Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«Российская академия народного хозяйства и государственной службы при Президенте Российской Федерации»

Нижегородский институт управления

Кафедра Информатики и информационных технологий

Дисциплина «Программирование»

ОТЧЕТ

О ПРОДЕЛАННОЙ ПРАКТИЧЕСКОЙ РАБОТЕ № 9

Выполнила студентка группы Иб-321

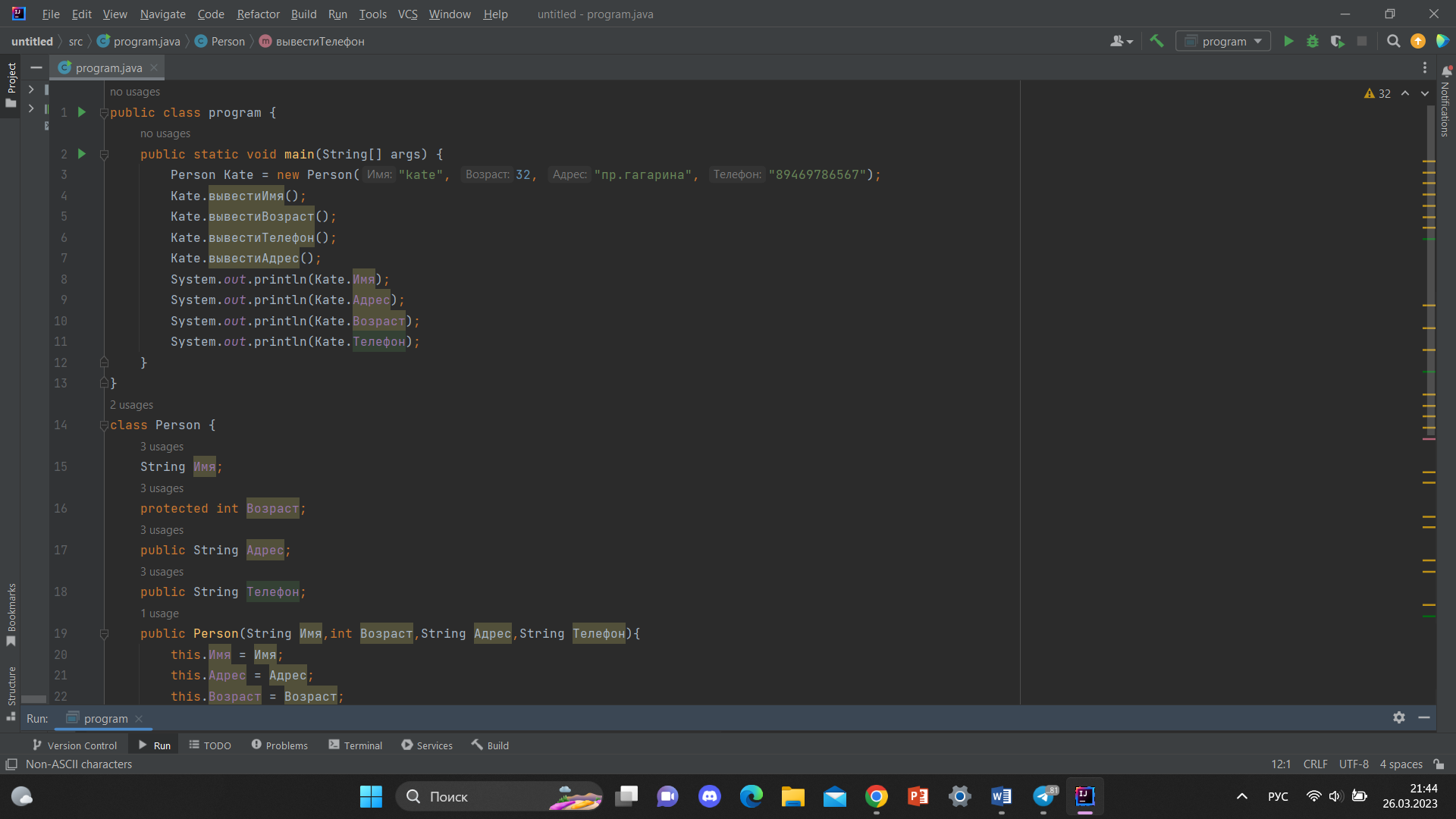
