МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”
КАФЕДРА ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Отчёт
по лабораторной работе №5

Выполнил:
студент группы ПО-9
Ступак Д.Р

Проверил:
Крощенко А. А.

Брест 2024

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

Вариант 7

Задание 1

Реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов

interface Корабль ← abstract class Военный Корабль ← class Авианосец

Код программы

```
interface Ship {
    void Move();
}

// Абстрактный класс Военный Корабль, реализующий интерфейс Корабль
abstract class Battleship implements Ship {
    protected String Name;
    protected int Length;

    public Battleship(String Name, int Length) {
        this.Name = Name;
        this.Length = Length;
    }

    public abstract void fight();
}

// Класс Авианосец, наследующийся от Военного Корабля
class AircraftCarrier extends Battleship {
    private final int jetCount;

    public AircraftCarrier(String Name, int Length, int jetCount) {
        super(Name, Length);
        this.jetCount = jetCount;
    }

    @Override
    public void Move() {
        System.out.println("Авианосец " + Name + " плывет.");
    }

    @Override
    public void fight() {
        System.out.println("Авианосец " + Name + " ведет боевые действия с помощью " + jetCount
+ " самолетов.");
    }
}

// Пример использования классов
public class Lab5_1 {
    public static void main(String[] args) {
        AircraftCarrier airship = new AircraftCarrier("Адмирал Кузнецов", 1000, 50);
        airship.Move();
        airship.fight();
    }
}
```

```
Авианосец Адмирал Кузнецов плывет.
Авианосец Адмирал Кузнецов ведет боевые действия с помощью 50 самолетов.

Process finished with exit code 0
```

Задание 2

В следующих заданиях требуется создать суперкласс (абстрактный класс, интерфейс) и определить общие методы для данного класса. Создать подклассы, в которых добавить специфические свойства

и методы. Часть методов переопределить. Создать массив объектов суперкласса и заполнить объектами подклассов. Объекты подклассов идентифицировать конструктором по имени или идентификационному номеру. Использовать объекты подклассов для моделирования реальных ситуаций и объектов.

Создать базовый класс Садовое дерево и производные классы Яблоня, Вишня, Груша и другие. С помощью конструктора автоматически становить номер каждого дерева. Принять решение о пересадке каждого дерева в зависимости от возраста и плодоношения.

Код программы

```
abstract class Tree {
    private static int nextId = 1;
    private int id;
    private int age;

    boolean isFruiting;

    public Tree (){
        this.id = nextId++;
        this.age = 0;
    }

    public void setFruiting(boolean isFruiting) {
        this.isFruiting = isFruiting;
    }
    public void setAge(int age) {
        this.age = age;
    }
    public int getAge(){
        return age;
    }

    public abstract void grow();
    public int getId(){
        return id;
    }

    public String toString() {
        return "Садовое дерево #" + id + " (" + getAge() + " лет, " + (isFruiting ?
"плодоносит" : "не плодоносит") + ")";
    }
    public abstract boolean replace();
}

class AppleTree extends Tree{
    public AppleTree(){
        super();
    }

    @Override
    public void grow() {
        setAge(getAge() + 1);
        if (getAge() >= 2) {
            setFruiting(true);
        }
    }
    @Override
    public boolean replace() {
        return getAge() >=3;
    }
    @Override
    public String toString() {
        return super.toString() + ":Яблоня";
    }
}

class CherryTree extends Tree{
    public CherryTree(){
        super();
```

```
    }

    @Override
    public void grow() {
        setAge(getAge() + 1);
        if (getAge() >= 3) {
            setFruiting(true);
        }
    }
    @Override
    public boolean replace() {
        return getAge() >=5;
    }
    @Override
    public String toString() {
        return super.toString() + ":Вишня";
    }
}

class PearTree extends Tree{
    public PearTree(){
        super();
    }
    @Override
    public void grow() {
        setAge(getAge() + 1);
        if (getAge() >= 4) {
            setFruiting(true);
        }
    }
    @Override
    public boolean replace() {
        return getAge() >=7;
    }
    @Override
    public String toString() {
        return super.toString() + ":Груша";
    }
}


public class Lab5_2 {
    public static void main(String[] args) {
        Tree[] trees = new Tree[3];
        trees[0] = new AppleTree();
        trees[1] = new CherryTree();
        trees[2] = new PearTree();

        for (int i = 1; i < 5; i++) {
            for (Tree tree : trees) {
                tree.grow();
                System.out.println(tree);
            }
        }

        System.out.println("Деревья, которые нужно пересадить:");
        for (Tree tree : trees) {
            if (tree.replace() && !tree.isFruiting) {
                System.out.println(tree);
            }
        }

    }
}
```

Вывод

```
Садовое дерево #1 (1 лет, не плодоносит):Яблоня
Садовое дерево #2 (1 лет, не плодоносит):Вишня
Садовое дерево #3 (1 лет, не плодоносит):Груша
Садовое дерево #1 (2 лет, плодоносит):Яблоня
Садовое дерево #2 (2 лет, не плодоносит):Вишня
Садовое дерево #3 (2 лет, не плодоносит):Груша
Садовое дерево #1 (3 лет, плодоносит):Яблоня
Садовое дерево #2 (3 лет, плодоносит):Вишня
Садовое дерево #3 (3 лет, не плодоносит):Груша
Садовое дерево #1 (4 лет, плодоносит):Яблоня
Садовое дерево #2 (4 лет, плодоносит):Вишня
Садовое дерево #3 (4 лет, плодоносит):Груша
Садовое дерево #1 (5 лет, плодоносит):Яблоня
Садовое дерево #2 (5 лет, плодоносит):Вишня
Садовое дерево #3 (5 лет, плодоносит):Груша
Деревья, которые нужно пересадить:
Садовое дерево #1 (5 лет, плодоносит):Яблоня
Садовое дерево #2 (5 лет, плодоносит):Вишня

Process finished with exit code 0
```

Задание 3

В задании 3 ЛР No4, где возможно, заменить объявления суперклассов объявлениями абстрактных классов или интерфейсов.

