Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №5

По дисциплине: «СПП»

Выполнил:

студент 3 курса

группы ПО-3

Горбун В.О.

Проверил:

Крощенко А.А.

2020

**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.

**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 установить номер каждого дерева. Принять решение о пересадке каждого дерева в зависимости от возраста и плодоношения.

**Задание 3:** в задании 3 ЛР №4, где возможно, заменить объявления суперклассов объявлениями абстрактных классов или интерфейсов.

**Ход работы:**

**Задание 1:**

**Текст программы:**

public class Zadanie1 {

public static void main(String[] args) {

AircraftCarrier trAirCar1 = new AircraftCarrier("HMS Argus", 5000, 3);

AircraftCarrier trAirCar2 = new AircraftCarrier("Mistral", 20000, 5);

trAirCar1.show();

trAirCar2.showAll();

}

}

interface Ship {//корабль

void show();

void showAll();

}

abstract class WarShip implements Ship{//военный корабль

private String name;

private int torpedo;

public WarShip(String \_name, int \_torpedo) {

name = \_name;

torpedo = \_torpedo;

}

public String getName() {

return name;

}

public int getTorpedo() {

return torpedo;

}

}

class AircraftCarrier extends WarShip {//авианосец

private int cost;

public AircraftCarrier (String \_name, int \_torpedo, int \_cost) {

super(\_name, \_torpedo);

cost = \_cost;

}

@Override

public void show() {

System.out.println("AircraftCarrier " + super.getName() + " has " + super.getTorpedo() + " torpedo.");