Солдатова Елена Николаевна

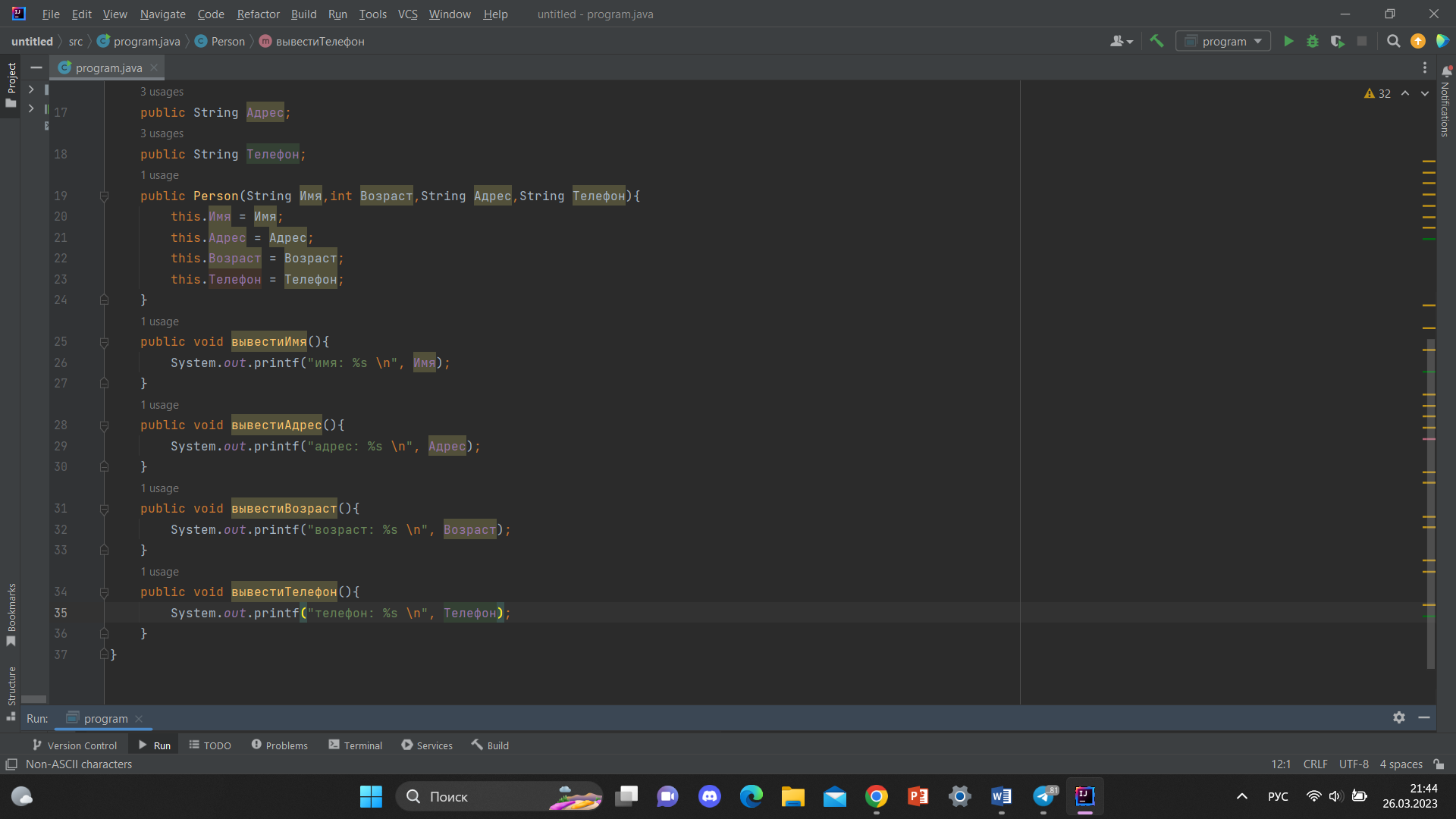
Нижний Новгород

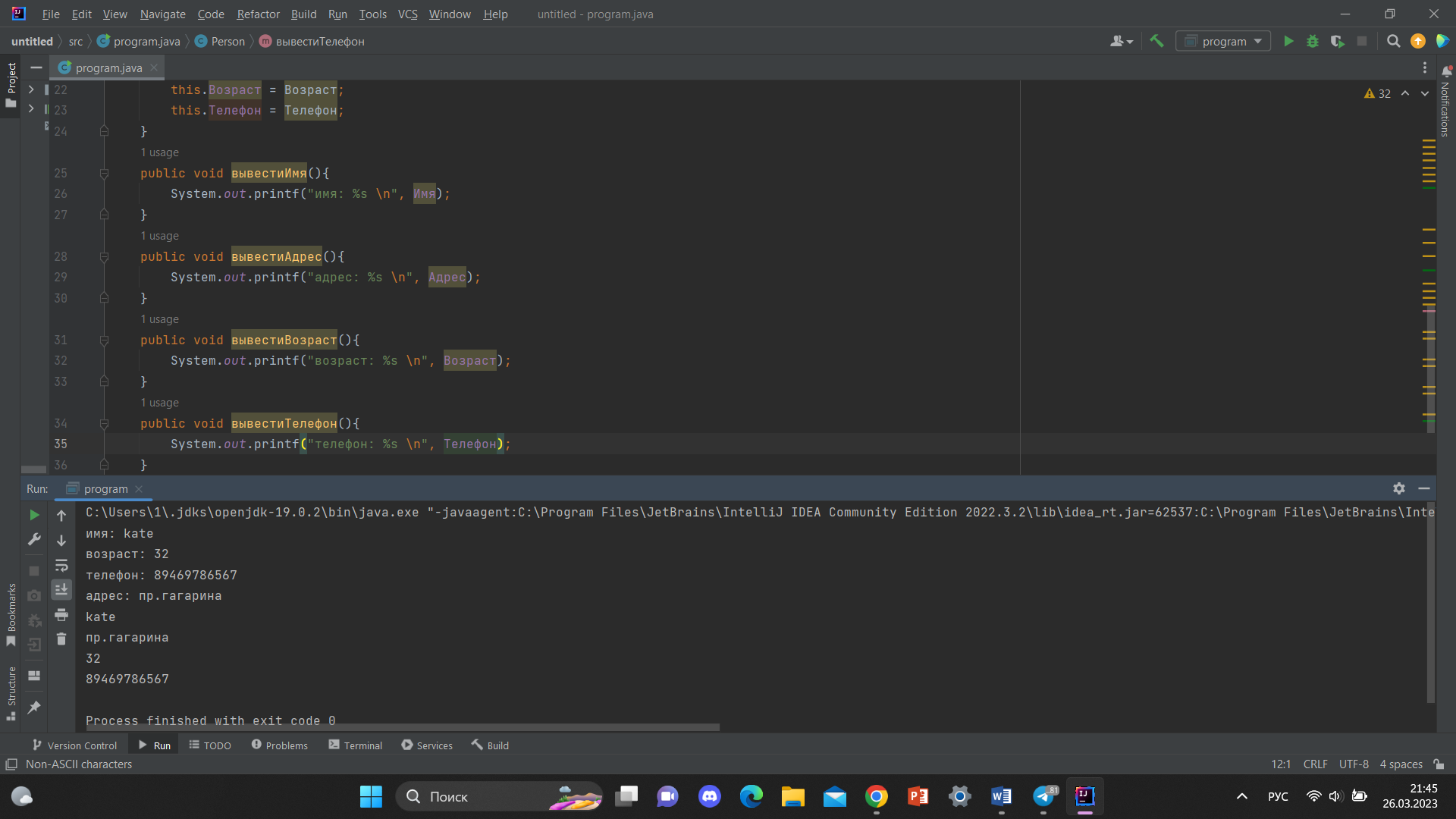
2023 г.

**Задание 9.1**

* 1. Реализовать проекты с программами рисунков 9.1 и 9.2.



