|  |  |
| --- | --- |
|  |  |
| МИНОБРНАУКИ РОССИИ |  |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** |  |

**Институт информационных технологий**

КАФЕДРА ИНСТРУМЕТНАЛЬНОГО И ПРИКЛАДНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ (ИиППО)

Практические РАБОТы

по дисциплине «Программирование на языке Джава»

Выполнил студент группы \_\_\_\_\_\_\_\_\_\_\_\_\_\_ *Фамилия И.О.*

*(учебная группа)*

Принял старший преподаватель *Рачков А.В.*

Практические работы работа выполнены «\_\_\_»\_\_\_\_\_\_\_2021г. *подпись студента*

«Зачтено» «\_\_\_»\_\_\_\_\_\_\_2021г. *подпись преподавателя*

Москва 2021

**Оглавление**

[Практическая работа № 1 3](#_Toc81488490)

[Практическая работа № 2 5](#_Toc81488491)

# Практическая работа № 1

**Цель работы**

*Классы, как новые типы данных. Поля данных и методы.*

**Теоретическое введение**

*(не более 1 стр., лучше 0,5 стр.)*

Класс — это тип данных, создаваемый программистом длярешения задач. Он представляет из себя шаблон, или прототип, которыйопределяет и описывает статические свойства и динамическое поведение, общиедля всех объектов одного и того же вида. Экземпляр класса - реализация конкретного объекта типа класса. Все экземпляры класса имеютаналогичные свойства, как задано в определении класса. Например, вы можетеопределить класс с именем "Студент " и создать три экземпляра класса"Студент": " Петр", " Павел" и " Полина ".

Чтобы создать экземпляр класса, вы должны выполнить следующиедействия:

* объявить идентификатор экземпляра (имя экземпляра) конкретного класса
* cконструировать экземпляр класса (то есть выделить память дляэкземпляра и инициализировать его) с помощью оператора "new".

Доступ к компонентам класса осуществляется с помощью операции

получения доступа, а именно операции точка “.”

Переменные — поля данных класса и методы класса и являются компонентами класса. Для ссылки на переменную-поле данных класса или метод, вы должны:

* сначала создать экземпляр класса, который вам нужен;
* затем, использовать оператор точка “.” чтобы сослаться на элемент класса (переменную-поле данных или метод класса).

Метод:

* принимает параметры из вызова (как в функции);
* выполняет операции, описанные в теле метода, и;
* возвращает часть результата (или void) в точку вызова.

Конструктор – это специальный метод класса, который имеет то же имя, что используется в качестве имени класса. Он отличается от обычного метода следующим:

* название метода-конструктора совпадает с именем класса, а имя класса по конвенции, начинается с заглавной буквы;
* конструктор не имеет возвращаемого значения типа, таким образом, нет объявления типа возвращаемого значения при объявлении;
* конструктор может быть вызван только через оператор «new», он может быть использован только один раз, чтобы инициализировать построенный
* экземпляр.
* вы не можете впоследствии вызвать конструктор в теле программы подобно обычным методам (функциям);
* конструкторы не наследуется (будет объяснено позже).

Конструктор без параметров называется конструктором по умолчанию, который инициализирует переменные-поля данных через их значения по умолчанию.

**Выполнения лабораторной работы**

*Задание:*

*Если нет конкретного, сформулировать самостоятельно.*

*Реализовать простейший класс «Книга».*

*Решение:*

*Здесь могут быть схемы. коды, скриншоты и т.п. Обращай те внимание на подписи таблиц (вверху по левому краю) и рисунков (снизу по центру). Перед их появление, в тексте должна быть ссылка. Весь отчет оформляется в соответствии с ГОСТ.*

Содержимое файла Book.java:

package ru.mirea.lab1;  
import java.lang.\*;

public class Book {  
 private String name;  
 private String author;  
 private int pages;  
  
 public Book(String name, String author, int pages) {  
 this.name = name;  
 this.author = author;  
 this.pages = pages;  
 }  
  
 public Book(String name) {  
 this.name = name;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getAuthor() {  
 return author;  
 }  
  
 public void setAuthor(String author) {  
 this.author = author;  
 }  
  
 public int getPages() {  
 return pages;  
 }  
  
 public void setPages(int pages) {  
 this.pages = pages;  
 }  
 public String toString() {  
 return "Книга: " + name + "; Автор: " + author + "; Страниц: " + pages;  
 }  
}

