**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 1

по дисциплине «Технология разработки программного обеспечения»

**Развитие представлений о разработке программ. Объектно-ориентированное программирование в *Java***

Выполнил студент

группы ИТИ-21

Цветков А.А.

Проверил преподаватель

Гуменников Е.Д.

Гомель 2023

**Цель работы:** освоить объектно-ориентированного программирования в *Java.* Развитие представлений о разработке программ.

**Задание**

1. Разработать UML-диаграмму иерархии классов, согласно варианту (таблица 1).

2. При наименовании компонентов руководствоваться соглашением о наименовании (<https://www.oracle.com/technetwork/java/codeconventions-150003.pdf>).

3. При описании иерархии использовать наследование и композицию.

4. На основе *UML*-диаграммы разработать иерархию классов на языке *Java*.

5. Весь код должен быть снабжен элементами документирования (<https://www.jetbrains.com/help/idea/working-with-code-documentation.html>).

6. Разработанную иерархию поместить в .jar файл для дальнейшего использования в качестве библиотечных классов.

7. Создать консольное приложение для демонстрации работы созданных классов.

8. Составить отчет о проделанной работе.

Таблица 1 – Условие задачи

|  |  |
| --- | --- |
| **Вариант** | **Условие задачи** |
| 3 | 1.1 Создать класс *Sneakers* (Кроссовки) с полями *name, cost, type, producer.*  Кроссовки могут быть футбольные, баскетбольные и т.п. (реализовать через наследование)  1.2 Создать приватный внутренний класс *Producer* с полями *name* и *country.*  1.3 Создать не менее 10 объектов, содержащих информацию о различных кроссовках  1.4 Подсчитать количество производителей.  1.5 Подсчитать среднюю стоимость обуви по каждому производителю.  1.6 Подсчитать среднюю стоимость обуви по каждому типу.  1.7 Дополнить иерархию классов новым типом кроссовок, при этом нельзя изменять методы, реализующие пункты 1.4 – 1.6 |

**Ход работы**

На рисунке 1 показана *UML*-диаграмма иерархии классов.