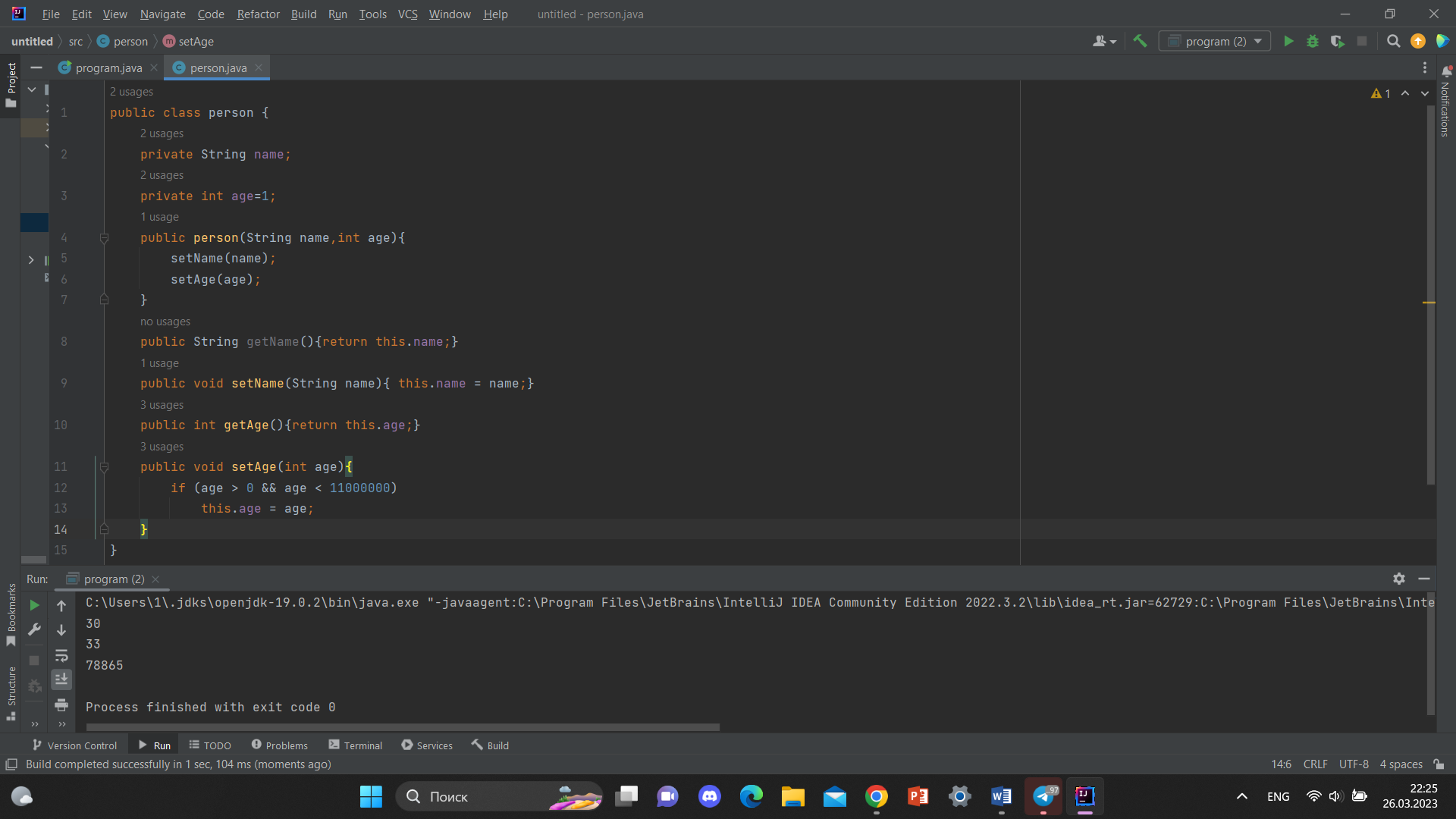


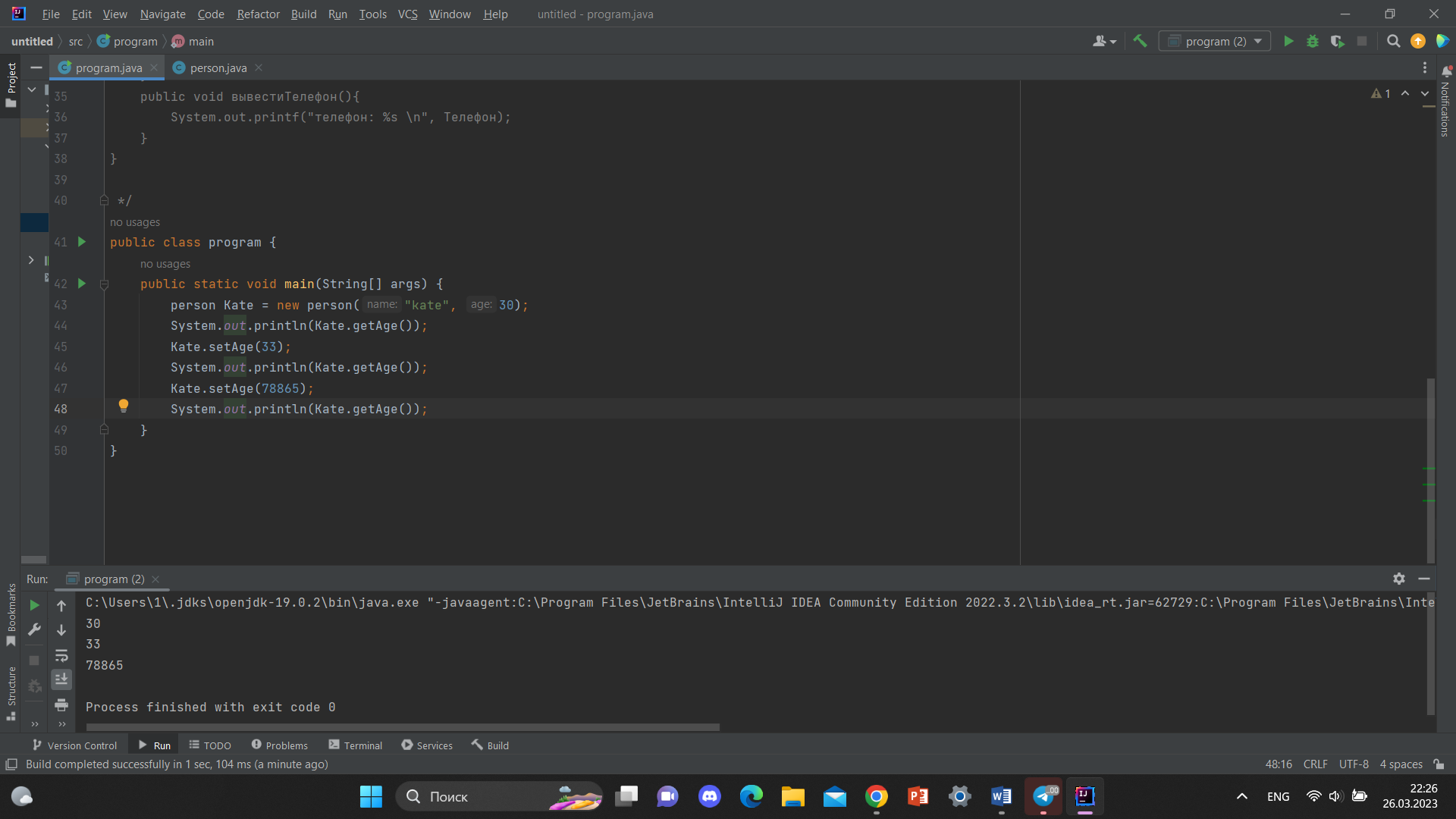


* 1. Определить (идентифицировать) причины ошибок проекта рисунка 9.1 и исправить их.

Причина первой ошибки была в том, что метод вывода адреса был в режиме доступа private, из-за этого вызвать метод вывода адреса в классе программ не удалось. Нужно было изменить режим доступа private на любой другой. Вторая ошибка с атрибутом номера телефона также была связана с режимом доступа.

