**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«РОССИЙСКАЯ АКАДЕМИЯ НАРОДНОГО ХОЗЯЙСТВА   
И ГОСУДАРСТВЕННОЙ СЛУЖБЫ**

**ПРИ ПРЕЗИДЕНТЕ РОССИЙСКОЙ ФЕДЕРАЦИИ»**

**НИЖЕГОРОДСКИЙ ИНСТИТУТ УПРАВЛЕНИЯ – филиал РАНХиГС**

Отчет по программированию – лекция №9

Выполнил: студент группы Иб-321

Наумов Александр Евгеньевич

**Оглавление**

[**Задание 1** 3](#_Toc130854176)

[**1.1 и 1.2** 3](#_Toc130854177)

[**1.3** 6](#_Toc130854178)

[**1.4** 7](#_Toc130854179)

[**1.5** 8](#_Toc130854180)

[**Задание 2** 9](#_Toc130854181)

[**1.1** 9](#_Toc130854182)

[**1.2** 10](#_Toc130854183)

[**Геттеры и сеттеры для String** 10](#_Toc130854184)

[**Геттеры и сеттеры для объектов типа Date** 10](#_Toc130854185)

[**1.3** 12](#_Toc130854186)

# **Задание 1**

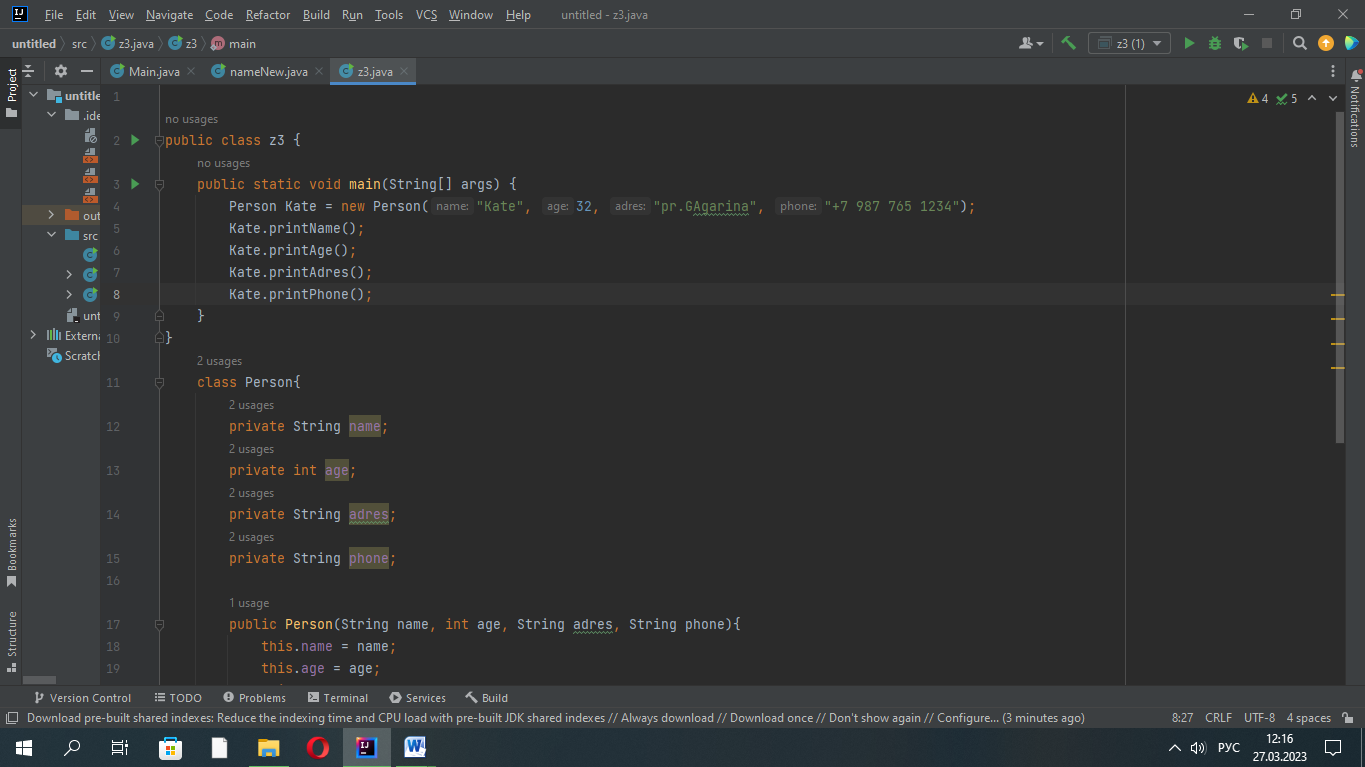
Текст задания

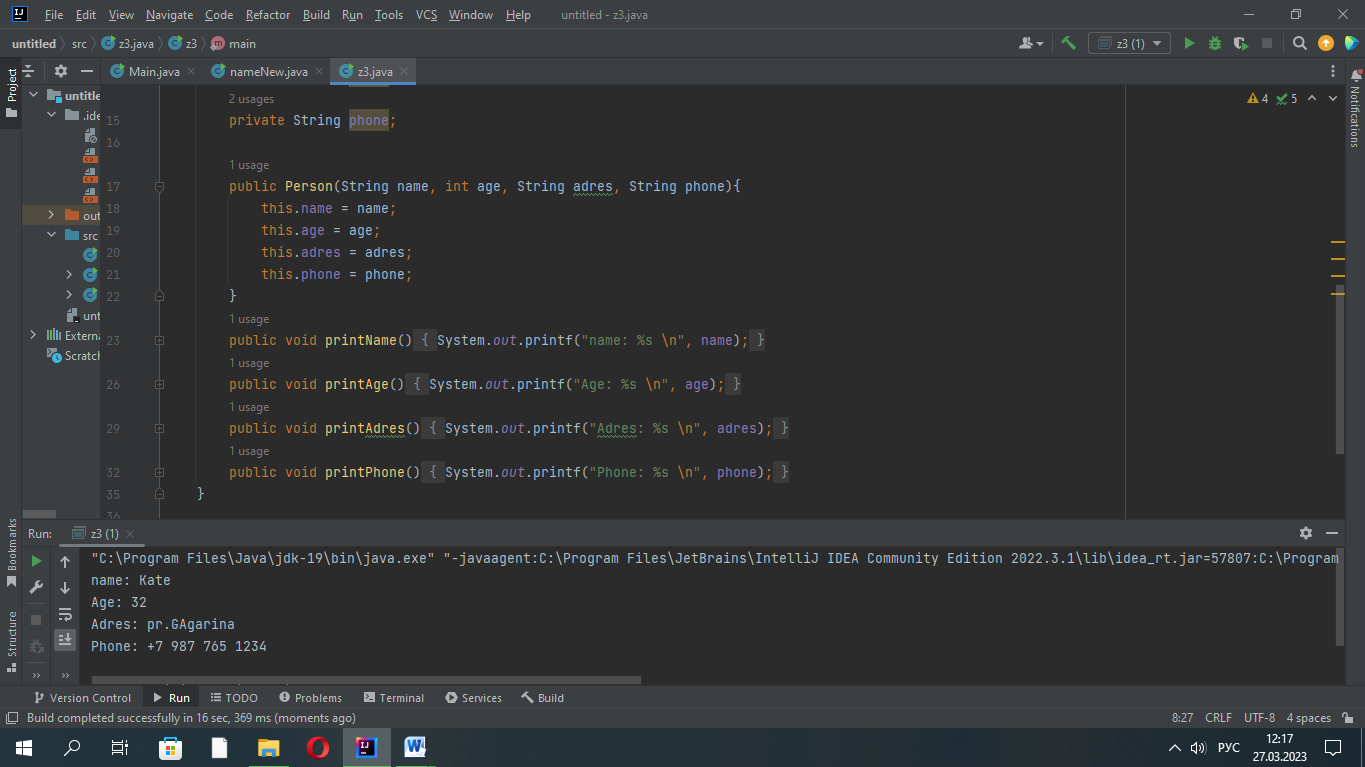
* 1. Реализовать проекты с программами рисунков 9.1 и 9.2.
  2. Определить (идентифицировать) причины ошибок проекта рисунка 9.1 и исправить их.
  3. В проекте рисунка 9.2 объяснить работу кодов строчек 6-10 и добиться вывода другого значения возраста по методу **.setAge**.
  4. Разобраться в коде, приведённом в <https://3ty.ru/languages/java/gettery-i-settery-v-java> и реализовать его в соответствующем проекте
  5. Сформулировать преимущества и недостатки инкапсуляции