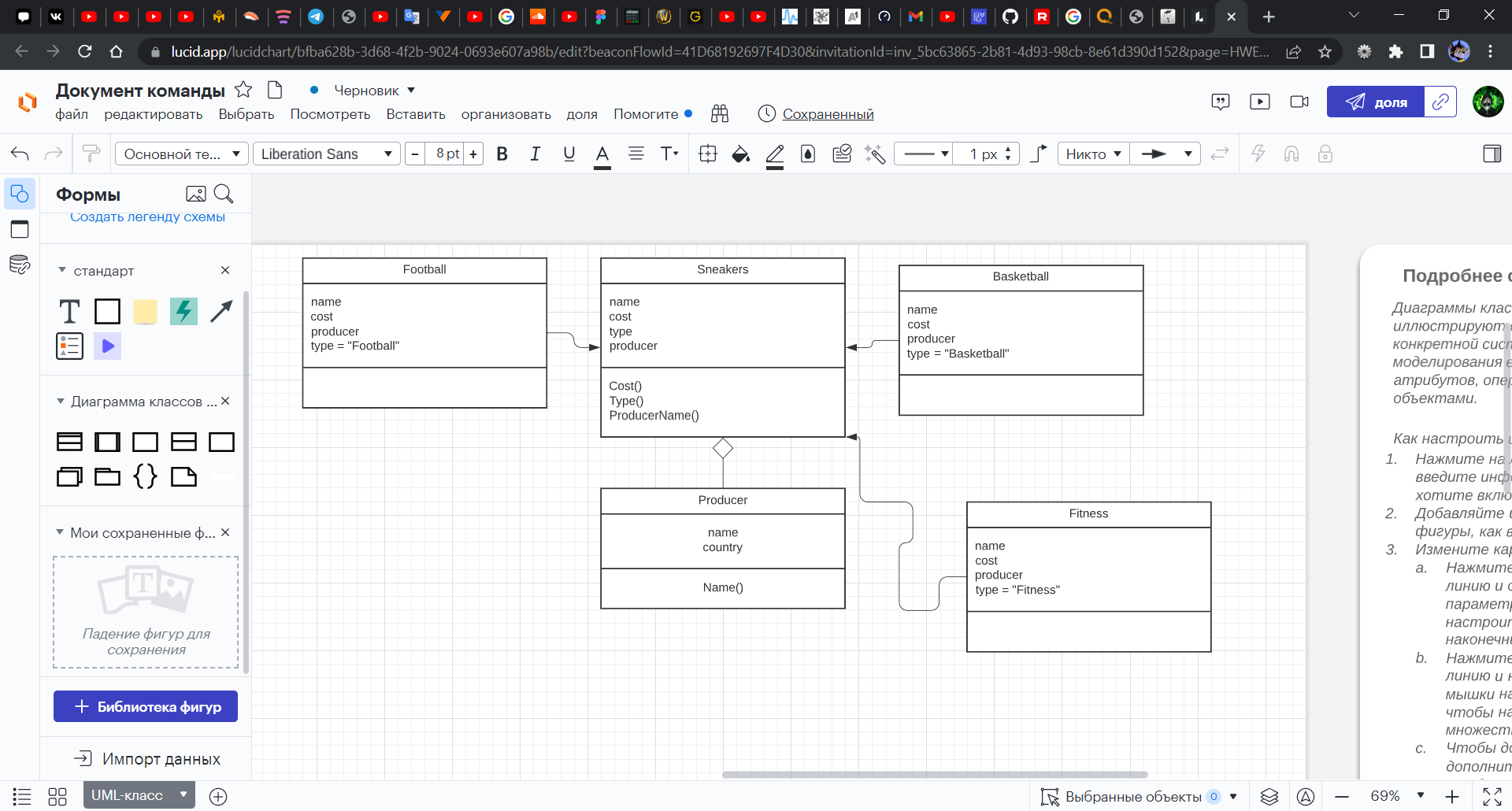


Рисунок 1 – *UML*-диаграмма иерархии классов

Согласно пункту 1.7 задания был добавлен новый класс *“Fitness”*.

На рисунке 2 показан запуск программы, где пользователю предлагается выбрать одну из опций меню.

