

Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

ОТЧЕТ
Лабораторная работа №4

Выполнила:
Студентка 4 курса
Группы АС-50
Клиницкая Р.П.
Проверил:
Крощенко А.А.

Брест 2020

Цель работы:

приобрести практические навыки в области объектно-ориентированного проектирования.

Задание. Вариант 5.

1. Создать класс Department (отдел фирмы) с внутренним классом, с помощью объектов которого можно хранить информацию обо всех должностях отдела и обо всех сотрудниках, когда-либо занимавших конкретную должность.

Код программы:

```
package com.company;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
//делаем общий класс, который будет содержать вложенные классы
class Department {
    //создадим класс, содержащий информацию о должностях
    class Roles {
        //так как это должности, то уделние отсутствует
        private int id;
        private String name;

        public Roles(int id, String name){
            this.id = id;
            this.name = name;
        }

        public String forPrint(){
            return "id="+id+", должность="+name;
        }
    }

    //внутренний клас, содержащий информацию о работниках фирмы!
    class Worker{
        private int id;
        private String FIO = "";
        private Roles roles;
        private LocalDate whenStartWork = null;
        private LocalDate whenEndWork = null;

        //добавление информации о рабочем
        public Worker(int id, String FIO, Roles roles){
            this.id = id;
            this.FIO = FIO;
            this.roles = roles;
            this.whenStartWork = LocalDate.now();
        }

        //просто вывод на экран информации по рабочему
        @Override
        public String toString(){
            StringBuilder builder = new StringBuilder();
            builder.append("id: "+id+", фио: "+ FIO +", должность: "+
            roles.name+", статус: ");
            if(whenEndWork ==null) {
                builder.append("работает"+", дата трудоустройства: "
                +
                whenStartWork.format(DateTimeFormatter.ofPattern("dd/MM/yyyy")));
            }
            else
```

```

        builder.append("уволен"+"", дата трудоустройства: "
            +
whenStartWork.format(DateTimeFormatter.ofPattern("dd/MM/yyyy"))+"", дата
увольнения: "
            +
whenEndWork.format(DateTimeFormatter.ofPattern("dd/MM/yyyy")));

        return builder.toString();
    }
}
//закончили создание классов. пишем функционал для main

//содадим хранилище для рабочих и их должностей
ArrayList<Roles> roles = new ArrayList<Roles>();
ArrayList<Worker> workers = new ArrayList<Worker>();

//начальные инициализации должностей
public Department() {
    roles.add(new Roles(1, "директор"));
    roles.add(new Roles(2, "секретарь"));
    roles.add(new Roles(3, "бухгалтер"));
    roles.add(new Roles(4, "сотрудник"));
}

public void showRoles() {
    for(int i = 0; i< roles.size(); i++)
        System.out.println(roles.get(i).forPrint());
}

public void showWorkers() {
    for(int i=0; i<workers.size(); i++)
        System.out.println(workers.get(i).toString());
}

public void addWorker(int workerId, String fio, int functionId){
    Roles roles = null;
    for(int i = 0; i< this.roles.size(); i++)
        if(this.roles.get(i).id==functionId)
        {
            roles = this.roles.get(i);
            break;
        }

    if(roles ==null)
        return;

    Worker worker = new Worker(workerId, fio, roles);
    workers.add(worker);
}

public void dismissWorker(int workerId){
    for(int i=0; i<workers.size(); i++)
        if(workers.get(i).id==workerId)
        {
            workers.get(i).whenEndWork = LocalDate.now();
            return;
        }
}
}

//показываем работу программы
public class Main {

