

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

ОТЧЕТ  
По лабораторной работе №4

Выполнил:  
Студент 4 курса  
группы АС-50  
Ольховик И.Ю.  
Проверил:  
Крощенко А.А.

Брест 2020

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

## Задание 1

Создать класс City (город) с внутренним классом, с помощью объектов которого можно хранить информацию о проспектах, улицах, площадях.

Код программы:

```
package com.company;

import java.util.ArrayList;

public class Main {

    public static void main(String[] args) {
        String name = "Брест";
        ArrayList<String> streets = new ArrayList<>();
        streets.add("Улица В.И. Ленина");
        streets.add("Улица Московская");
        ArrayList<String> squares = new ArrayList<>();
        squares.add("Площадь Ленина");
        ArrayList<String> avenues = new ArrayList<>();
        avenues.add("Проспект Машерова");
        avenues.add("Партизанский проспект");
        City.Communicate communicate = new City.Communicate();
        communicate.setStreets(streets);
        communicate.setAvenues(avenues);
        communicate.setSquares(squares);
        communicate.addStreet("Улица Янки Купалы");
        communicate.addAvenue("Проспект Республики");
        communicate.addSquare("Площадь Свободы");

        City city = new City(name, communicate);
        city.printInfo();
    }
}
```

```
package com.company;

import java.util.ArrayList;

public class City {
    String name;
    Communicate communicate = new Communicate();
    static class Communicate {
        ArrayList<String> squares;
        ArrayList<String> avenues;
        ArrayList<String> streets;

        public ArrayList<String> getSquares() {
            return squares;
        }

        public void addSquare(String value) {
            squares.add(value);
        }

        public void setSquares(ArrayList<String> squares) {
```

```

        this.squares = squares;
    }

    public ArrayList<String> getAvenues() {
        return avenues;
    }

    public void addAvenue(String value) {
        avenues.add(value);
    }

    public void setAvenues(ArrayList<String> avenues) {
        this.avenuess = avenues;
    }

    public ArrayList<String> getStreets() {
        return streets;
    }

    public void addStreet(String value) {
        streets.add(value);
    }

    public void setStreets(ArrayList<String> streets) {
        this.streets = streets;
    }
}

public City(String name, Communicate communicate) {
    this.name = name;
    this.communicate = communicate;
}

public void printinfo() {
    System.out.println("Город: " + name);
    System.out.println("В нём есть такие площади как:" +
communicate.getSquares());
    System.out.println("В нём есть такие улицы как:" +
communicate.getStreets());
    System.out.println("В нём есть такие проспекты как:" +
communicate.getAvenues());
}
}

```

## Скриншоты выполнения программы:

```

Город: Брест
В нём есть такие площади как:[Площадь Ленина, Площадь Свободы]
В нём есть такие улицы как:[Улица В.И. Ленина, Улица Московская, Улица Янки Купалы]
В нём есть такие проспекты как:[Проспект Машерова, Партизанский проспект, Проспект Республики]

```

## Задание 2

Создать класс Страница, используя классы Строка, Слово.

Код программы:

```
package com.company;
```

```

public class Main {

    public static void main(String[] args) {
        Word word = new Word("Hello");
        Word word1 = new Word("world");
        Word word2 = new Word("it`s");
        Word word3 = new Word("test");

        Line line = new Line();
        line.addValue(word);
        line.addValue(word1);
        Line line1 = new Line();
        line1.addValue(word2);
        line1.addValue(word3);

        Page page = new Page();
        page.addBody(line);
        page.addBody(line1);
        page.addBody(word1);
        page.addBodyNL(word3);

        System.out.println("Our page: " + page.getBody());
    }
}

```

```

package com.company;

public class Word {
    private String value;
    public Word(String value) {
        this.value = value;
    }

    public String getValue() {
        return value;
    }
}

```

```

package com.company;

public class Page {
    private String body = "";

    public void addBody(Word word) {
        body += " " + word.getValue();
    }

    public void addBodyNL(Word word) {
        body += "\n " + word.getValue();
    }

    public void addBody(Line line) {
        body += "\n" + line.getValue();
    }

    public String getBody() {
        return body;
    }
}

```

```

package com.company;

public class Line {
    private String value="";

    public void addValue(Word word) {
        value += " " + word.getValue();
    }

    public String getValue() {
        return value;
    }
}

```

### Скриншоты выполнения работы:

```

Our page:
Hello world
it's test world
test

```

### Задание 3

Система Автобаза. Диспетчер распределяет заявки на Рейсы между Водителями и назначает для этого Автомобиль. Водитель может сделать заявку на ремонт. Диспетчер может отстранить Водителя от работы. Водитель делает отметку о выполнении Рейса и состоянии Автомобиля.