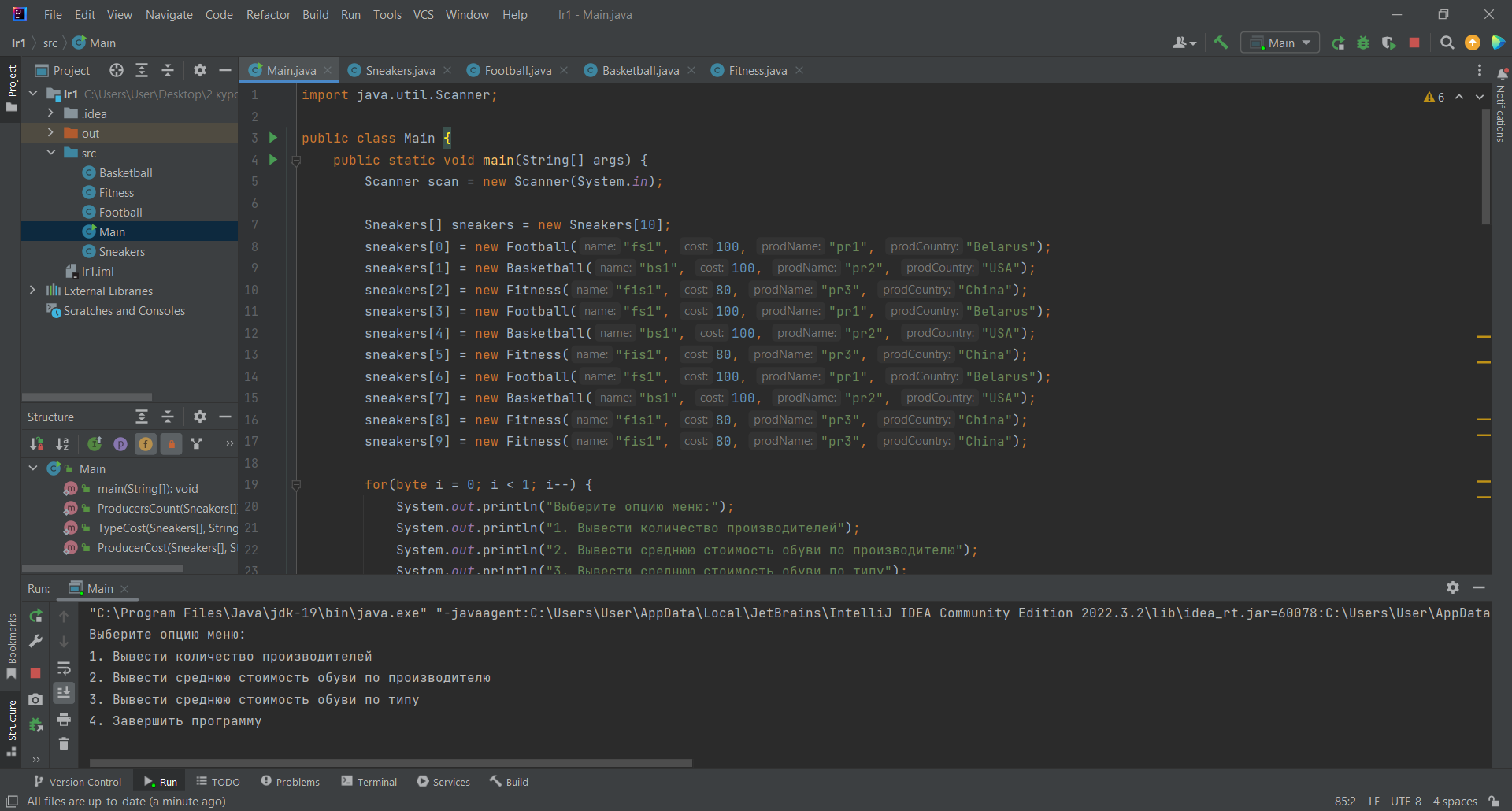


Рисунок 2 – Запуск программы

На рисунке 3 показано окно программы с выводом количества производителей.