```

```

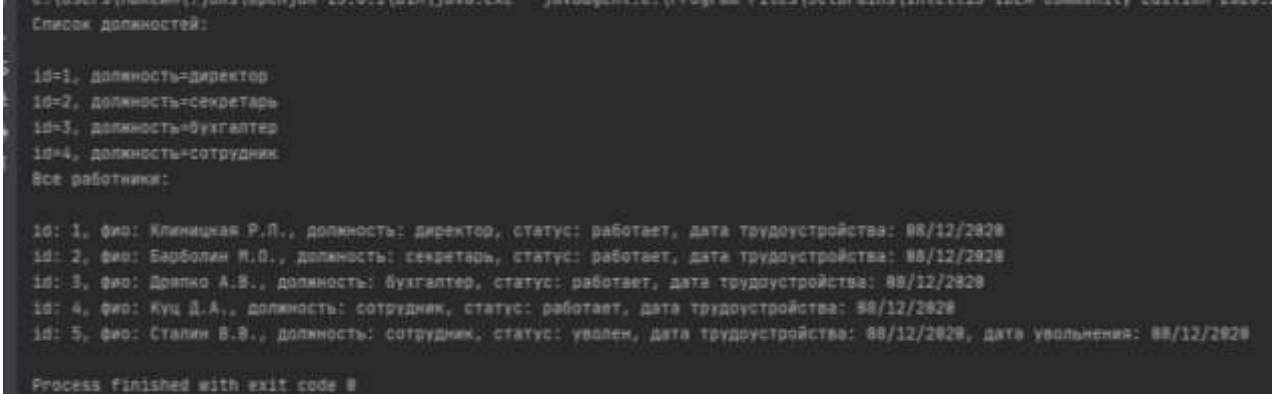
public static void main(String[] args) {
    Department department = new Department();

    System.out.println("Список должностей:\n");
    department.showRoles();
    //добавляем рабочих
    department.addWorker(1, "Клиническая Р.П.", 1);
    department.addWorker(2, "Барболин М.О.", 2);
    department.addWorker(3, "Дряпко А.В.", 3);
    department.addWorker(4, "Куц Д.А.", 4);
    department.addWorker(5, "Сталин В.В.", 4);
    //удаляем Сталина
    department.dismissWorker(5);
    //Вывод оставшихся рабочих
    System.out.println("Все работники:\n");

    department.showWorkers();
}
}

```

Скриншоты:



```

Список должностей:
1d=1, должность=директор
1d=2, должность=секретарь
1d=3, должность=бухгалтер
1d=4, должность=сотрудник
Все работники:

1d: 1, ф.ио: Клиническая Р.П., должность: директор, статус: работает, дата трудоустройства: 08/12/2020
1d: 2, ф.ио: Барболин М.О., должность: секретарь, статус: работает, дата трудоустройства: 08/12/2020
1d: 3, ф.ио: Дряпко А.В., должность: бухгалтер, статус: работает, дата трудоустройства: 08/12/2020
1d: 4, ф.ио: Куц Д.А., должность: сотрудник, статус: работает, дата трудоустройства: 08/12/2020
1d: 5, ф.ио: Сталин В.В., должность: сотрудник, статус: уволен, дата трудоустройства: 08/12/2020, дата увольнения: 08/12/2020

Process finished with exit code 0

```

2. Создать класс Абзац, используя класс Слово.

Код программы:

```

package com.company;
import java.util.ArrayList;

//создали класс для слова
//слово можно задавать при инициализации класса или задавать
//так же выводить на печать.
class Word {

    String word;

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

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

    public void print(){
        System.out.print(word);
    }

    //т.к. класс - слово, то заданная строка обязана содержать неразрывные

```

```

буквы
    public boolean isCorrectWord() {
        //если больше одного слова (пошли на улицу), то мы определяем это по
пробелам
        String[] temp = word.split(" ");
        if(temp.length==1)
            return true;
        else
            return false;
    }
}
//создаем класс Абзаца
class Paragraph {
    //создаем список абзаца из слов
    ArrayList<Word> paragraph = new ArrayList<Word>();
    //отступ
    int margin = 5;

    public void setMargin(int margin){
        this.margin = margin;
    }
    //добавить слово в абзац
    public boolean addWord(String word){
        Word temp = new Word(word);
        if(!temp.isCorrectWord())
            return false;

        paragraph.add(temp);
        return true;
    }
    //вывести абзац на печать
    public void print(){
        if(paragraph.size()==0)
            System.out.println("Пустой абзац");
        else
        {
            for(int i=0; i<margin; i++)
                System.out.print(" ");
            for(int i = 0; i< paragraph.size(); i++) {
                paragraph.get(i).print();
                System.out.print(" ");
            }
            System.out.println();
        }
    }
}

public class Main {

    public static void main(String[] args) {
        Paragraph paragraph = new Paragraph();
        paragraph.addWord("This");
        paragraph.addWord("is");
        paragraph.addWord("test");
        paragraph.addWord("start");
        paragraph.addWord("aaa aaa aaa");
        paragraph.addWord("!");
        paragraph.print();
    }
}

```

Скриншоты:

```
This is test start !  
Process finished with exit code 0
```

3. Система Библиотека. Читатель оформляет Заказ на Книгу. Система осуществляет поиск в Каталоге. Библиотекарь выдает Читателю Книгу на абонемент или в читальный зал. При невозвращении Книги Читателем он может быть занесен Администратором в «черный список».

