java

```

import java.util.Vector;

class Administrator extends User{
    private Vector<Abonent> mAbonents;

    public Administrator(String name) {
        super(name, User.mPosition.ADMINISTRATOR);
        mAbonents = new Vector<>();
    }

    public Vector<Abonent> getAbonents(){
        return this.mAbonents;
    }

    public int findAbonent(Abonent findedAbonent){
        int i = 0;
        for(Abonent abonent : this.mAbonents){
            if(abonent.getName().equals(findedAbonent.getName())){
                return i;
            }
            ++i;
        }
        return -1;
    }
}

```

```

public boolean addAbonent(Abonent abonent){
    if(this.findAbonent(abonent) == -1){
        this.mAbonents.add(abonent);
        System.out.print("Abonent ");
        System.out.print(abonent.getName());
        System.out.print(" added to administrator ");
        System.out.println(this.getName());
        return true;
    }
    else {
        System.out.print("Abonent ");
        System.out.print(abonent.getName());
        System.out.print(" has been already added to administrator ");
        System.out.println(this.getName());
        return false;
    }
}

public boolean deleteAbonent(Abonent abonent, String number){
    int index = this.findAbonent(abonent);
    if(index != -1){
        this.mAbonents.remove(index);
        System.out.print("Abonent ");
        System.out.print(abonent.getName());
        System.out.print(" was removed from administrator ");
        System.out.println(this.getName());
        return true;
    }
    else {
        System.out.print("Abonent ");
        System.out.print(abonent.getName());
        System.out.print(" isn't controlled by administrator ");
        System.out.println(this.getName());
        return false;
    }
}

public boolean changeNumber(Abonent abonent, Number number, String newNumber){
    int index = this.findAbonent(abonent);
    if(index != -1){
        number.setNumber(newNumber);
        System.out.print("New number of abonent ");
        System.out.print(abonent.getName());
        System.out.print(" is ");
        System.out.println(newNumber);
        return true;
    }
    else {
        System.out.print("Abonent ");
        System.out.print(abonent.getName());
        System.out.print(" isn't controlled by administrator ");
        System.out.println(this.getName());
        return false;
    }
}

```

```

}

public boolean blockAbonent(Abonent abonent){
    int index = this.findAbonent(abonent);
    if(index != -1){
        if(!this.mAbonents.elementAt(index).isNumberPaid()){
            this.mAbonents.elementAt(index).blockAbonent();
            return true;
        }
        else{
            System.out.print("Abonent ");
            System.out.print(abonent.getName());
            System.out.println(" isn't blocked. Number is paid");
        }
    }
    else {
        System.out.print("Abonent ");
        System.out.print(abonent.getName());
        System.out.print(" isn't controlled by administrator ");
        System.out.println(this.getName());
    }
    return false;
}

public boolean unblockAbonent(Abonent abonent){
    int index = this.findAbonent(abonent);
    if(index != -1){
        if(this.mAbonents.elementAt(index).isNumberPaid()){
            this.mAbonents.elementAt(index).unlockAbonent();
            return true;
        }
        else{
            System.out.print("Abonent ");
            System.out.print(abonent.getName());
            System.out.println(" isn't unblocked. Number isn't paid");
        }
    }
    else {
        System.out.print("Abonent ");
        System.out.print(abonent.getName());
        System.out.print(" isn't controlled by administrator ");
        System.out.println(this.getName());
    }
    return false;
}

public boolean turnOffService(Abonent abonent, Service service){
    int index = this.findAbonent(abonent);
    if(index != -1){
        if(this.mAbonents.elementAt(index).findService(service) != -1){
            service.turnOffService();
            System.out.print("Service ");
            System.out.print(service.getServiceName());
            System.out.print(" was turned off for abonent ");
            System.out.println(abonent.getName());
        }
    }
}

```



```

        return true;
    }
    else{
        System.out.print("Abonent ");
        System.out.print(abonent.getName());
        System.out.print(" hasn't got service ");
        System.out.println(service.getServiceName());
    }
}
else {
    System.out.print("Abonent ");
    System.out.print(abonent.getName());
    System.out.print(" isn't controlled by administrator ");
    System.out.println(this.getName());
}
return false;
}

public boolean turnOnService(Abonent abonent, Service service){
    int index = this.findAbonent(abonent);
    if(index != -1){
        if(this.mAbonents.elementAt(index).findService(service) != -1){
            service.turnOnService();
            System.out.print("Service ");
            System.out.print(service.getServiceName());
            System.out.print(" was turned on for abonent ");
            System.out.println(abonent.getName());
            return true;
        }
        else{
            System.out.print("Abonent ");
            System.out.print(abonent.getName());
            System.out.print(" hasn't got service ");
            System.out.println(service.getServiceName());
        }
    }
    else {
        System.out.print("Abonent ");
        System.out.print(abonent.getName());
        System.out.print(" isn't controlled by administrator ");
        System.out.println(this.getName());
    }
    return false;
}
}
}

```

Рисунки с результатами работы программы

```
Name : KATE
Position : abonent
Number : +375-29-111-99-55
Abonent KATE added to administrator ADMIN
Service MMS isn't turned on
Service MMS was turned on for abonent KATE
Service MMS is used. Money is disappearing...
Number +375-29-111-99-55 is ringing. Money is disappearing...
Number +375-29-111-99-55 isn't paid
Abonent KATE pays for bill
Number +375-29-111-99-55 was paid
Service SMS has been already paid
Service MMS was paid
Service INTERNET has been already paid
Bill is paid
Service SMS was turned off for abonent KATE
New number of abonent KATE is +375-33-000-11-87
Name : KATE
Position : abonent
Number : +375-33-000-11-87
Abonent KATE isn't blocked. Number is paid
Number +375-33-000-11-87 is ringing. Money is disappearing...
Service INTERNET isn't turned on
Abonent KATE has been already added to administrator ADMIN
Abonent KATE is blocked
Abonent KATE pays for bill
Number +375-33-000-11-87 was paid
Service SMS has been already paid
Service MMS has been already paid
Service INTERNET has been already paid
Bill is paid
Abonent KATE is unblocked
```

Вывод: приобрела практические навыки в области объектно-ориентированного проектирования.