Содержимое файла TestBook.java:

package ru.mirea.lab1;  
import java.lang.\*;  
  
public class TestBook {  
 public static void main(String[] args) {  
 Book b1 = new Book("Война и мир", "Лев Толстой", 1696);  
 Book b2 = new Book("Мертвые души", "Николай Гоголь", 355);  
 Book b3 = new Book("Евгений Онегин");  
 b3.setAuthor("Александр Пушкин");  
 b3.setPages(448);  
 System.out.println(b1.toString());  
 System.out.println(b2.toString());  
 System.out.print(b3.toString());  
 }  
}

На рисунке 1 показан вывод программы. По нему можно убедится, что программа работает правильно.

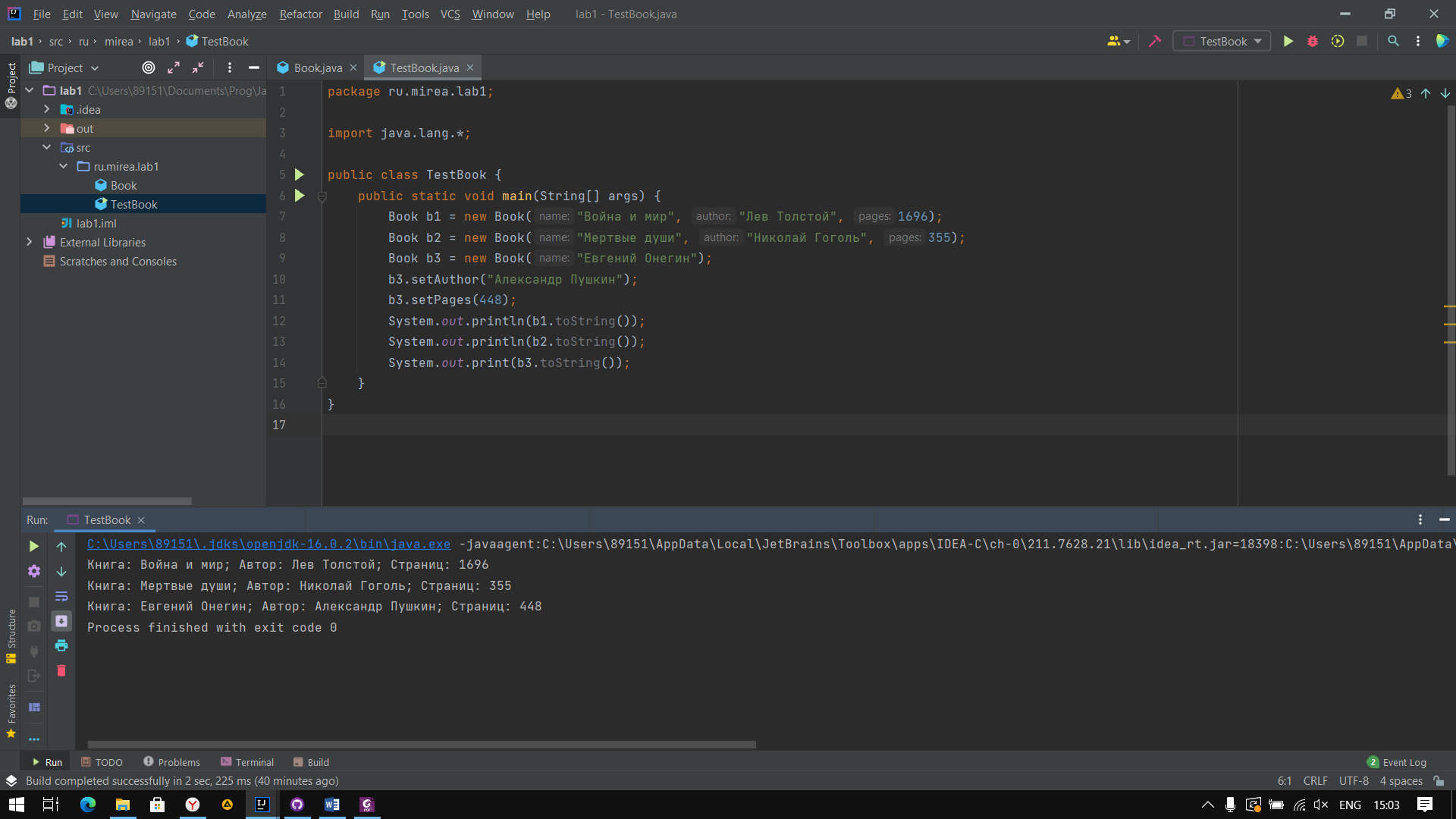


Рисунок 1 - Вывод программы.

**Выводы по работе:**

*Делается кратко по каждой работе.*

*Выполняя данную практическую работу, я научился создавать класс, поля, методы и конструкторы в нем, а также создавать экземпляры класса и обращаться к его полям и методам.*

# Практическая работа № 2

**Цель работы**

*Использование UML диаграмм в объектно-ориентированном программировании*

**Теоретическое введение**

*(не более 1 стр., лучше 0,5 стр.)*

Язык моделирования Unified Modeling Language (UML) является стандартом с 1998 года для проектирования и документирования объектно-ориентированных программ.

Средствами UML в виде диаграмм можно графически изобразить класс и экземпляр класса. Графически представляем класс в виде прямоугольника, разделенного на три области – область именования класса, область инкапсуляции данных и область операций (методы).

Имя определяет класс. Переменные содержат статические атрибуты класса, или описывают свойства класса. Методы описывают динамическое поведение класса. Другими словами, класс инкапсулирует статические свойства (данные) и динамические модели поведения (операции, которые работают с данными) в одном месте (“коробке” или прямоугольнике).

На рисунке 2 приведен общий вид UML диаграммы класса.