## **1.1 и 1.2**

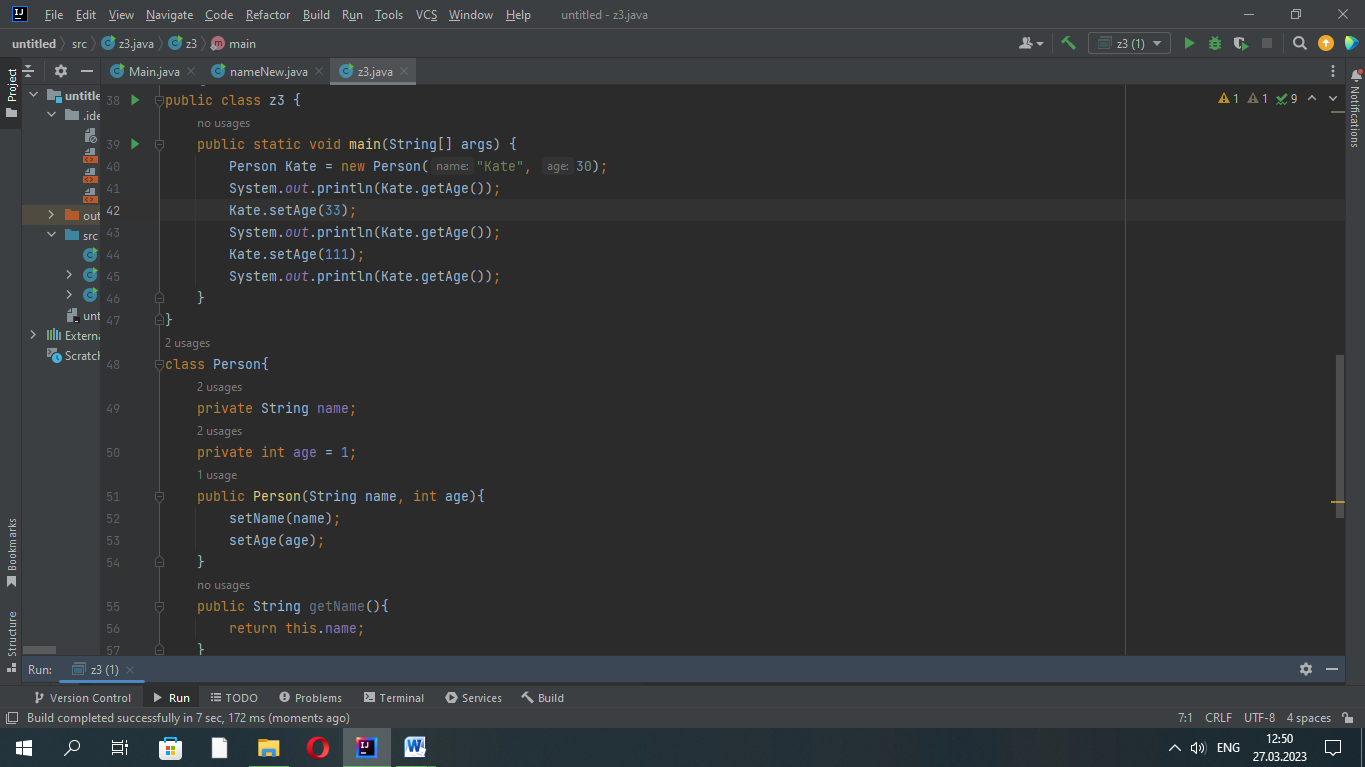
Код 9.1 исправленный, мы исправили:

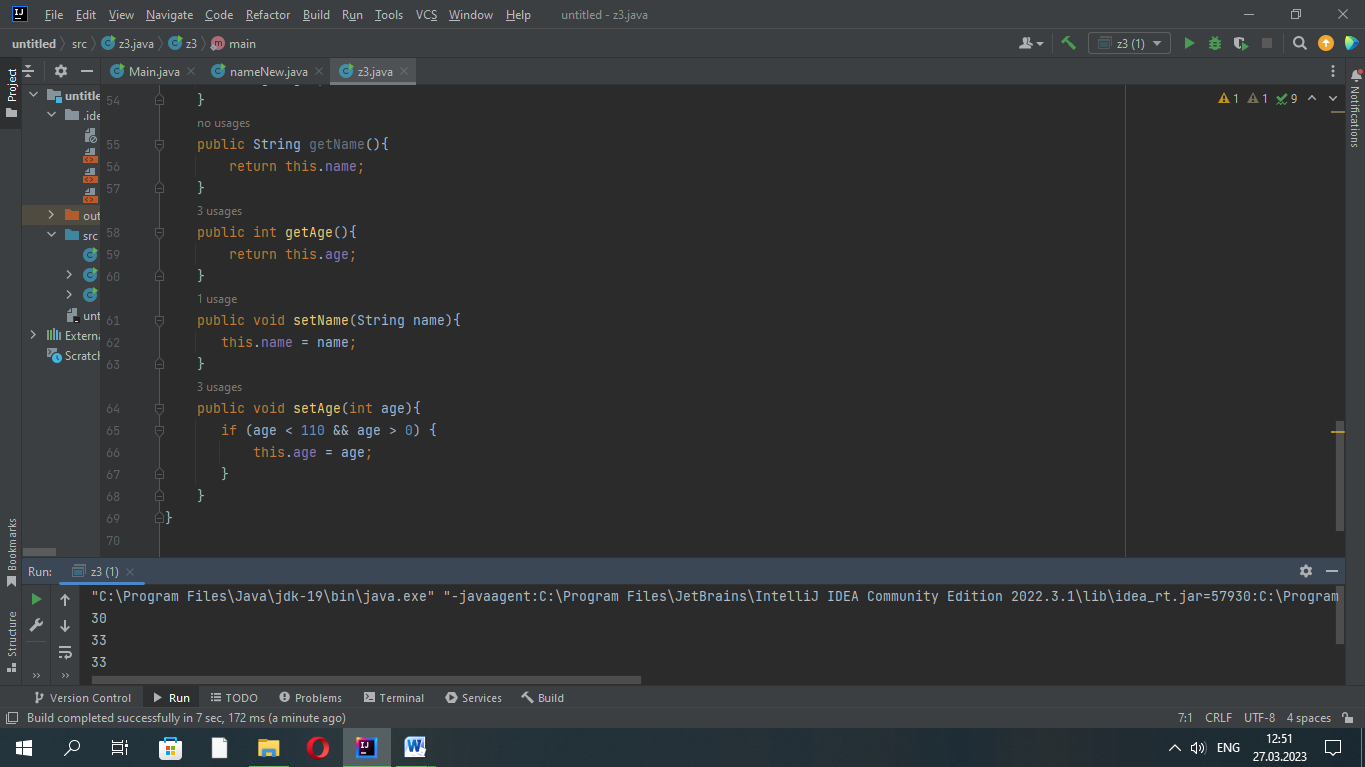
1. Все переменные класса должны быть приватными.
2. Все методы класса должны быть публичными или protected или дефолтными.