Код программы:

```
package com.company;  
import java.util.HashMap;  
import java.util.ArrayList;  
  
//книга  
class Book {  
  
    //id книги, название и автор. их инициализация и получение:  
    private int id;  
    private String title;  
    private String author;  
  
    public Book(int id, String title, String author) {  
        this.id = id;  
        this.title = title;  
        this.author = author;  
    }  
  
    public int getId() {  
        return id;  
    }  
  
    public void setId(int id) {  
        this.id = id;  
    }  
  
    public String getTitle() {  
        return title;  
    }  
  
    public void setTitle(String title) {  
        this.title = title;  
    }  
  
    public String getAuthor() {  
        return author;  
    }  
  
    public void setAuthor(String author) {  
        this.author = author;  
    }  
  
    //метод проверяющий книгу на совпадение (по идентификатору)  
    @Override  
    public boolean equals(Object obj) {  
        return ((Book)obj).id == id;  
    }  
}
```

```

        public void print(){
            System.out.print("id: "+id+"; title: "+ title +"; author: "+author);
        }
    }
    //каталог книг
    class Catalog {
        //создаем список из книг, методы на добавление, удаление книг из списка
        private ArrayList<Book> books = new ArrayList<Book>();

        public void addBook(Book book) {
            books.add(book);
        }
        //Удаление и поиск книги происходят по идентификатору
        public void deleteBookById(int bookId) {
            for(int i=0; i<books.size(); i++)
                if(books.get(i).getId() == bookId) {
                    books.remove(i);
                    return;
                }
        }

        public Book getBookById(int bookId) {
            for(int i=0; i<books.size(); i++)
                if(books.get(i).getId() == bookId)
                    return books.get(i);

            return null;
        }

        public void show() {
            System.out.println("Каталог:");
            for(int i=0; i<books.size(); i++) {
                books.get(i).print();
                System.out.println();
            }
        }
    }
}

//человек. может как быть читателем, так и находиться в черной книге, быть
администратором и т.д.
//класс содержит идентификатор и ФИО человека
abstract class Person {

    protected int id;
    protected String fio;

    public Person(int id, String fio) {
        this.id = id;
        this.fio = fio;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getFio() {
        return fio;
    }
}

```

```

        public void setFio(String fio) {
            this.fio = fio;
        }
    }

    //читатель
    class Reader extends Person {
        //может взять на чтение несколько книг, в зависимости от заказа.
        //изначально НЕ находится в "черном" списке
        private HashMap<Book, Integer> books = new HashMap<Book, Integer>();
        private boolean blacklist = false;

        public Reader(int id, String fio) {
            super(id, fio);
        }

        //методы на взятие или возврат книги, добавление и проверку "черного"
        //списка
        public void takeBook(Book book, int status){
            books.put(book, status);
        }

        public void returnBook(Book book){
            books.remove(book);
        }

        public boolean isBlacklist() {
            return blacklist;
        }

        public void setBlacklist(boolean blacklist) {
            this.blacklist = blacklist;
        }

        //читатель делает заказ на взятие новой кники
        public Order makeOrder(int id, int idBook, int isHaveBook){
            return new Order(id, idBook, isHaveBook);
        }

        public void show(){
            System.out.println("id: "+id+"; fio: "+fio+"; in blacklist:
            "+blacklist);
            System.out.println("Взятые книги:");
            for (HashMap.Entry<Book, Integer> whereIs : books.entrySet()) {
                whereIs.getKey().print();
                System.out.print("; тип выдачи: ");
                if (whereIs.getValue() == 0)
                    System.out.println("абонемент");
                else if (whereIs.getValue() == 1)
                    System.out.println("читальный зал");
            }
        }
    }

    //заказ на получение книги
    class Order {
        //номер заказа, книги, информация по количеству книг
        private int id;
        private int idBook;
        private int isHaveBook;

        //инициализация и получение информации
    }