Код программы:

```

package com.company;

import java.util.ArrayList;

public class Main {

    public static void main(String[] args) {
        ArrayList<Request> requestsList = new ArrayList<>();
        requestsList.add(new Request("Машерова 65", "Московская 267"));
        requestsList.add(new Request("Машерова 10", "Московская 265"));
        requestsList.add(new Request("Машерова 6", "Московская 263"));
        ArrayList<Driver> driverList = new ArrayList<>();
        driverList.add(new Driver("Водитель1"));
        driverList.add(new Driver("Водитель2"));
        driverList.add(new Driver("Водитель3"));
        driverList.add(new Driver("Водитель4"));
        driverList.add(new Driver("Водитель5"));
        ArrayList<Car> carsList = new ArrayList<>();
        carsList.add(new Car("Volvo", "2625ki4", "Полностью исправно"));
        carsList.add(new Car("Mercedes", "2655ki4", "Полностью исправно"));
        carsList.add(new Car("Vw", "2225ki4", "Полностью исправно"));
        carsList.add(new Car("Volvo", "2125ki4", "Требуется ремонт"));
        ArrayList<Dispatcher> dispatchersList = new ArrayList<>();
        Dispatcher mainDispatcher = new Dispatcher("Главный диспетчер",
driverList);
        dispatchersList.add(0,mainDispatcher);
    }
}

```

```

        Department department = new
        Department(driverList, carsList, dispatchersList);
        mainDispatcher.sentRequest(driverList.get(0),
        requestsList.get(0), carsList.get(0));
        mainDispatcher.sentRequest(driverList.get(1),
        requestsList.get(1), carsList.get(1));
        mainDispatcher.sentRequest(driverList.get(2),
        requestsList.get(2), carsList.get(2));
        String cause = driverList.get(3).requestToRepair();
        mainDispatcher.sentRepairRequest(driverList.get(3),
        cause, carsList.get(3));
        mainDispatcher.comleteRequest(driverList.get(0), "Авто полностью
        исправно", carsList.get(0));
        mainDispatcher.suspendedDriver(driverList.get(1), requestsList.get(1),
        driverList.get(4), carsList.get(1));
        mainDispatcher.rejectedRequest(driverList.get(2), "Авто полностью
        исправно", carsList.get(2));
        mainDispatcher.comleteRequest(driverList.get(3), "Авто полностью
        исправно", carsList.get(3));
        mainDispatcher.comleteRequest(driverList.get(4), "Авто полностью
        исправно", carsList.get(1));
    }
}

```

```

package com.company;

import java.util.ArrayList;

class Driver {
    String name;
    Car car;
    ArrayList<Request> requests = new ArrayList<>();

    public Driver(String name) {
        this.name = name;
    }

    public Driver(String name, ArrayList<Request> requests, Car car) {
        this.name = name;
        this.requests = requests;
        this.car = car;
    }

    public void addRequest(Request request, Car car) {
        requests.add(request);
        if (request.startPoint.toLowerCase().equals("ремонт"))
            System.out.println("Авто " + car.model + " " + car.number + "
        отправлено на " + request.startPoint + ", по причине: " + request.endPoint);
        else
            System.out.println("Водителю " + name + " назначен заказ: " +
        request.startPoint + " - " + request.endPoint);
    }

    public void removeRequest(Request request, Car car) {
        requests.remove(request);
        System.out.println("Авто " + car.model + " " + car.number + "
        возвращено на базу, состояние: " + car.state);
    }

    public String requestToRepair() {
        String cause = "Проблемы с двигателем";
        System.out.println("Водитель " + name + " создал заявку на ремонт
        авто, по причине: " + cause);
    }
}

```

```

        return cause;
    }

    public void completeRequest(Request request, String state, Car car) {
        request.completeOrder();
        if (request.startPoint.toLowerCase().equals("ремонт")) {
            System.out.println("Состояние авто " + car.model + " " +
car.number + " y " + name + " после завершения ремонта: " + state);
            car.setState(state);
        } else {
            System.out.println("Состояние авто " + car.model + " " +
car.number + " y " + name + " после выполнения заказа: " + state);
            car.setState(state);
        }
    }

    public void rejectedRequest(Request request, String state, Car car) {
        request.rejectedOrder();
        System.out.println("Состояние авто " + car.model + " " +
car.number + " y " + name + " после возвращения на базу из-за отмены заказа: "
+ state);
        car.setState(state);
    }
}

class Dispatcher {
    String name;
    ArrayList<Driver> drivers;

    public Dispatcher(String name, ArrayList<Driver> driversList) {
        this.name = name;
        this.drivers = driversList;
    }

    public void sentRequest (Driver driver, Request request, Car car) {
        drivers.get(drivers.indexOf(driver)).addRequest(request, car);
    }

    public void sentRepairRequest(Driver driver, String cause, Car car) {
        drivers.get(drivers.indexOf(driver)).addRequest(new
Request("Ремонт", cause), car);
    }

    public void comleteRequest(Driver driver, String state, Car car) {
drivers.get(drivers.indexOf(driver)).completeRequest(driver.requests.get(0), s
tate, car);
    }

    public void rejectedRequest(Driver driver, String state, Car car) {
drivers.get(drivers.indexOf(driver)).rejectedRequest(driver.requests.get(0), s
tate, car);
    }

    public void suspendedDriver(Driver driver, Request request, Driver
driver1, Car car) {
        System.out.println(driver.name + " отстранён, его заказ переходит
" + driver1.name);
        driver.removeRequest(request, car);
        driver1.addRequest(request, car);
    }
}