Код программы

```
import java.util.ArrayList;
import java.util.List;

public class Lab5_3 {

    public class Trip {

        private String destination;
        private Car car;
        private Driver driver;
        private boolean isEnd;

        public Trip(String destination, Car car, Driver driver) {
            this.destination = destination;
            this.car = car;
            this.driver = driver;
            this.isEnd = false;
        }
        public boolean getStatus() {
            return isEnd;
        }

        public void setStatus(boolean isEnd) {
            this.isEnd = isEnd;
        }
        public String getDestination() {
            return destination;
        }


        public Car getCar() {
            return car;
```

```
    }

    public Driver getDriver() {
        return driver;
    }

    @Override
    public String toString() {
        return "Рейс: " + destination + " (" + car + ", " + driver + ")";
    }

    public void setCar(Car car) {
        this.car = car;
    }

    public void setDriver(Driver driver) {
        this.driver = driver;
    }
}
public class Driver {

    private String name;
    private boolean available;

    private Trip trip;
    public Driver(String name) {
        this.name = name;
        this.available = true;
    }
    public void sendCarToRepair(){
        if (trip !=null){
            trip.getCar().setAvailable(false);
        }
    }

    public void setTrip(Trip trip){
        this.trip=trip;
    }
    public void markTripCompleted() {
        if (trip !=null){
            trip.setStatus(true);
            this.available=true;
            trip.getCar().setAvailable(true);
            return;
        }
        System.out.println("У водителя нет рейса");
    }
    public String getName() {
        return name;
    }

    public boolean isAvailable() {
        return available;
    }

    public void setAvailable(boolean available) {
        this.available = available;
    }

    @Override
    public String toString() {
        return "Водитель: " + name;
    }
}

public class Car {

    private String model;
    private String licensePlate;
    private boolean available;
```

```
        public Car(String model, String licensePlate) {
            this.model = model;
            this.licensePlate = licensePlate;
            this.available = true;
        }

        public String getModel() {
            return model;
        }

        public String getLicensePlate() {
            return licensePlate;
        }

        public boolean isAvailable() {
            return available;
        }

        public void setAvailable(boolean available) {
            this.available = available;
        }

        @Override
        public String toString() {
            return "Автомобиль: " + model + " (" + licensePlate + ")";
        }
    }
    public class Dispatcher {

        private List<Car> cars;
        private List<Driver> drivers;
        private List<Trip> trips;

        public Dispatcher() {
            this.cars = new ArrayList<>();
            this.drivers = new ArrayList<>();
            this.trips = new ArrayList<>();
        }

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

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

        public Trip assignTrip(String destination) {
            Car car = findAvailableCar();
            Driver driver = findAvailableDriver();
            Trip trip= new Trip(destination, car, driver);
            driver.setTrip(trip);
            trip.setCar(car);
            trip.setDriver(driver);
            car.setAvailable(false);
            driver.setAvailable(false);
            trips.add(trip);
            System.out.println("На рейс назначен: " + driver.toString() + " " +
car.toString());
            return trip;
        }

        private Car findAvailableCar() {
            for (Car car : cars) {
                if (car.isAvailable()) {
                    return car;
                }
            }
            return null;
        }
```

```
        private Driver findAvailableDriver() {
            for (Driver driver : drivers) {
                if (driver.isAvailable()) {
                    return driver;
                }
            }
            return null;
        }

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



        public List<Car> getAvailableCars() {
            List<Car> result = new ArrayList<Car>();
            for (Car car : cars){
                if (car.isAvailable())
                    result.add(car);
            }
            return result;
        }

        public List<Driver> getAvailableDrivers() {
            List<Driver> result = new ArrayList<Driver>();
            for (Driver driver : drivers){
                if (driver.isAvailable())
                    result.add(driver);
            }
            return result;
        }
    }

    public void main(String[] args) {
        Dispatcher dispatcher = new Dispatcher();
        Car car1 = new Car("Lada Granta", "A1234");
        Car car2 = new Car("Toyota Camry", "B5678");
        dispatcher.addCar(car1);
        dispatcher.addCar(car2);

        Driver driver1 = new Driver("Иван");
        Driver driver2 = new Driver("Петр");
        dispatcher.addDriver(driver1);
        dispatcher.addDriver(driver2);

        Trip trip= dispatcher.assignTrip("Владивосток");
        System.out.println(trip);
        dispatcher.removeDriver(driver2);
        driver1.markTripCompleted();
        for (Car car : dispatcher.getAvailableCars()) {
            System.out.println(car);
        }
        for (Driver driver : dispatcher.getAvailableDrivers()) {
            System.out.println(driver);
        }
    }


}
```

```
На рейс назначен: Водитель: Иван Автомобиль: Lada Granta (A1234)
Рейс: Владивосток (Автомобиль: Lada Granta (A1234), Водитель: Иван)
Автомобиль: Lada Granta (A1234)
Автомобиль: Toyota Camry (B5678)
Водитель: Иван
```