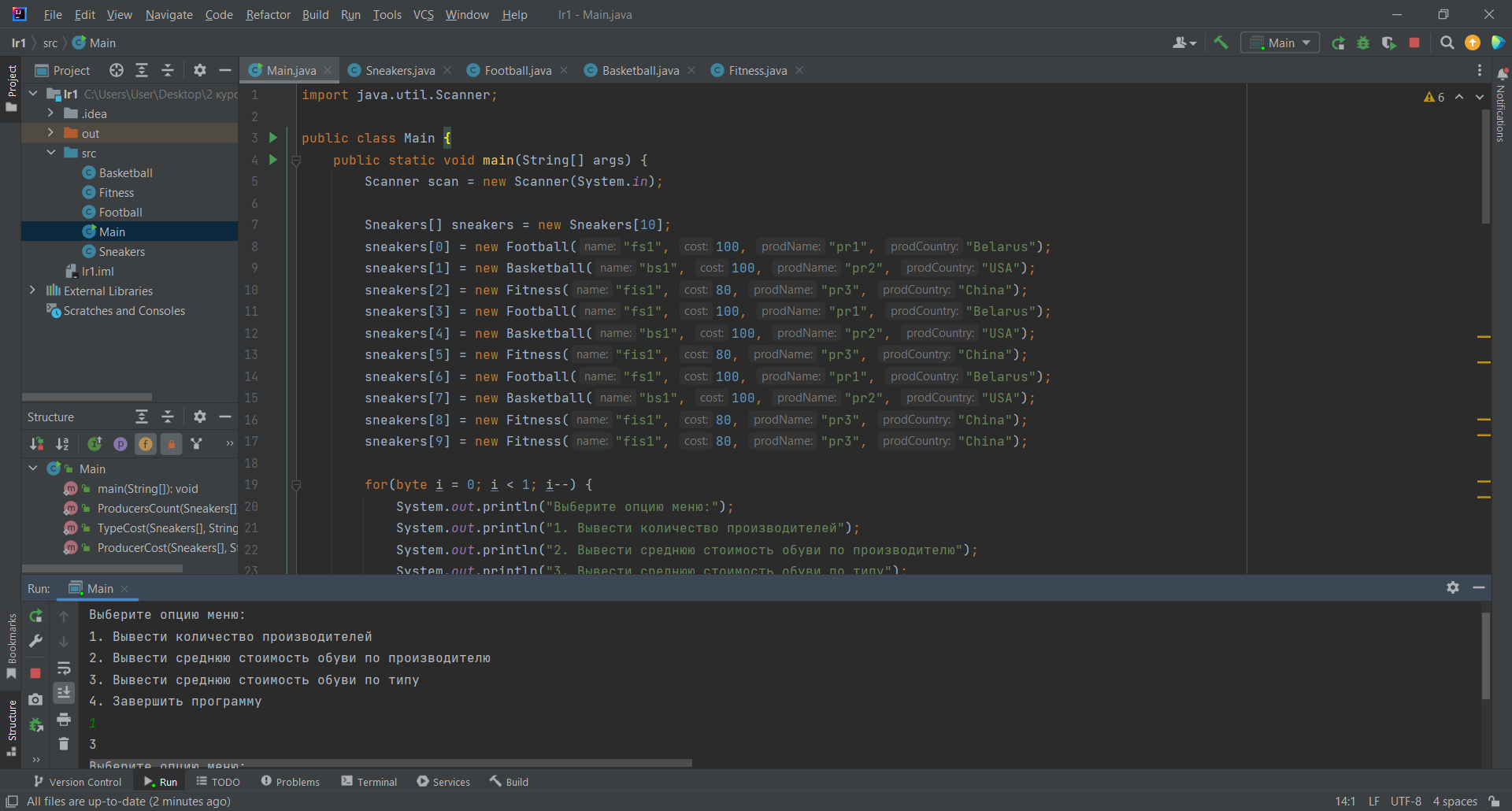


Рисунок 3 – Окно программы

На рисунке 4 показано окно программы с выводом средней стоимости обуви по заданному пользователем производителю.