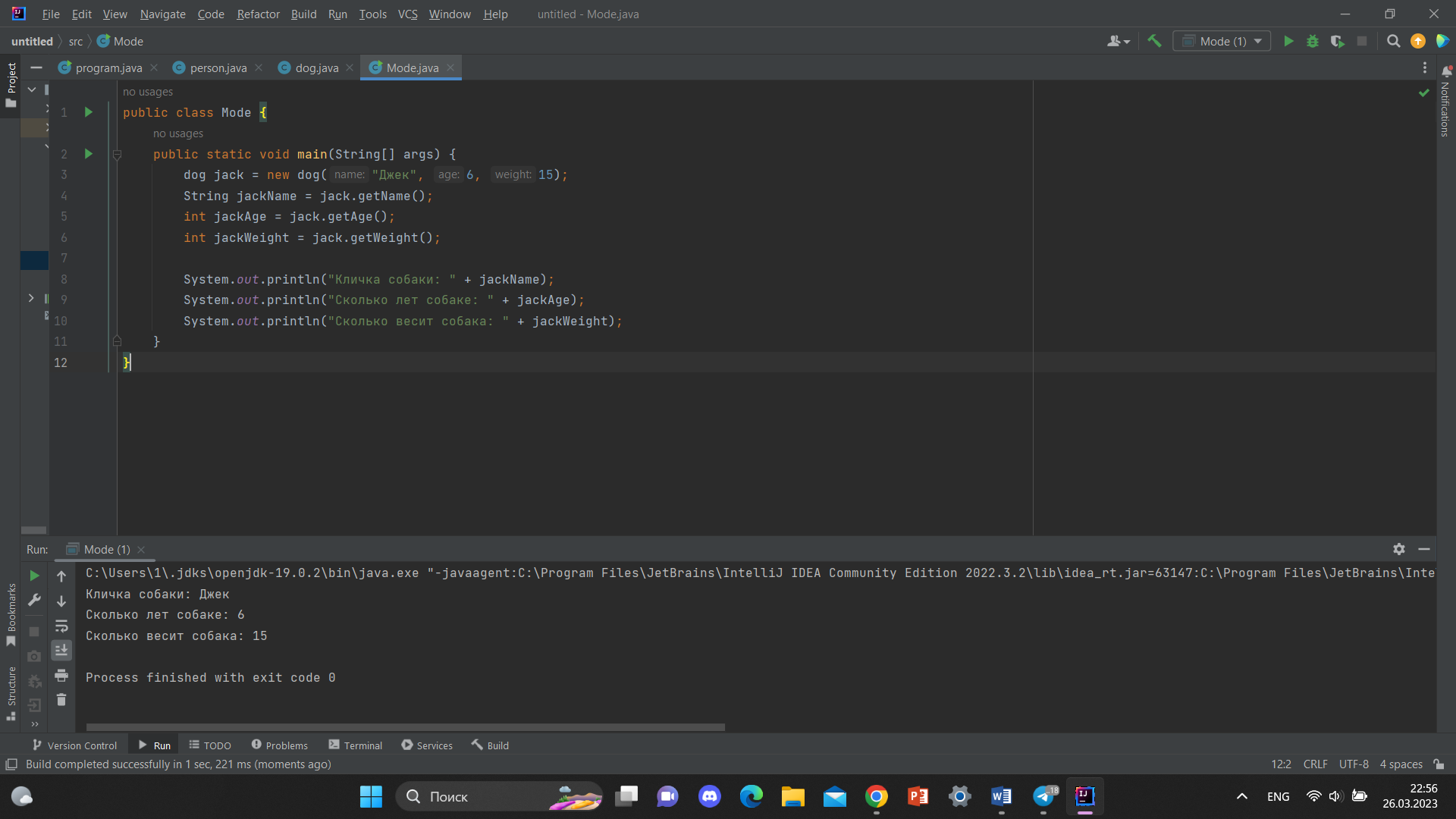
* 1. В проекте рисунка 9.2 объяснить работу кодов строчек 6-10 и добиться вывода другого значения возраста по методу **.setAge**.