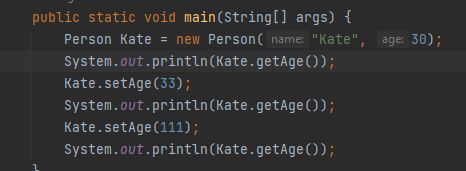
Код 9.2

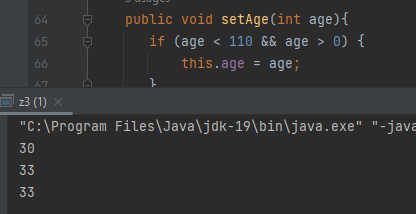




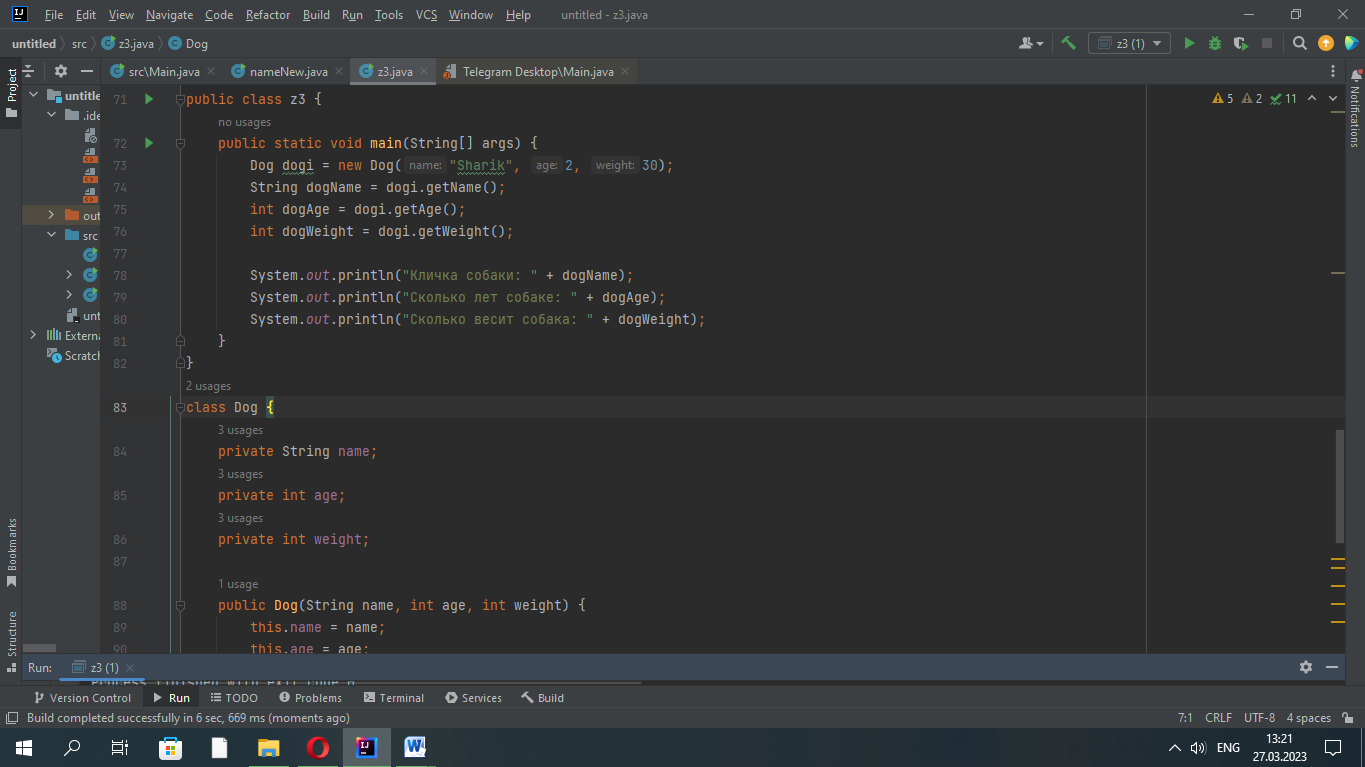
## **1.3**

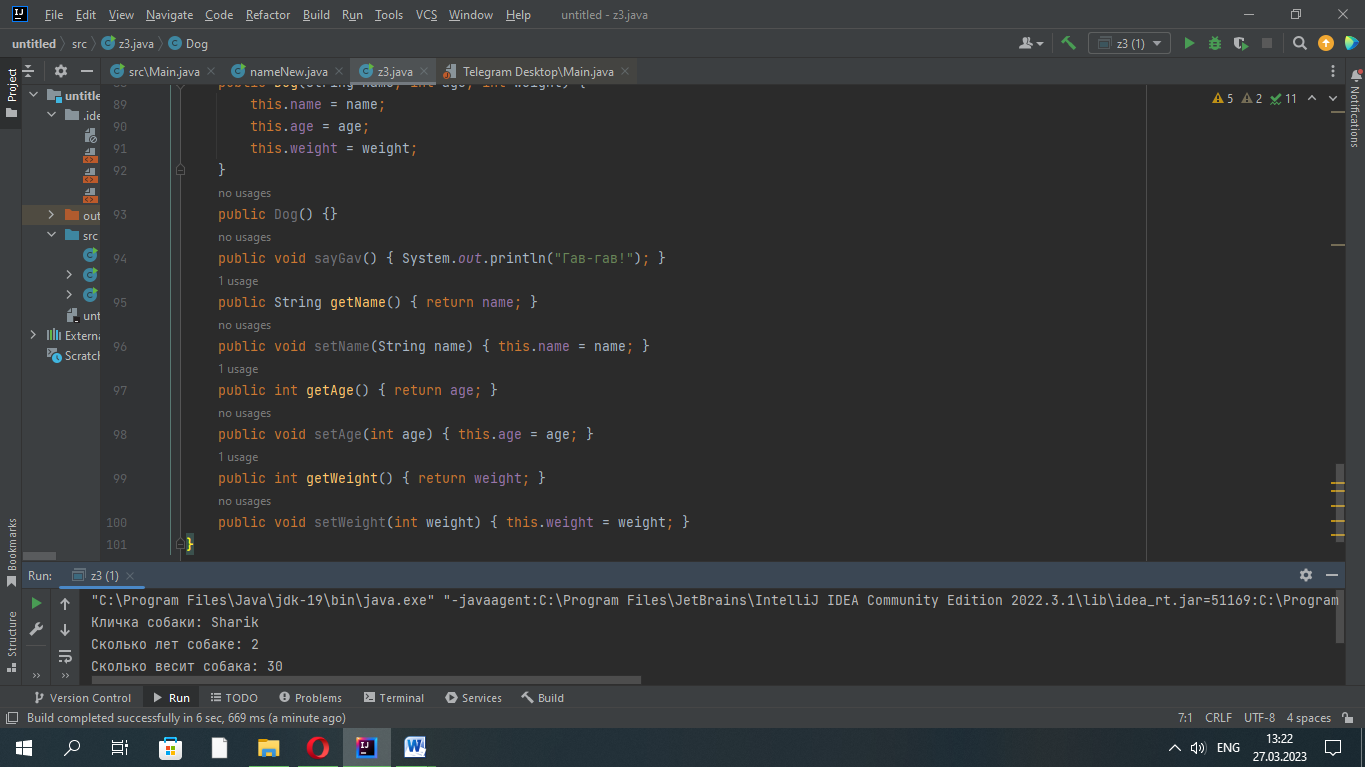
Строчки 6-10 устанавливают новые значения для переменной age объекта Kate. Перед добавлением возраста стоит ограничение от 1 до 109.





## **1.4**





Класс Dog ограничивает доступ к переменным name, age и weight. Но с помощью публичных геттеров и сеттеров возможно получать и изменять значения этих переменных.

## **1.5**

Инкапсуляция – это механизм объединения данных и методов в один класс, с возможностью определять для всех элементов уровень доступа.

Преимущества:

1. Читаемость кода и легкость внесения изменений.
2. Упрощает работу в большой команде.
3. Полный контроль над входящими и исходящими данными - предотвращает допущения ошибок и утечек данных.

Недостатки:

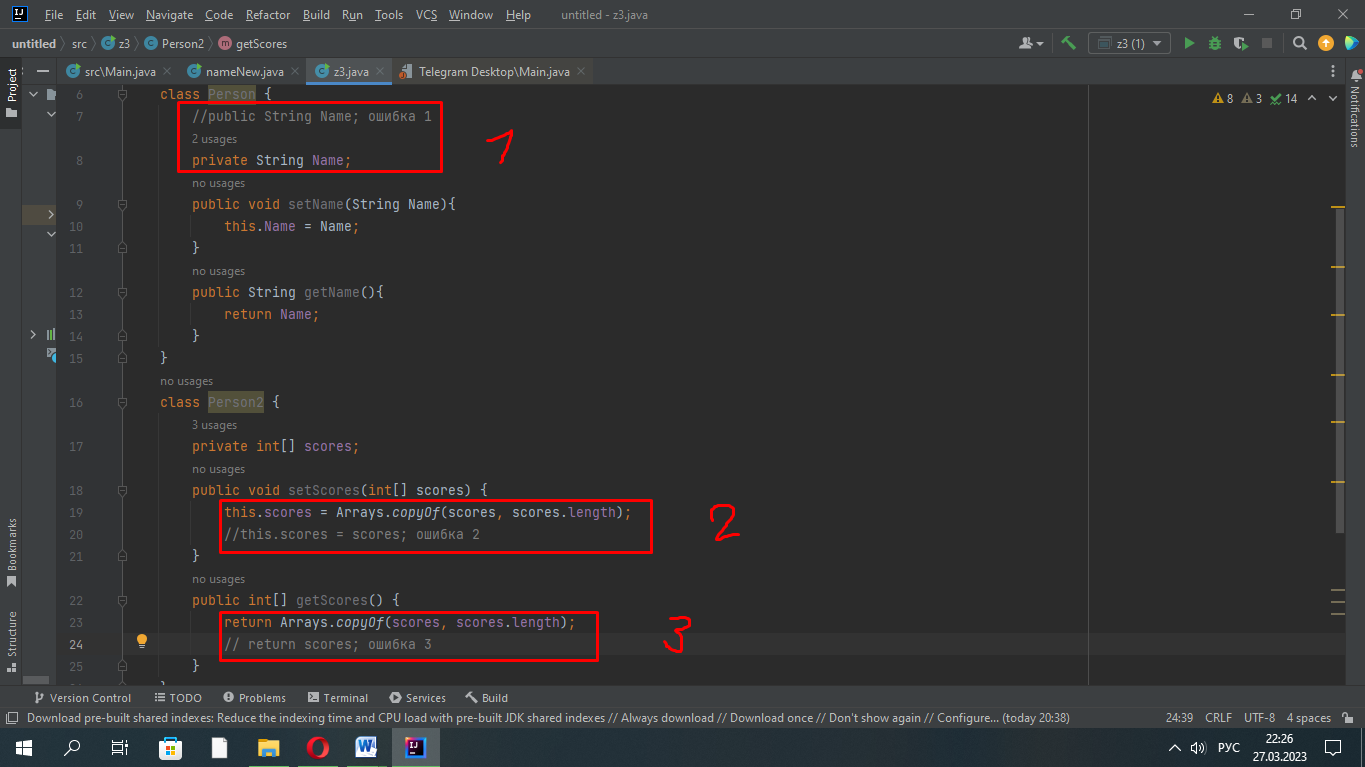
1. Программисты не доверяют чужим «черным ящикам».
2. В чужих «чёрных ящиках» случаются ошибки, которые приходится фиксить, влезая в их внутренности (и ломая этим всю затею).
3. Входы и выходы не всегда описаны понятно. Иногда бывает проще создать свой велосипед, чем разбираться в том, как поехать на чужом.

# **Задание 2**

Текст задания

1. Программно реализовать ошибки и их исправление из раздела 9.3.3;
2. Разобрать разделы 9.3.5.1 и 9.3.5.2
3. Программно реализовать геттеры и сеттеры собственного класса с использованием метода clone().

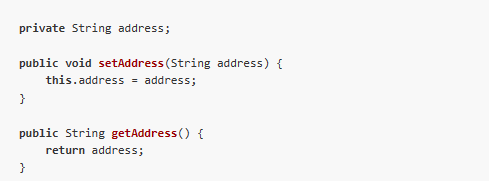
## **1.1**



## **1.2**

### **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, можно безопасно реализовать геттер и сеттер для переменной String:

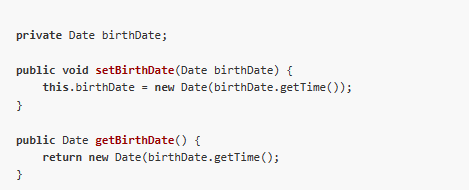


### **Геттеры и сеттеры для объектов типа Date**

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.

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 следует использовать метод clone() для создания копии параметра, тип которого позволяет нена­дежным сторонам создавать подклассы.

Чтобы это исправить следует каждый раз создавать новый экземпляр класса:



## **1.3**

Мы создали полный клон объекта store1.