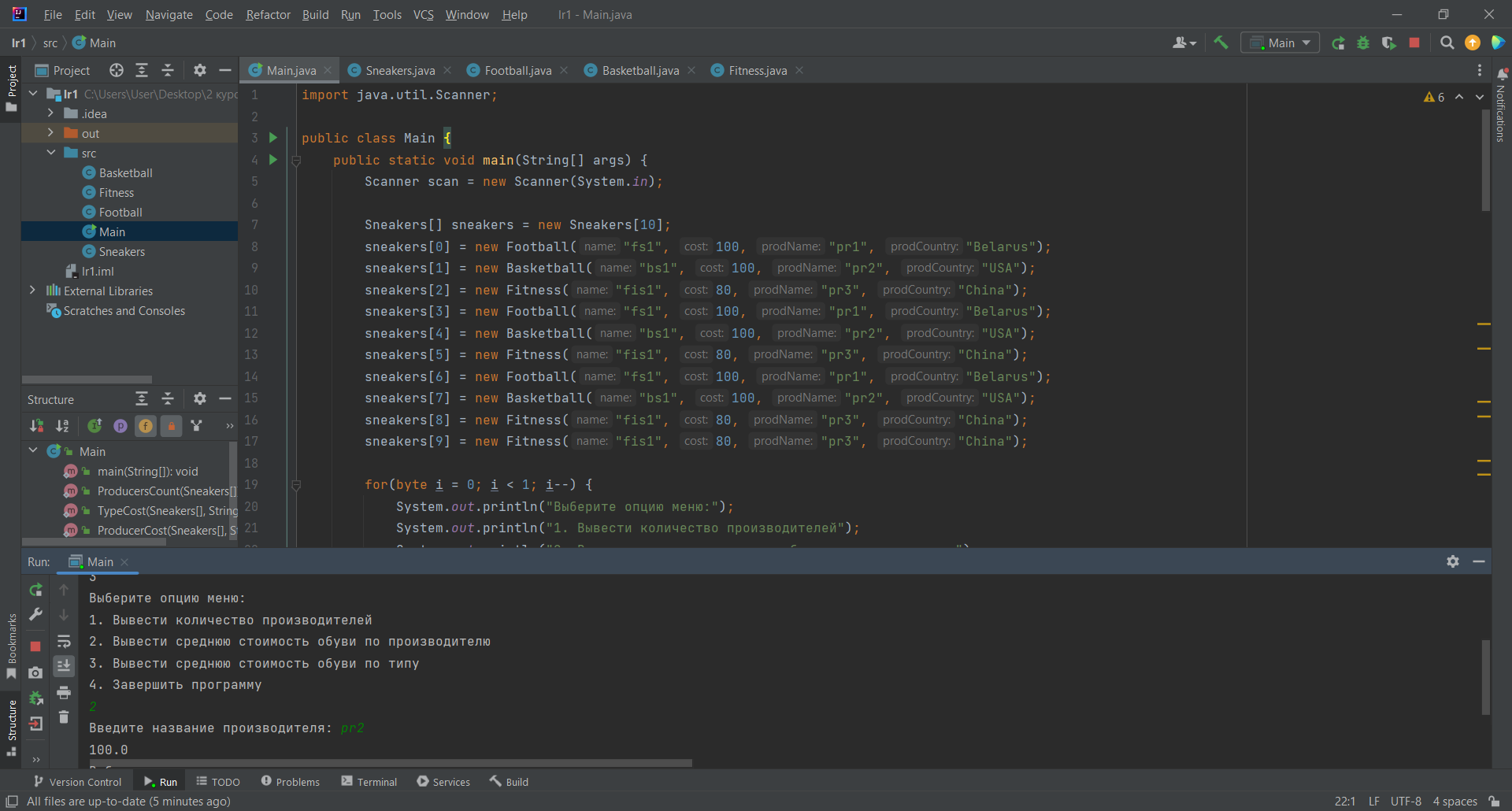


Рисунок 4 – окно программы

На рисунке 5 показано окно программы с выводом средней стоимости обуви по типу, заданному пользователем.

