|  |  |
| --- | --- |
|  | |
| МИНОБРНАУКИ РОССИИ | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | |
| Институт инновационных технологий и государственного управления (ИНТЕГУ) | |
| Кафедра информационных технологий в государственном управлении (ИТГУ) | |

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ** | |
| **по дисциплине** | |
| **«Объективно-ориентированное программирование»**  **на тему**  **«ИСПОЛЬЗОВАНИЕ UML ДИАГРАММ В ОБЪЕКТНО- ОРИЕНТИРОВАННОМ ПРОГРАММИРОВАНИИ»** | |
|  | |
| Выполнил студент группы ГИБО-03-17 | *Стояновска М.* |
| Принял  *Старший преподаватель* | *Хлебникова В.Л*  *.* |

|  |  |  |
| --- | --- | --- |
| Практическая работа выполнена | «\_\_»\_\_\_\_\_\_\_201\_\_\_ г. |  |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_201\_\_\_ г. |  |

Москва 2019

Оглавление

[Цель работы 3](#_Toc26999557)

[Зaдание 4](#_Toc26999558)

[Код и результат 5](#_Toc26999559)

# Цель работы

Цель данной практической работы – работа с UML-диаграммами классов.

# 

# Зaдание

**Упражнение 1.**

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 2.4.

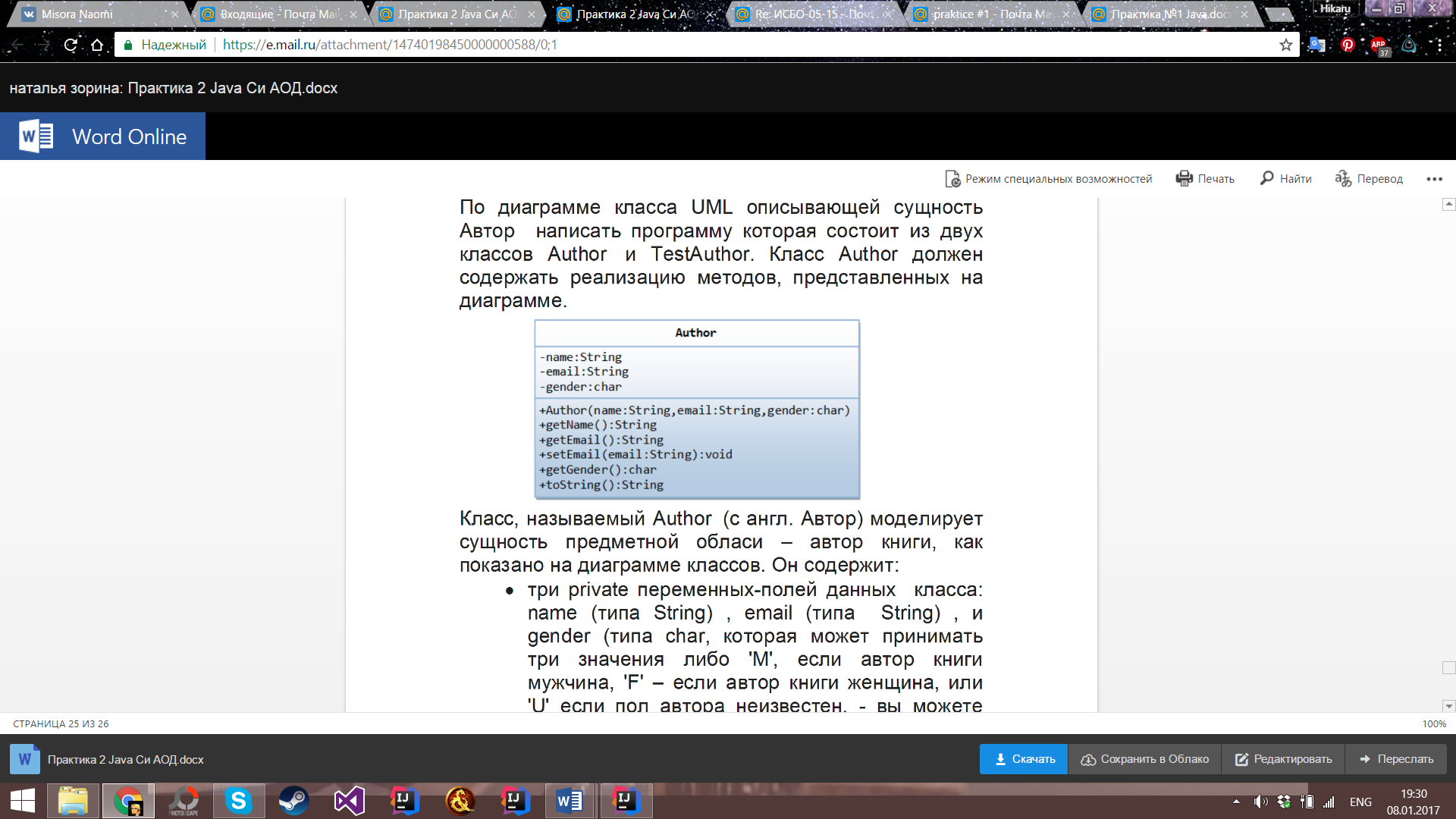


Рисунок 2.4 - Диаграмма класса Author.

**Упражнение 2.**

По UML диаграмме класса, представленной на рисунке 2.5. написать программу, которая состоит из двух классов. Один из них Ball должен реализовывать сущность мяч, а другой с названием TestBall тестировать работу созданного класса. Класс Ball должен содержать реализацию методов, представленных на UML. Диаграмма на рисунке описывает сущность Мяч написать программу.