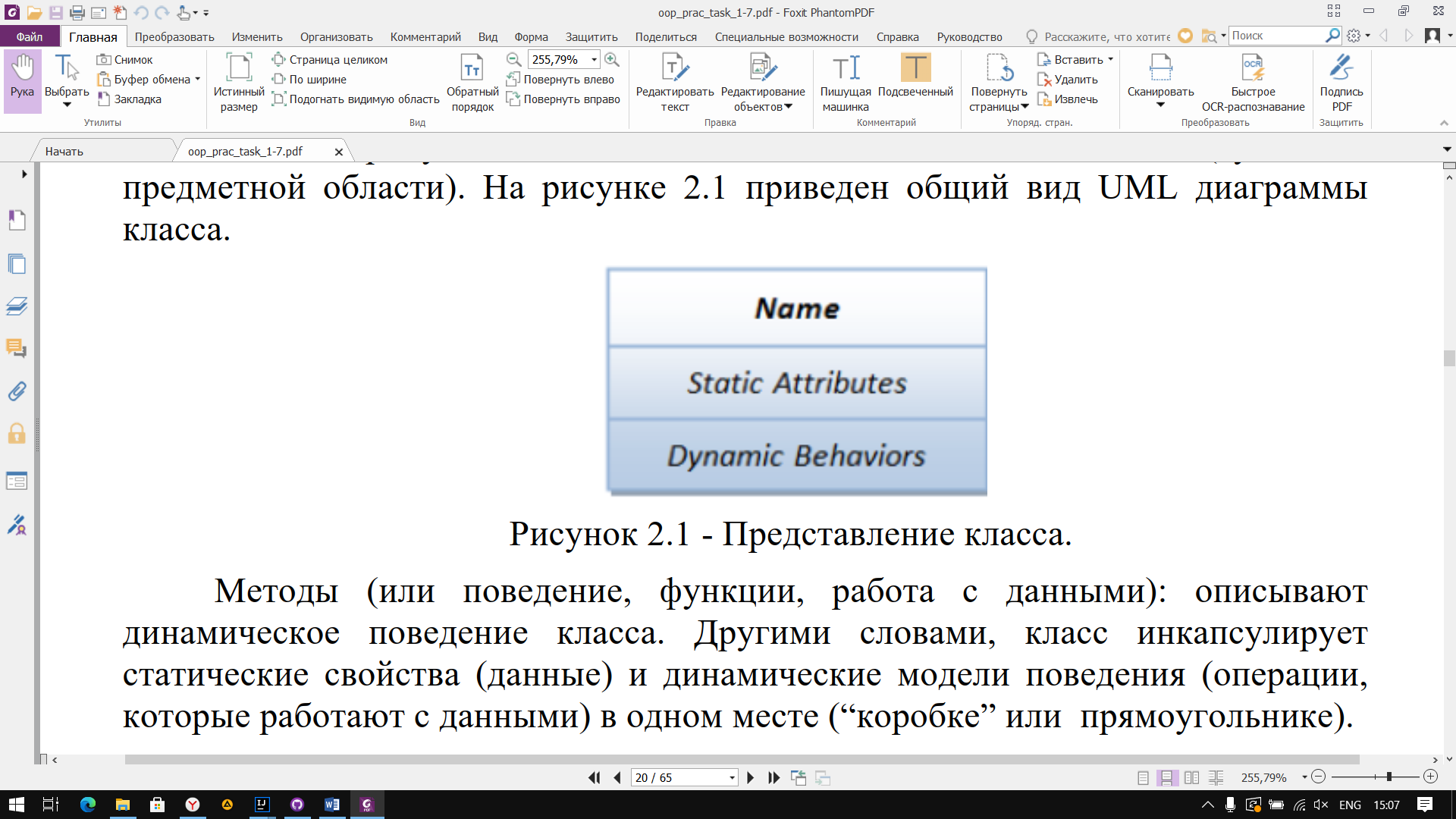


Рисунок 2 - Представление класса.

На рисунке 3 показаны два экземпляра класса типа Student "paul" и"peter".