```

```

package com.company;

import java.util.ArrayList;

class Department {
    public ArrayList<Driver> drivers;
    public ArrayList<Car> cars;
    public ArrayList<Dispatcher> dispatchers;

    public Department(ArrayList<Driver> drivers, ArrayList<Car> cars,
ArrayList<Dispatcher> dispatchers) {
        this.drivers = drivers;
        this.cars = cars;
        this.dispatchers = dispatchers;
    }

    public void addCar(Car car) {
        cars.add(car);
    }

    public void removeCar(Car car) {
        cars.remove(car);
    }

    public void addDispatcher(Dispatcher dispatcher) {
        dispatchers.add(dispatcher);
    }

    public void removeDispatcher(Dispatcher dispatcher) {
        dispatchers.remove(dispatcher);
    }

    public void addDriver(Driver driver) {
        drivers.add(driver);
    }

    public void removeDriver(Driver driver) {
        drivers.remove(driver);
    }
}

class Request {
    public String startPoint;
    public String endPoint;

    public enum Status {During, Finished, Rejected}

    public Status CurrentStatus;

    public Request(String startPoint, String endPoint) {
        this.startPoint = startPoint;
        this.endPoint = endPoint;
        CurrentStatus = Status.During;
    }

    public void completeOrder() {
        CurrentStatus = Status.Finished;
        if (startPoint.toLowerCase().equals("ремонт"))
            System.out.println(startPoint + " - " + endPoint + " завершён");
        else
            System.out.println("Заказ: " + startPoint + " - " + endPoint + "
Выполнен");
    }
}

```



```

        public void rejectedOrder() {
            CurrentStatus = Status.Rejected;
            System.out.println("Заказ: " + startPoint + " - " + endPoint + "
отменён");
        }

        public Status getCurrentStatus() {
            return CurrentStatus;
        }

        public void setCurrentStatus(Status currentStatus) {
            CurrentStatus = currentStatus;
        }
    }

    class Car {
        String model;
        String number;
        String state;

        public Car(String model, String number, String state) {
            this.model = model;
            this.number = number;
            this.state = state;
        }

        public String getModel() {
            return model;
        }

        public void setModel(String model) {
            this.model = model;
        }

        public String getNumber() {
            return number;
        }

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

        public String getState() {
            return state;
        }

        public void setState(String state) {
            this.state = state;
        }
    }
}

```

**Скриншоты выполнения программы:**

Водителю Водитель1 назначен заказ: Машерова 65 - Московская 267  
Водителю Водитель2 назначен заказ: Машерова 10 - Московская 265  
Водителю Водитель3 назначен заказ: Машерова 6 - Московская 263  
Водитель Водитель4 создал заявку на ремонт авто, по причине: Проблемы с двигателем  
Авто Volvo 2125ki4 отправлено на Ремонт, по причине: Проблемы с двигателем  
Заказ: Машерова 65 - Московская 267 Выполнен  
Состояние авто Volvo 2625ki4 у Водитель1 после выполнения заказа: Авто полностью исправно  
Водитель2 отстранён, его заказ переходит Водитель5  
Авто Mercedes 2655ki4 возвращено на базу, состояние: Полностью исправно  
Водителю Водитель5 назначен заказ: Машерова 10 - Московская 265  
Заказ: Машерова 6 - Московская 263 отменён  
Состояние авто Vw 2225ki4 у Водитель3 после возвращения на базу из-за отмены заказа: Авто полностью исправно  
Ремонт - Проблемы с двигателем завершён  
Состояние авто Volvo 2125ki4 у Водитель4 после завершения ремонта: Авто полностью исправно  
Заказ: Машерова 10 - Московская 265 Выполнен  
Состояние авто Mercedes 2655ki4 у Водитель5 после выполнения заказа: Авто полностью исправно

**Вывод:** приобрели практические навыки в области объектно-ориентированного проектирования.