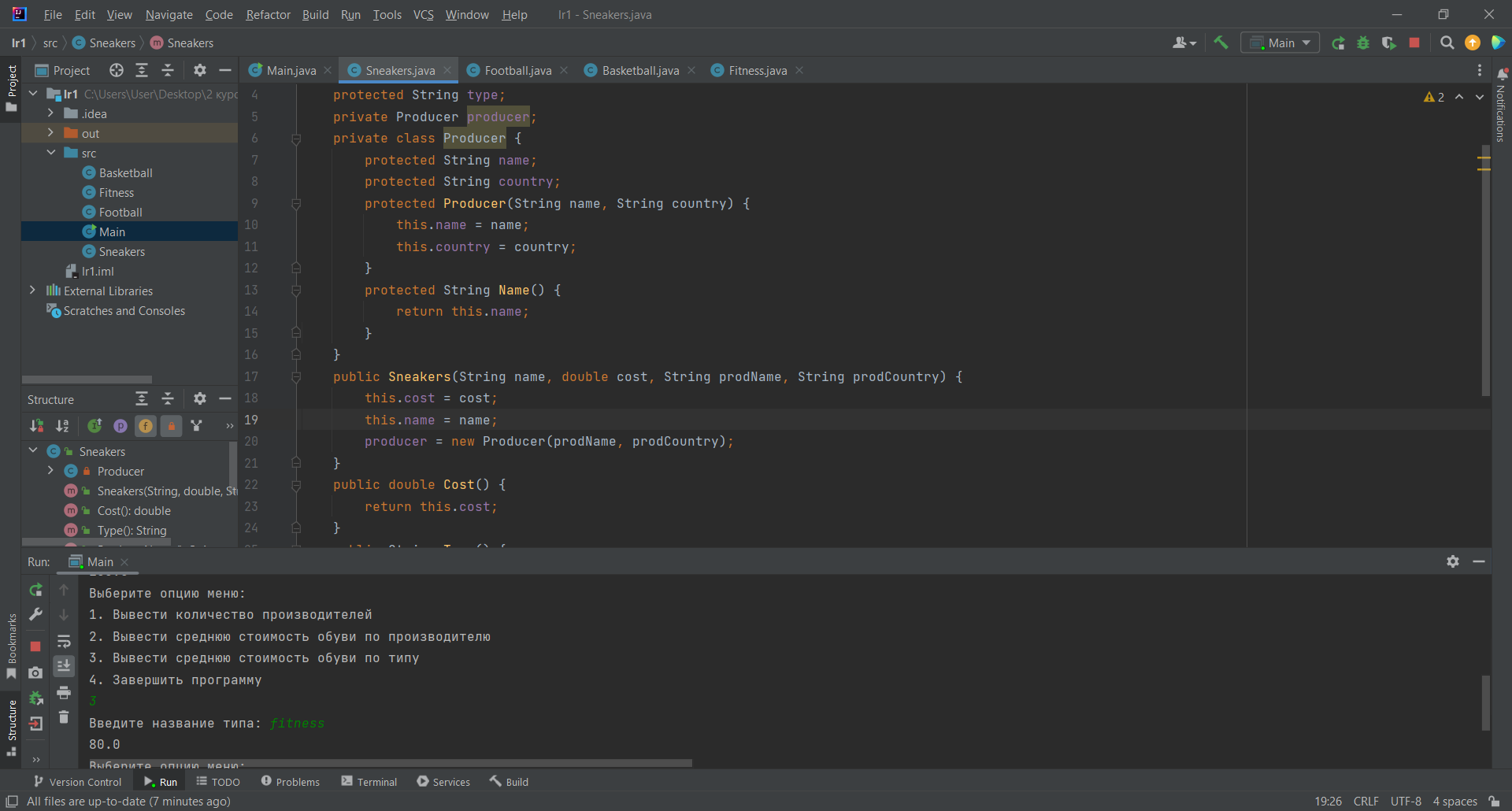


Рисунок 5 – Окно программы

На рисунке 6 показано завершение программы.