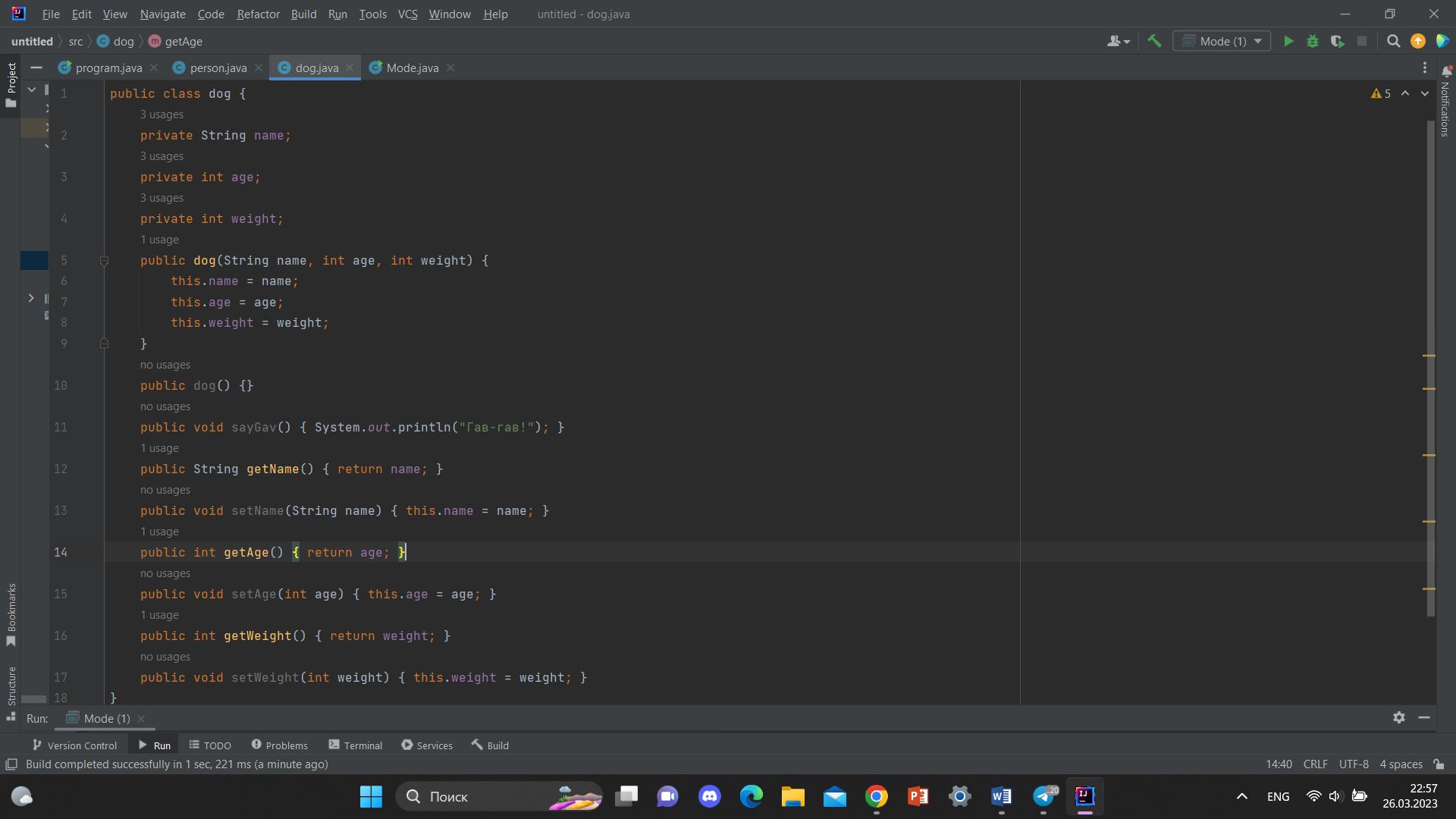




Сначала при помощи метода getAge мы выводим значение атрибута. Затем при помощи метода setAge мы изменим значение возраста и опять выводим значение возраста. Чтобы у нас второй раз изменилось значение возраста, атрибут должен подходить под условие, поэтому чтобы возраст изменился нужно просто изменить диапазон. Изменив с 0 до 110, на 0 – 11000000.

* 1. Разобраться в коде, приведённом в <https://3ty.ru/languages/java/gettery-i-settery-v-java> и реализовать его в соответствующем проекте





В этих программах создаются два класса dog and mode. Класс собак содержит приватные атрибуты с геттерами и сеттерами. В классе мод создается экземпляр класса собак и класс мод получает атрибуты класса собак с помощью методов геттеров.

* 1. Сформулировать преимущества и недостатки инкапсуляции

Преимущества:

1. Полный контроль над входящими и исходящими данными.
2. Пожалуй, основное преимущество связано с полиморфизмом подтипов. Если мы вызываем функцию, то это всегда некоторая конкретная функция, импортированная из конкретного пространства имён. А вот если мы вызываем метод, то появляются варианты. Когда интерпретатор доходит до кода с вызовом метода, он не может сразу сказать, где определён данный метод, потому что ответ на этот вопрос зависит от того, какой тип у obj. Отсюда следует, что, если разные объекты содержат методы с одинаковым именем (и сигнатурой), то их можно прозрачно (для вызывающего кода) подменять. На практике такая возможность местами упрощает код (становится меньше условных конструкций), но главное — делает его расширение проще.
3. Можно без боязни сломать все остальное приложение, править реализацию методов компонентов. Так как во всем остальном приложении фигурирует только интерфейс, а Вы меняете только реализацию логики.