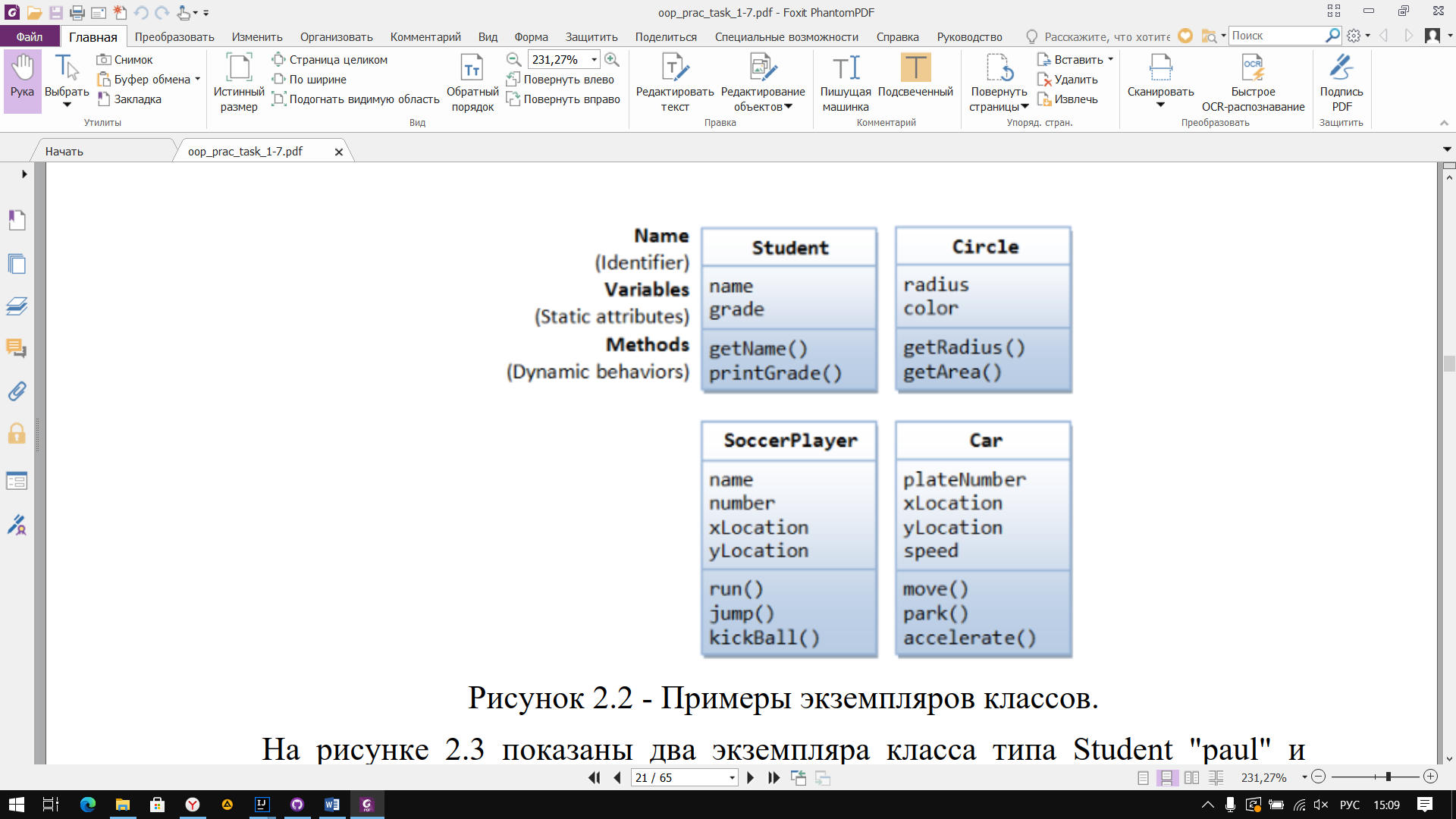


Рисунок 3 - Примеры экземпляров классов.

Приведенные выше диаграммы классов описаны в соответствии с UML

нотацией. Класс представляется в этой нотации как прямоугольник, разделенный

на три области, одна содержит название, две вторых содержат поля и методы класса, соответственно. Имя класса выделено жирным шрифтом и находится посредине. Экземпляр также представляется в виде прямоугольника, разделенного на три части, в первой части помещается надпись с именем экземпляра.