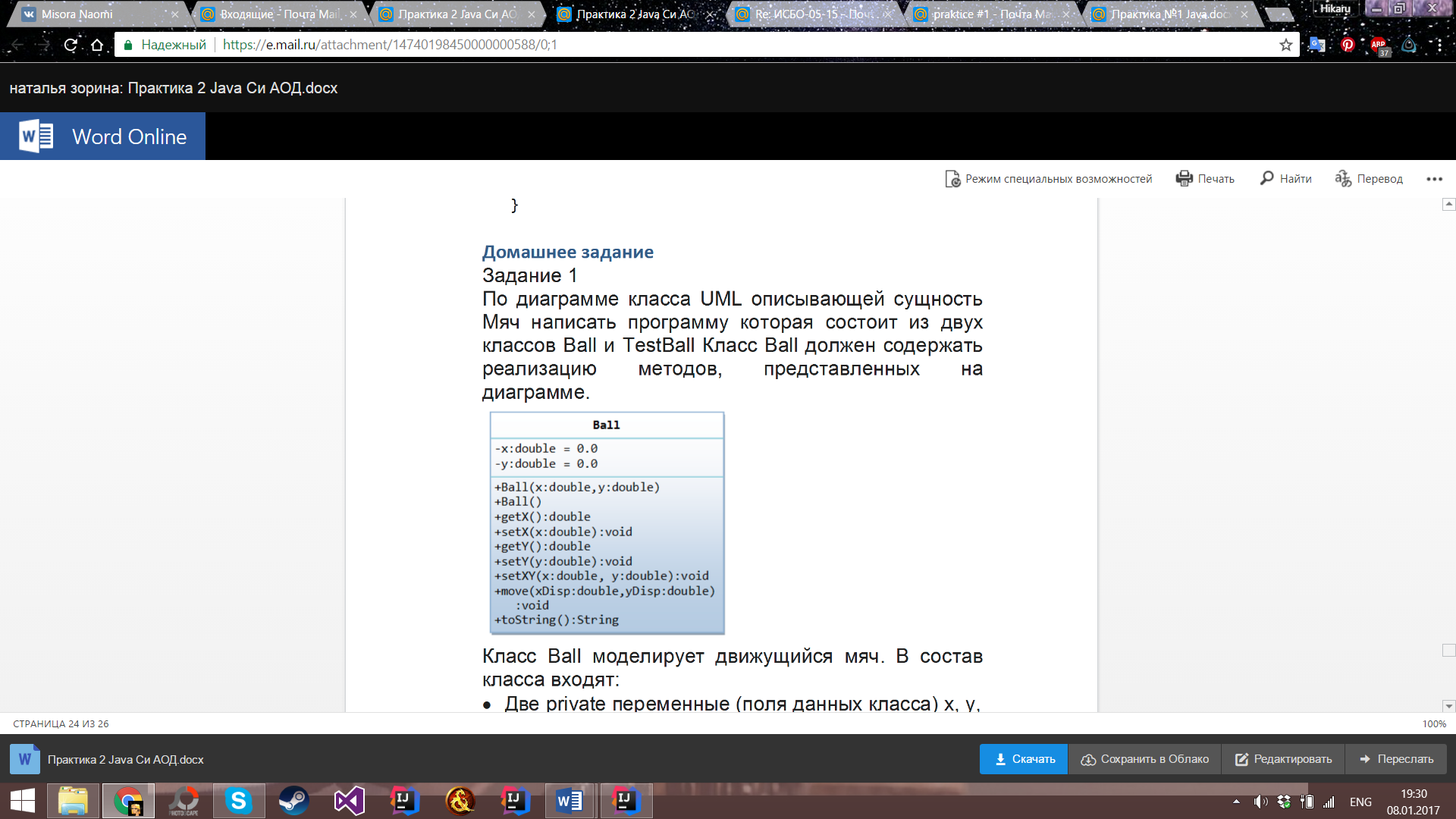


Рисунок 2.5 - Диаграмма класса Ball.

Класс Ball моделирует движущийся мяч. В состав класса входят:

* Две переменные с модификатором private (поля данных класса): х, у, которые описывают положение мяча на поле.
* Конструкторы, public методы получения и записи значений для private переменных.
* Метод setXY (), который задает положение мяча и метод setXYSpeed(), чтобы задать скорость мяча
* Метод move() , позволяет переместить мяч, так что что увеличивает х и у на данном участке на xDisp и yDisp, соответственно.
* Метод toString(), который возвращает "Ball @ (х , у) " .

**ПРИМЕР ВЫПОЛНЕНИЯ ЗАДАНИЯ.**

Класс, называемый Author (с англ. Автор), как показано на диаграмме классов, моделирует сущность предметной области – автор книги.

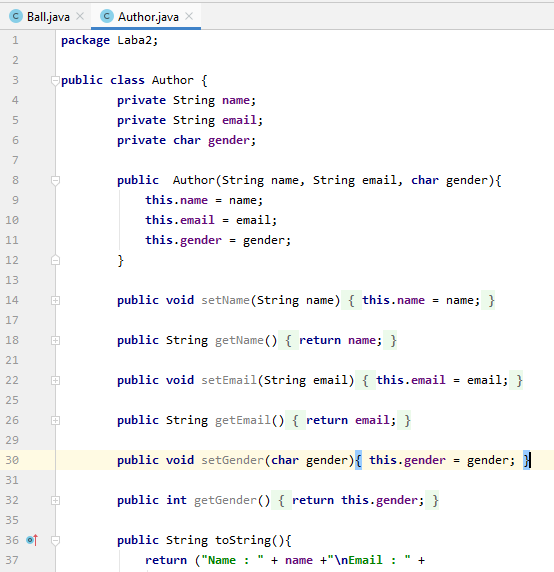
Он содержит:

* три private переменных-поля данных класса: name (типа String) , email (типа String) , и gender (типа char, которая может принимать три значения либо 'M', если автор книги мужчина, 'F' – если автор книги женщина, или 'U' если пол автора неизвестен, - вы можете также использовать для реализации логическую переменную под названием male для обозначения пола автора, которая будет принимать значение истина или ложь);
* один конструктор для инициализации переменных name, email и gender с заданными значениями. (здесь не будет использоваться конструктор по умолчанию, так как нет значений по умолчанию: ни для имени, ни для электронной почты или пола).
* стандартные методы класса, геттеры/сеттеры: getName(), должны быть объявлены с модификатором public;
* методы getEmail(), setEmail(), and getGender(), нужно упомянуть, что класс не содержит методов сеттеров для полей данных - имени и пола, так как эти атрибуты не могут изменяться;
* метод toString(), который должен возвращать следующий текст "автор - имя (пол) на адрес электронной почты, например, " Ivan Popov(m) at ivPopov@somewhere.com ", или "Anna Ivanova (ms) at anIvanova@somewhere.com ", то есть в строке должно быть записано имя[пробел](пол)[пробел]at[пробел]емайл