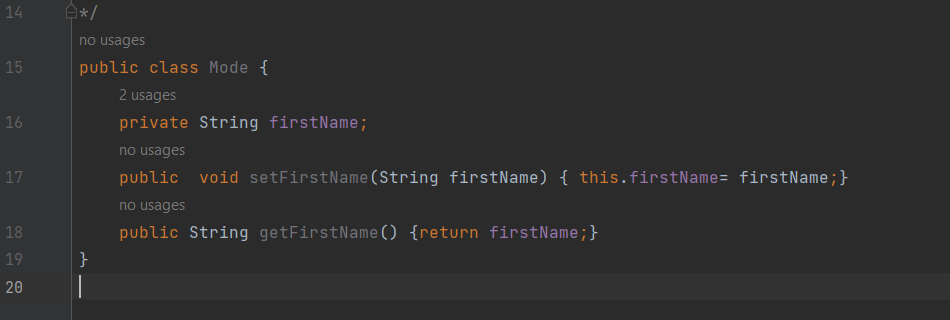
Недостатки:

1. Если Вы нашли ошибку, в библиотеке которую используете, то Вам будет трудно ее исправить.
2. Снижается скорость работы.

## Задание 9.2

1. Программно реализовать ошибки и их исправление из раздела 9.3.3;

Ошибка 1.



Ошибка 2.



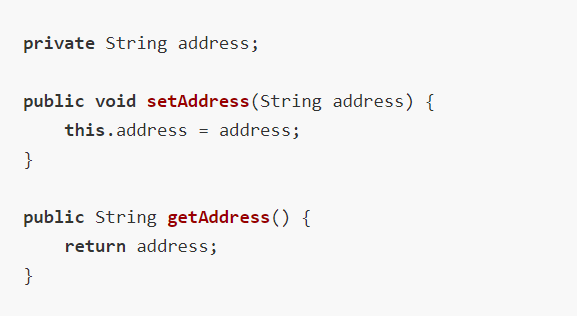
Ошибка 3.



1. Разобрать разделы 9.3.5.1 и 9.3.5.2

**Геттеры и сеттеры для String**

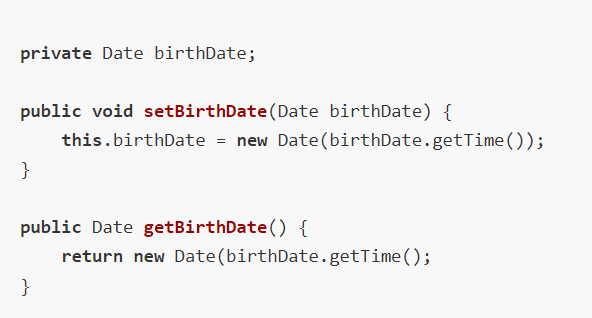
String — это immutable-тип. Это означает, что после создания объекта этого типа, его значение нельзя изменить. Любые изменения будут приводить к созданию нового объекта String. Таким образом, как и для примитивных типов, вы можете безопасно реализовать геттер и сеттер для переменной String:



**Геттеры и сеттеры для объектов типа Date**

Т.к. объекты класса *java.util.Date* являются изменяемыми, то внешние классы не должны иметь доступ к их оригиналам. Данный класс реализует метод *clone()* из класса *Object,*который возвращает копию объекта, но использовать его для этих целей не стоит.

По этому поводу Джошуа Блох пишет следующее: «Поскольку *Date*не является окончательным классом, нет га­рантии, что метод *clone()*возвратит объект, класс которого именно *java.util.Date*: он может вернуть экземпляр ненадежного подкласса, созданного специально для нанесения ущерба. Такой подкласс может, например, записы­вать ссылку на каждый экземпляр в момент создания последнего в закрытый статический список, а затем предоставить злоумышленнику доступ к этому списку. В результате злоумышленник получит полный контроль над всеми эк­земплярами копий. Чтобы предотвратить атаки такого рода, не используйте метод *clone()* для создания копии параметра, тип которого позволяет нена­дежным сторонам создавать подклассы».



1. Программно реализовать геттеры и сеттеры собственного класса с использованием метода clone().