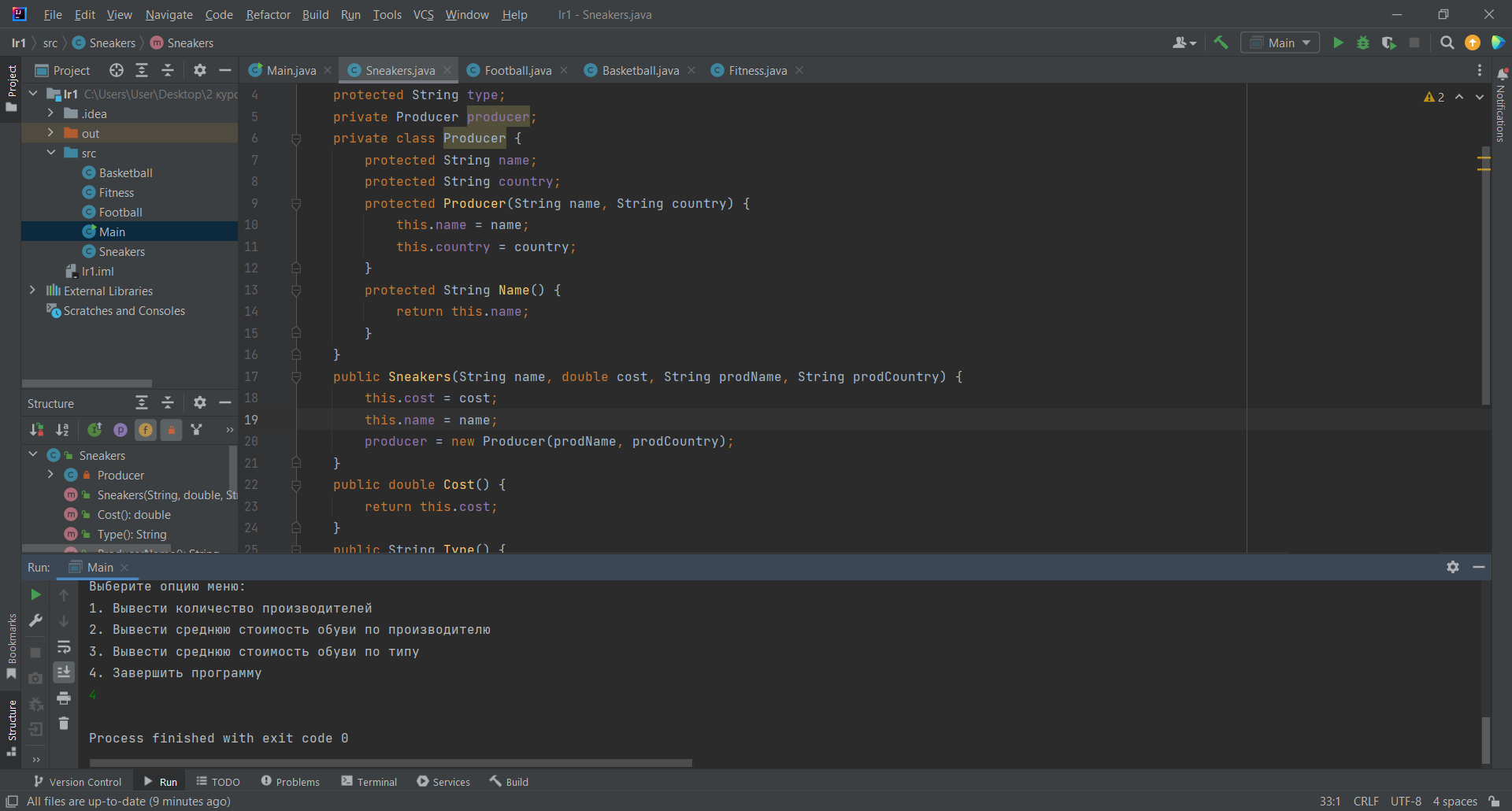


Рисунок 6 – Завершение программы

Для создания наследуемого класса «*Football*» был использован следующий скрипт:

public class Football extends Sneakers{  
 public Football(String name, double cost, String prodName, String prodCountry) {  
 super(name, cost, prodName, prodCountry);  
 this.type = "football";  
 }  
}

Для класса «*Basketball*»:

public class Basketball extends Sneakers{  
 public Basketball(String name, double cost, String prodName, String prodCountry) {  
 super(name, cost, prodName, prodCountry);  
 this.type = "basketball";  
 }  
}

Для класса «*Fitness*»:

public class Fitness extends Sneakers {  
 public Fitness(String name, double cost, String prodName, String prodCountry) {  
 super(name, cost, prodName, prodCountry);  
 this.type = "fitness";  
 }  
}

**Вывод:** в ходе выполнения данной лабораторной работы была составлена *UML*-диаграмма иерархии классов, составлена программа на основе составленной диаграмма на языке программирования *Java.*

**ПРИЛОЖЕНИЕ А**

(обязательное)