}

@Override

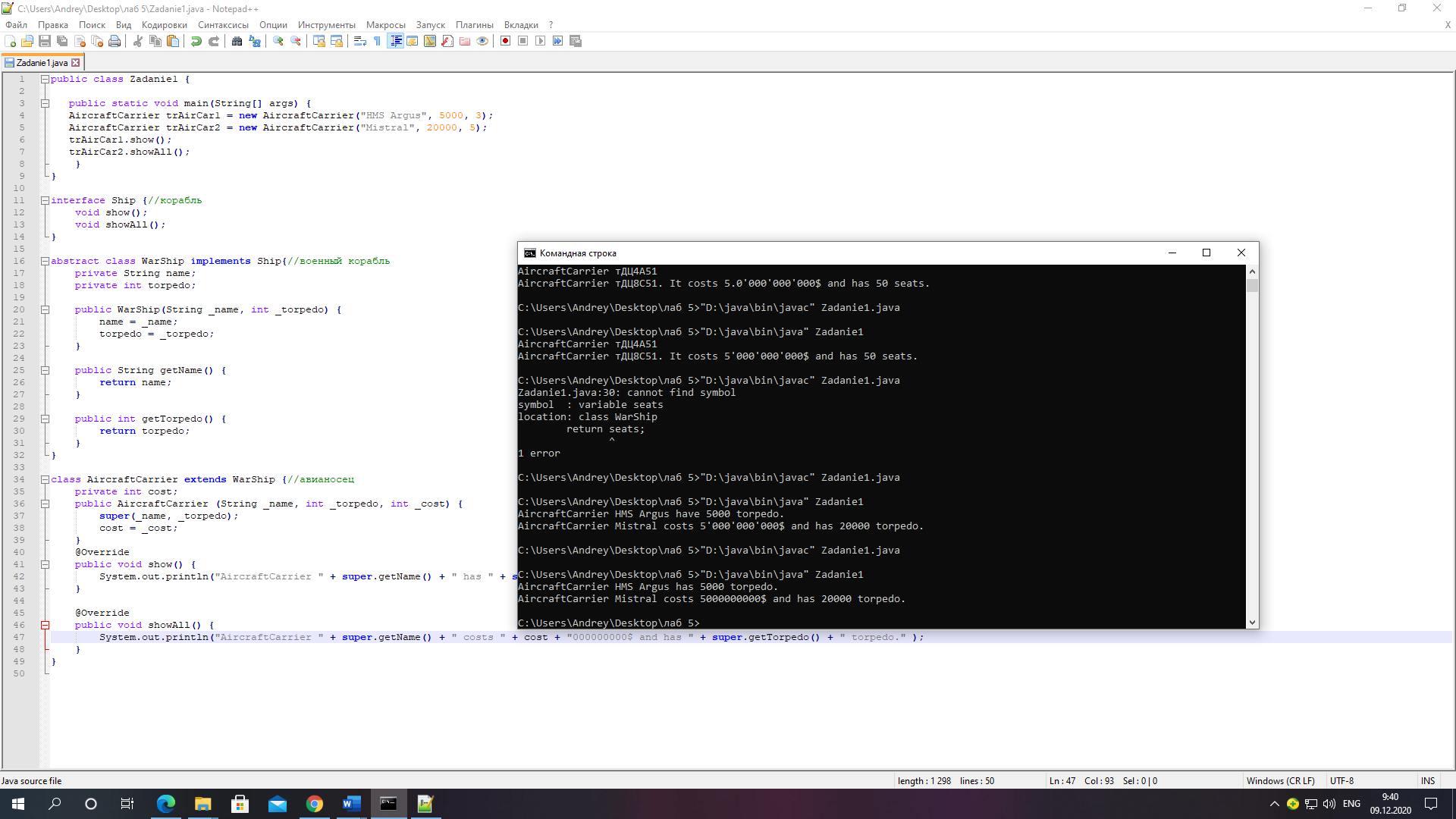
public void showAll() {

System.out.println("AircraftCarrier " + super.getName() + " costs " + cost + "000000000$ and has " + super.getTorpedo() + " torpedo." );

}

}

**Результат:**



**Задание 2:**

**Текст программы:**

public class Zadanie2 {

public static void main(String[] args) {

GardenTree[] trees1 = new GardenTree[3];

trees1[0] = new PearTree(1, 500, "pear tree", 3);

trees1[1] = new AppleTree(2, 600, "apple tree", 7);

trees1[2] = new CherryTree(3, 300, "cherry tree", 4);

for (GardenTree i: trees1)

i.show();

}

}

abstract class GardenTree {

private int number;

private int fruiting;

private String view;

private int age;

public GardenTree (int \_number, int \_fruiting, String \_view, int \_age) {

number = \_number;

fruiting = \_fruiting;

view = \_view;

age = \_age;

}

public int getNumber() {

return number;

}

public int getFruiting() {

return fruiting;

}

public String getView() {

return view;

}

public int getAge() {

return age;

}

public abstract void show();

}

class AppleTree extends GardenTree {

private String fruit = "apples";

public AppleTree (int \_number, int \_fruiting, String \_view, int \_age) {

super(\_number, \_fruiting, \_view, \_age);

}

public void show () {

System.out.print("The number this tree is " + super.getNumber() + ". This is " + super.getView() +

" and it is " + super.getAge() + " years old. It bears fruit up to " + super.getFruiting() + " kilograms " + fruit + " per year. ");

if (super.getAge() < 5) System.out.println("It can be transplanted");

}

}

class PearTree extends GardenTree {

private String fruit = "pears";

public PearTree (int \_number, int \_fruiting, String \_view, int \_age) {

super(\_number, \_fruiting, \_view, \_age);

}

public void show () {

System.out.print("The number this tree is " + super.getNumber() + ". This is " + super.getView() +

" and it is " + super.getAge() + " years old. It bears fruit up to " + super.getFruiting() + " kilograms " + fruit + " per year. ");

if (super.getAge() < 5) System.out.println("It can be transplanted");

}

}

class CherryTree extends GardenTree {

private String fruit = "cherries";

public CherryTree (int \_number, int \_fruiting, String \_view, int \_age) {

super(\_number, \_fruiting, \_view, \_age);

}

public void show () {

System.out.print("The number this tree is " + super.getNumber() + ". This is" + super.getView() +