```



```

    public Order(int id, int idBook, int isHaveBook) {
        this.id = id;
        this.idBook = idBook;
        this.isHaveBook = isHaveBook;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public int getIdBook() {
        return idBook;
    }

    public void setIdBook(int idBook) {
        this.idBook = idBook;
    }

    public int getIsHaveBook() {
        return isHaveBook;
    }

    public void setIsHaveBook(int isHaveBook) {
        this.isHaveBook = isHaveBook;
    }
}

//интерфейс! библиотекаря
interface LibrarianInterface {
    boolean giveBookToReader(Reader reader, Order order);
}

//сам класс библиотекаря
class Librarian extends Person implements LibrarianInterface {

    //инициализация и получение
    private Catalog catalog;

    public Librarian(int id, String fio, Catalog catalog) {
        super(id, fio);
        this.catalog = catalog;
    }

    public Catalog getCatalog() {
        return catalog;
    }

    public void setCatalog(Catalog catalog) {
        this.catalog = catalog;
    }

    //проверяем, не находится ли читатель в черном списке и выдаем ему книгу
    public boolean giveBookToReader(Reader reader, Order order) {
        if(reader.isBlacklist())
            return false;

        Book book = catalog.getBookById(order.getIdBook());

        if(book==null)
            return false;
    }
}

```

```

        reader.takeBook(book, order.getIsHaveBook());
        catalog.deleteBookById(book.getId());
        return true;
    }

    public void takeBookFromReader(Reader reader, Book book) {
        reader.returnBook(book);
        catalog.addBook(book);
    }
}
//администратор заносащий в "черный" список
class Admin extends Person {

    public Admin(int id, String fio) {
        super(id, fio);
    }
    //смысл жизни - добавлять в черный список.
    public void addToBlacklist(Reader reader){
        reader.setBlacklist(true);
    }
}
//демонстрация работы программы
public class Main {

    public static void main(String[] args) {
        //устанавливаем администратор, создаем каталог и добавляем в него
        //несколько книг
        Admin administrator = new Admin(0, "Барболин М.О.");

        Catalog catalog = new Catalog();

        Book book1 = new Book(5, "Сказка о веселых котят", "Достоевский");
        Book book2 = new Book(10, "Невская ночь", "Трибоедов");
        Book book3 = new Book(12, "Чистый код", "Молодец");
        catalog.addBook(book1);
        catalog.addBook(book2);
        catalog.addBook(book3);

        //после заполнения каталога книгами создаем объект библиотекаря и
        //сообщаем ему каталог
        Librarian librarian = new Librarian(1, "Любимин Л.П.", catalog);
        catalog.show();

        //создаем читателя и генерируем подачу запросов
        Reader reader1 = new Reader(987, "Вереметьев И.И.");
        Reader reader2 = new Reader(584, "Горшков А.М.");

        Order order1 = reader1.makeOrder(34, 10, 0);
        Order order2 = reader1.makeOrder(11, 12, 1);
        Order order3 = reader2.makeOrder(15, 5, 0);

        System.out.println("\n\n\n");
        //проверяем запрошенные книги и выдаем по запросу, если совпали
        if(!librarian.giveBookToReader(reader1, order1))
            System.out.println("order1 не удалось выдать reader1");

        if(!librarian.giveBookToReader(reader1, order2))
            System.out.println("order2 не удалось выдать reader1");

        if(!librarian.giveBookToReader(reader2, order3))
            System.out.println("order3 не удалось выдать reader2");
        //приемка книг (возврат)
        librarian.takeBookFromReader(reader1, book2);
        //обавление читателя в черный список
    }
}

```

```

        administrator.addToBlacklist(reader2);

        Order order4 = reader2.makeOrder(357, 10, 1);

        if(!librarian.giveBookToReader(reader2, order4))
            System.out.println("order4 не удалось выдать reader2");

        System.out.println("\n\n\n");

        catalog.show();

        System.out.println("\n\n\nReader1:");

        reader1.show();

        System.out.println("\n\n\nReader2:");

        reader2.show();
    }
}

```

Скриншоты:

```

Каталог:
id: 5; title: Сказка о веселых котятках; author: Достоевский
id: 10; title: Невская ночь; author: Грибоедов
id: 12; title: Чистый код; author: Молодец

order4 не удалось выдать reader2

Каталог:
id: 10; title: Невская ночь; author: Грибоедов

Reader1:
id: 987; fio: Вереметьев И.И; in blacklist: false
Взятые книги:
id: 12; title: Чистый код; author: Молодец; тип выдачи: читальный зал

Reader2:
id: 584; fio: Горшков А.М.; in blacklist: true
Взятые книги:
id: 5; title: Сказка о веселых котятках; author: Достоевский; тип выдачи: абонемент

Process finished with exit code 0

```

Вывод: в ходе лабораторной работы ознакомилась с ООП в java, приобрела практические навыки использования ООП в java, выполнила поставленные задачи.