**Выполнения лабораторной работы**

*Задание:*

*Если нет конкретного, сформулировать самостоятельно.*

*По диаграмме класса UML описывающей сущность Автор. Необходимо*

*написать программу, которая состоит из двух классов Author и TestAuthor. Класс*

*Author должен содержать реализацию методов, представленных на диаграмме*

*класса на рисунке 4.*

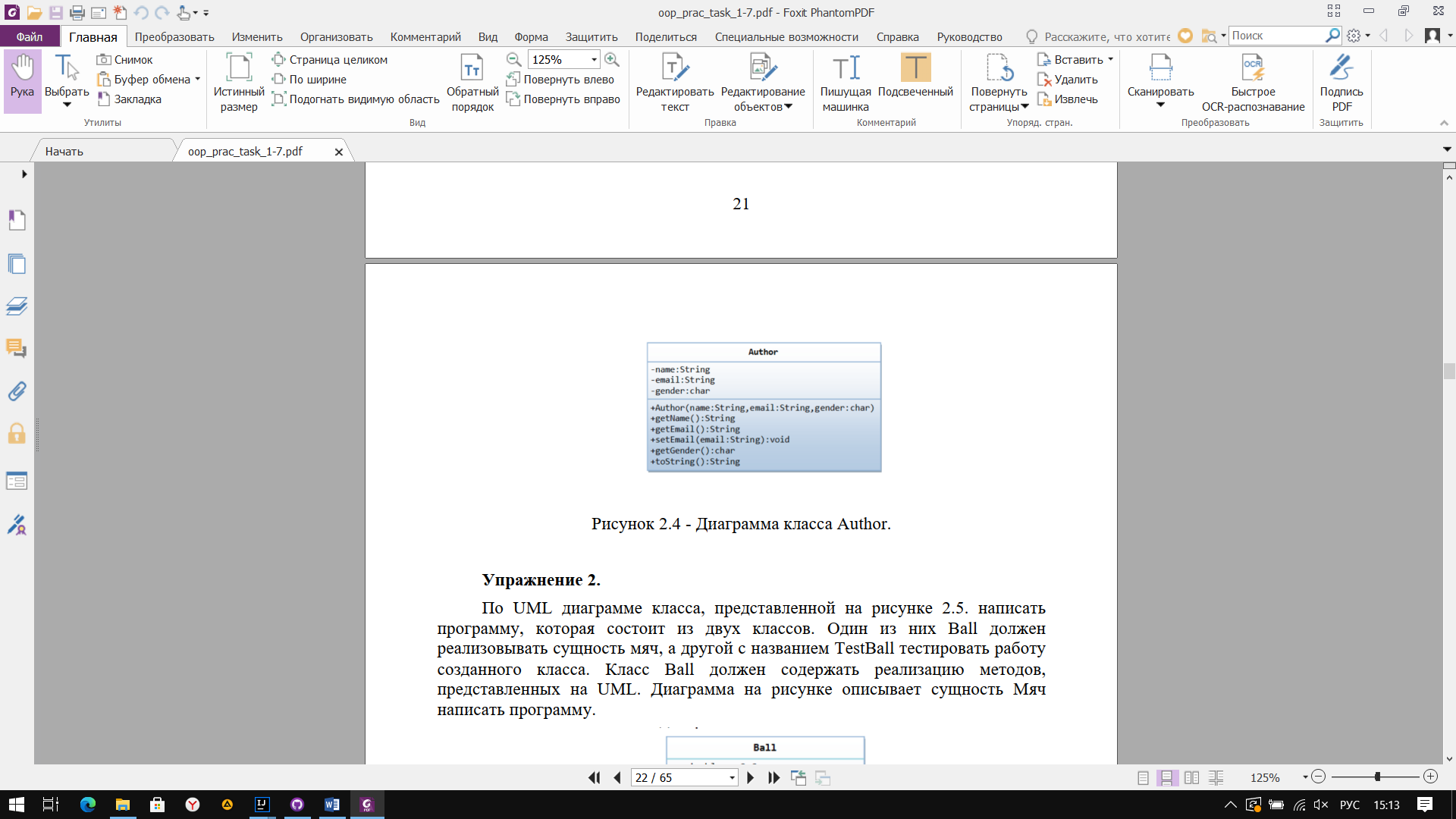


Рисунок 4 – Диаграмма класса Author.

*Решение:*

*Здесь могут быть схемы. коды, скриншоты и т.п. Обращай те внимание на подписи таблиц (вверху по левому краю) и рисунков (снизу по центру). Перед их появление, в тексте должна быть ссылка. Весь отчет оформляется в соответствии с ГОСТ.*

*Содержимое класса Author.java:*

package ru.mirea.lab2;

import java.lang.\*;

public class Author {

private String name;

private String email;

private char gender;

public Author(String name, String email, char gender) {

this.name = name;

this.email = email;

this.gender = gender;

}

public String getName() {

return name;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public char getGender() {

return gender;

}

public String toString() {

String genderForReturn;

if (gender == 'M') {

genderForReturn = "m";

} else if (gender == 'F') {

genderForReturn = "ms";

} else {

genderForReturn = "unknown";

}

return name + " (" + genderForReturn + ") at " + email;

}

}

*Содержимое класса TestAuthor.java:*

package ru.mirea.lab2;  
import java.lang.\*;  
  
public class TestAuthor {  
 public static void main(String[] args) {  
 Author a1 = new Author("Ivan Popov", "ivPopov@somewhere.com", 'M');  
 Author a2 = new Author("Anna Ivanova", "anIvanova@somewhere.com", 'F');  
 System.out.println(a1.toString());  
 System.out.print(a2.toString());  
 }  
}

На рисунке 5 показан вывод программы. По нему можно убедится, что программа работает правильно.