# Код и результат

**Класс Автор**

**Скрины:**

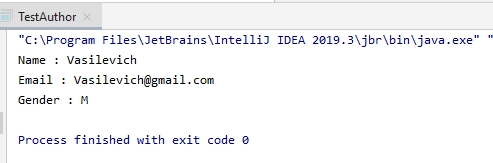




**Код программы:**

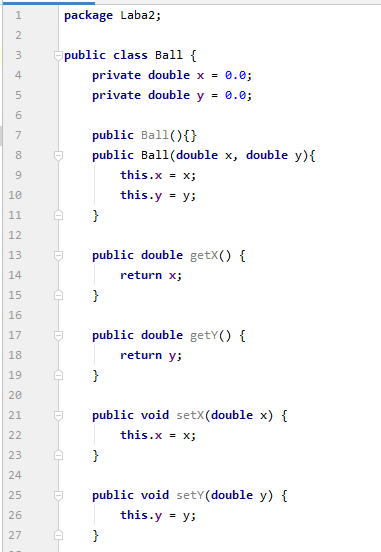
**package** Laba2;  
  
**public class** Author {  
 **private** String **name**;  
 **private** String **email**;  
 **private char gender**;  
  
 **public** Author(String name, String email, **char** gender){  
 **this**.**name** = name;  
 **this**.**email** = email;  
 **this**.**gender** = gender;  
 }  
  
 **public void** setName(String name){  
 **this**.**name** = name;  
 }  
  
 **public** String getName(){  
 **return name**;  
 }  
  
 **public void** setEmail(String email){  
 **this**.**email** = email;  
 }  
  
 **public** String getEmail(){  
 **return email**;  
 }  
  
 **public void** setGender(**char** gender){ **this**.**gender** = gender; }  
  
 **public int** getGender(){  
 **return this**.**gender**;  
 }  
  
 **public** String toString(){  
 **return** (**"Name : "** + **name** +**"\nEmail : "** +  
 **this**.**email** + **"\nGender : "** + **this**.**gender**);  
 }  
}

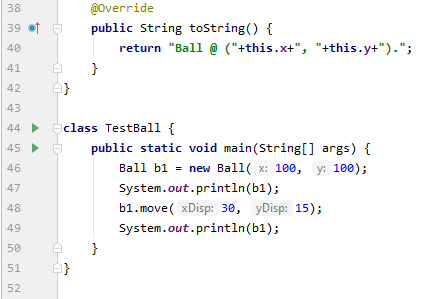
**public class** TestAuthor {  
 **public static void** main(String[] arg){  
 Author b1 = **new** Author(**"Vasilevich"**,**"Vasilevich@gmail.com"**, **'M'**);  
 System.***out***.println(b1.toString());  
 }  
}



**Далее мы реализуем класс “Ball”- мяч**

**Скрины:**





**Код программы:**

**package** Laba2;  
  
**public class** Ball {  
 **private double x** = 0.0;  
 **private double y** = 0.0;  
  
 **public** Ball(){}  
 **public** Ball(**double** x, **double** y){  
 **this**.**x** = x;  
 **this**.**y** = y;  
 }  
  
 **public double** getX() {  
 **return x**;  
 }  
  
 **public double** getY() {  
 **return y**;  
 }  
  
 **public void** setX(**double** x) {  
 **this**.**x** = x;  
 }  
  
 **public void** setY(**double** y) {  
 **this**.**y** = y;  
 }  
  
 **public void** setXY(**double** x, **double** y){  
 **this**.**x** = x;  
 **this**.**y** = y;  
 }  
  
 **public void** move( **double** xDisp, **double** yDisp){  
 **x**+=xDisp;  
 **y**+=yDisp;  
 }  
 @Override  
 **public** String toString() {  
 **return "Ball @ ("**+**this**.**x**+**", "**+**this**.**y**+**")."**;  
 }  
}  
  
**class** TestBall {  
 **public static void** main(String[] args) {  
 Ball b1 = **new** Ball(100, 100);  
 System.***out***.println(b1);  
 b1.move(30, 15);  
 System.***out***.println(b1);  
 }  
}