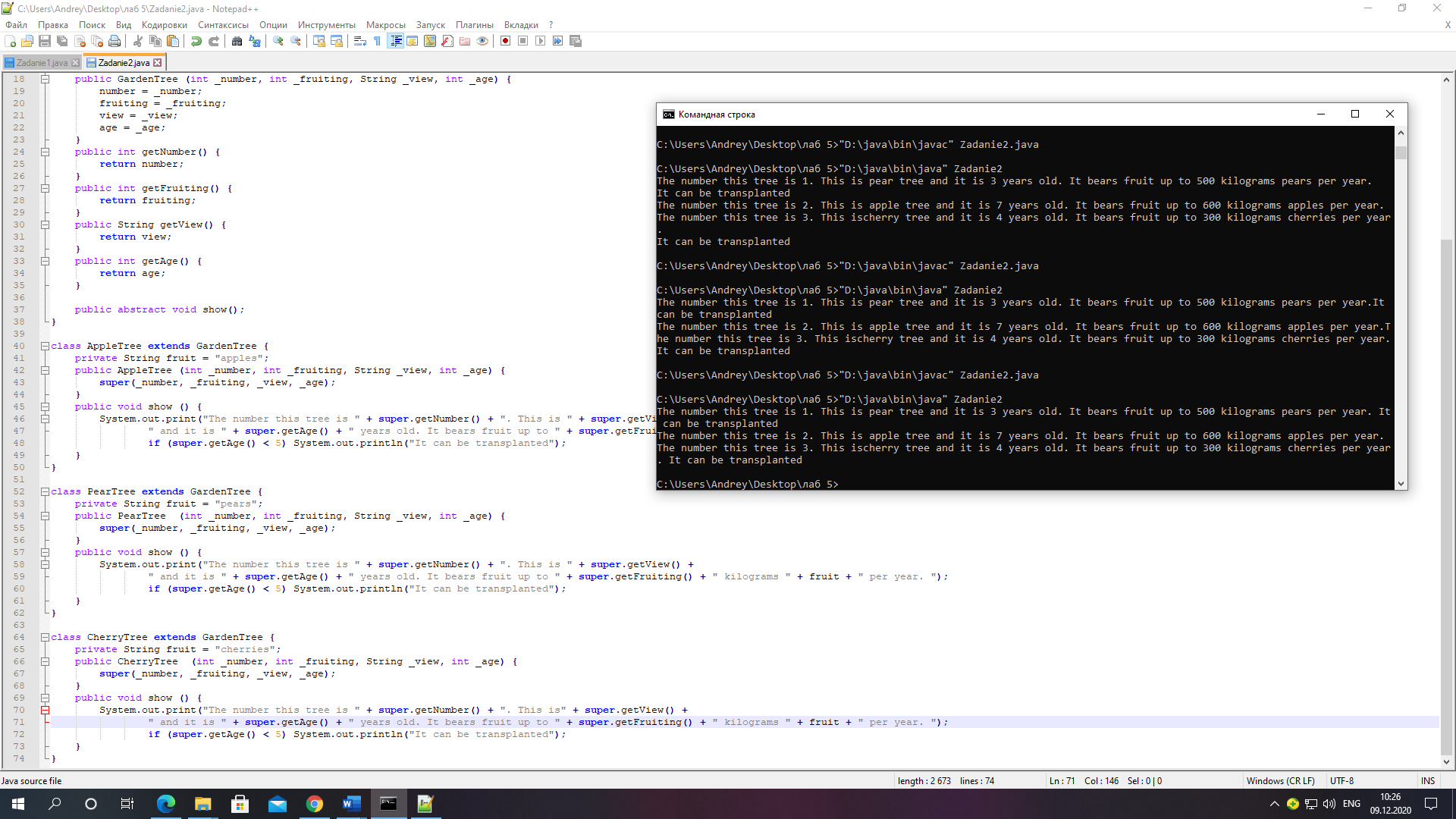
" and it is " + super.getAge() + " years old. It bears fruit up to " + super.getFruiting() + " kilograms " + fruit + " per year. ");

if (super.getAge() < 5) System.out.println("It can be transplanted");

}

}

**Результат:**



**Задание 3:**

**Часть текста программы из задания 3 лабораторной работы 4:**

class Person {

protected String name;

protected String surname;

public Person(String \_name, String \_surname) {

name = \_name;

surname = \_surname;

}

}

class Driver extends Person{

private Race race;

private HashMap<Car.States, Mark> results = new HashMap<Car.States, Mark>();

public Driver (String \_name, String \_surname) {

super(\_name, \_surname);

}

...

class dispatcher extends Person{

private Car.States state;

public dispatcher (String \_name, String \_surname, Car \_car) {

super(\_name, \_surname);

state = \_car.getState();

}

...

**Результаты не изменились.**

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