**Листинг программы**

**//**Листинг класса Sneakers`

public class Sneakers {  
 protected String name;  
 protected double cost;  
 protected String type;  
 private Producer producer;  
 private class Producer {  
 protected String name;  
 protected String country;  
 protected Producer(String name, String country) {  
 this.name = name;  
 this.country = country;  
 }  
 protected String Name() {  
 return this.name;  
 }  
 }  
 public Sneakers(String name, double cost, String prodName, String prodCountry) {  
 this.cost = cost;  
 this.name = name;  
 producer = new Producer(prodName, prodCountry);  
 }  
 public double Cost() {  
 return this.cost;  
 }  
 public String Type() {  
 return this.type;  
 }  
 public String ProducerName() {  
 return producer.Name();  
 }  
}

//Листинг файла Main.java

import java.util.Scanner;  
  
public class Main {  
 public static void main(String[] args) {  
 Scanner scan = new Scanner(System.in);  
  
 Sneakers[] sneakers = new Sneakers[10];  
 sneakers[0] = new Football("fs1", 100, "pr1", "Belarus");  
 sneakers[1] = new Basketball("bs1", 100, "pr2", "USA");  
 sneakers[2] = new Fitness("fis1", 80, "pr3", "China");  
 sneakers[3] = new Football("fs1", 100, "pr1", "Belarus");  
 sneakers[4] = new Basketball("bs1", 100, "pr2", "USA");  
 sneakers[5] = new Fitness("fis1", 80, "pr3", "China");  
 sneakers[6] = new Football("fs1", 100, "pr1", "Belarus");  
 sneakers[7] = new Basketball("bs1", 100, "pr2", "USA");  
 sneakers[8] = new Fitness("fis1", 80, "pr3", "China");  
 sneakers[9] = new Fitness("fis1", 80, "pr3", "China");  
  
 for(byte i = 0; i < 1; i--) {  
 System.out.println("Выберите опцию меню:");  
 System.out.println("1. Вывести количество производителей");  
 System.out.println("2. Вывести среднюю стоимость обуви по производителю");  
 System.out.println("3. Вывести среднюю стоимость обуви по типу");  
 System.out.println("4. Завершить программу");  
 byte action = scan.nextByte();  
 if (action == 1) {  
 System.out.println(ProducersCount(sneakers));  
 } else if(action == 2) {  
 System.out.print("Введите название производителя: ");  
 scan.nextLine();  
 String producer = scan.nextLine();  
 System.out.println(ProducerCost(sneakers, producer));  
 } else if(action == 3) {  
 System.out.print("Введите название типа: ");  
 scan.nextLine();  
 String type = scan.nextLine();  
 System.out.println(TypeCost(sneakers, type));  
 } else if (action == 4){  
 break;  
 } else {  
 System.out.println("Ошибка выбора.");  
 }  
 }  
 }  
 public static int ProducersCount(Sneakers sneakers[]) {  
 int count = 0;  
 String[] memory = new String[10];  
 for(byte i = 0; i < sneakers.length; i++) {  
 for(byte j = 0; j < memory.length; j ++) {  
 if(sneakers[i].ProducerName().equals(memory[j]) == false & j == sneakers.length - 1) {  
 memory[i] = sneakers[i].ProducerName();  
 count++;  
 break;  
 } else if(sneakers[i].ProducerName().equals(memory[j])) {  
 break;  
 }  
 }  
 }  
 return count;  
 }  
 public static double TypeCost(Sneakers sneakers[], String type) {  
 double cost = 0;  
 double count = 0;  
 for(byte i = 0; i < sneakers.length; i++) {  
 if(sneakers[i].Type().equals(type)) {  
 cost += sneakers[i].Cost();  
 count++;  
 }  
 }  
 cost /= count;  
 return cost;  
 }  
 public static double ProducerCost(Sneakers sneakers[], String producer) {  
 double cost = 0;  
 double count = 0;  
 for(byte i = 0; i < sneakers.length; i++) {  
 if(sneakers[i].ProducerName().equals(producer)) {  
 cost += sneakers[i].Cost();  
 count++;  
 }  
 }  
 cost /= count;  
 return cost;  
 }  
}