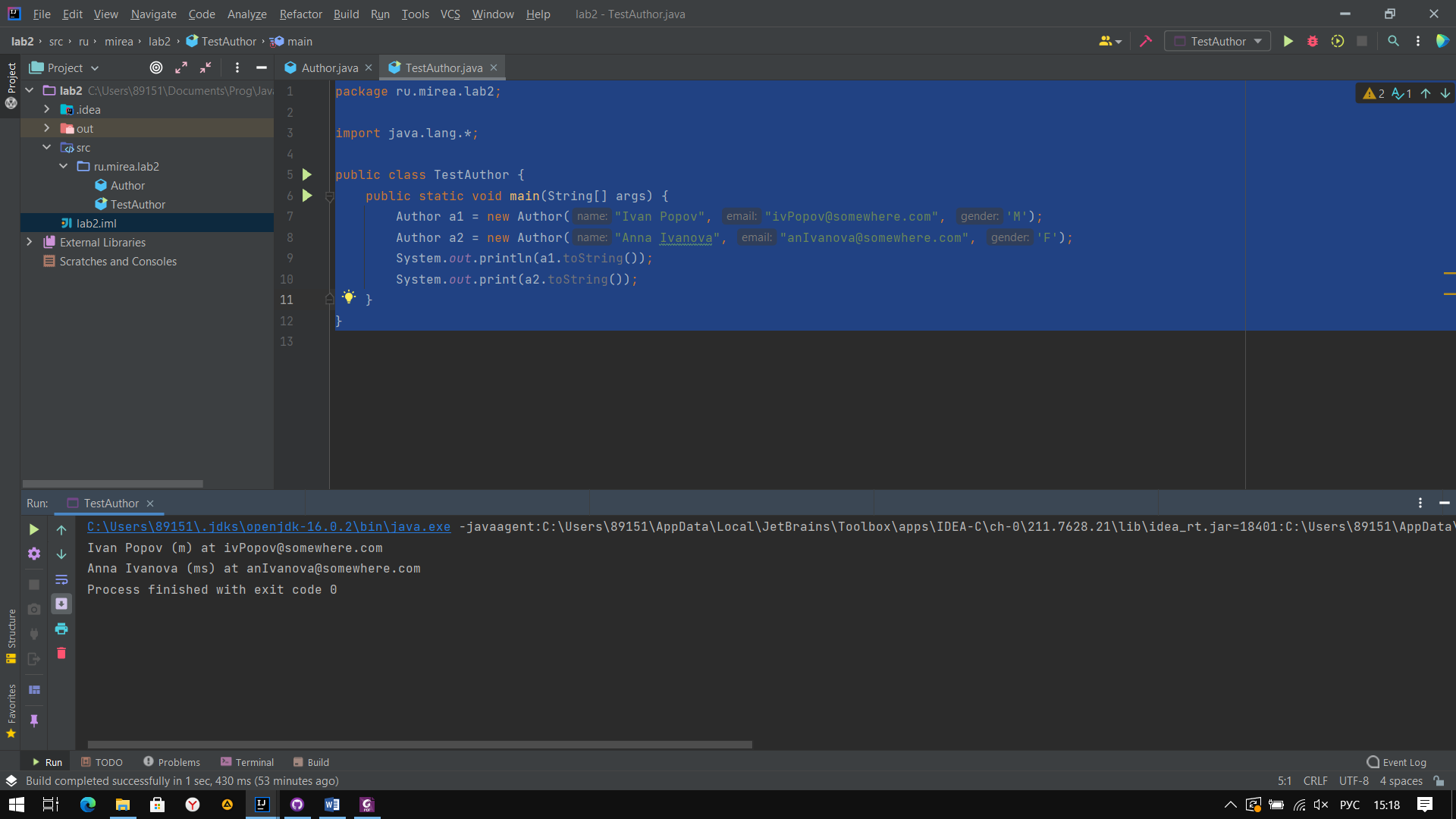


Рисунок 5 - Вывод программы.

**Выводы по работе:**

*Делается кратко по каждой работе.*

*Выполняя данную практическую работу, я научился создавать класс, поля, методы и конструкторы в нем по диаграммам класса UML, а также обращаться к его полям и методам.*