**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

Комышков Владислав Дмитриевич

**Оглавление**

[**Задание 1** 3](#_Toc130766663)

[**Пункт 1.1** 4](#_Toc130766664)

[**Пункт 1.2** 6](#_Toc130766665)

[**Пункт 1.3** 7](#_Toc130766666)

[**Пункт 1.4** 9](#_Toc130766667)

[**Пункт 1.5** 10](#_Toc130766668)

[**Задание 2** 11](#_Toc130766669)

[**Пункт 1.1** 12](#_Toc130766670)

[**Пункт 1.2** 14](#_Toc130766671)

[**Геттеры и сеттеры для String** 14](#_Toc130766672)

[**Геттеры и сеттеры для объектов типа Date** 14](#_Toc130766673)

[**Пункт 1.3** 16](#_Toc130766674)

# **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

## **Пункт 1.1**

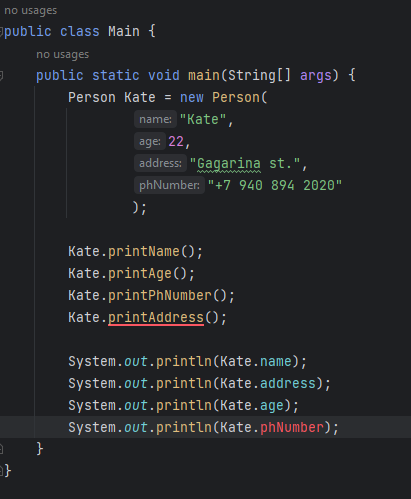


Рисунок 1.1 – скрипт Main рисунка 9.1



Рисунок 1.2 – скрипт Person рисунка 9.1

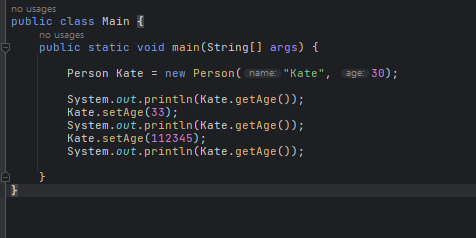


Рисунок 1.3 – скрипт Main рисунка 9.2

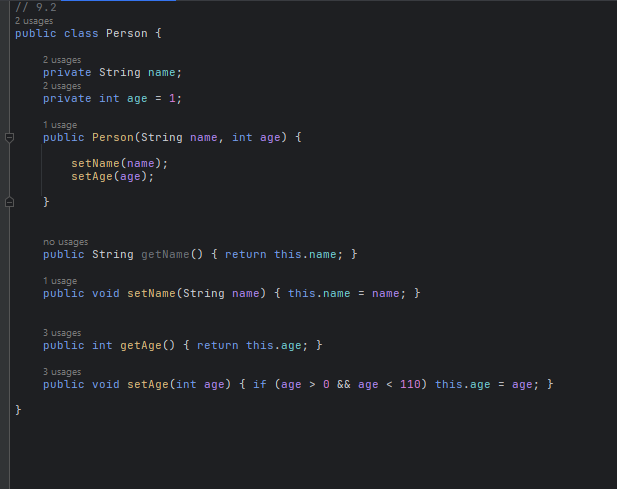


Рисунок 1.4 – скрипт Person рисунка 9.2

## **Пункт 1.2**

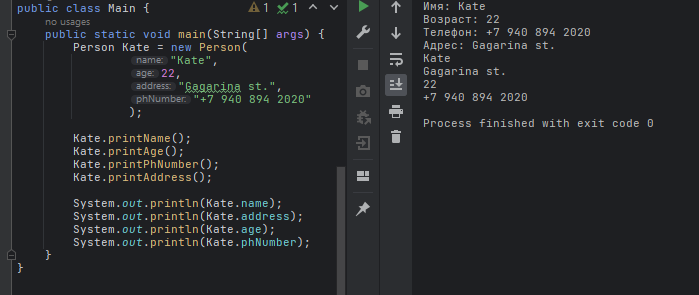


Рисунок 1.5 – скрипт Main рисунка 9.1 и вывод в консоль

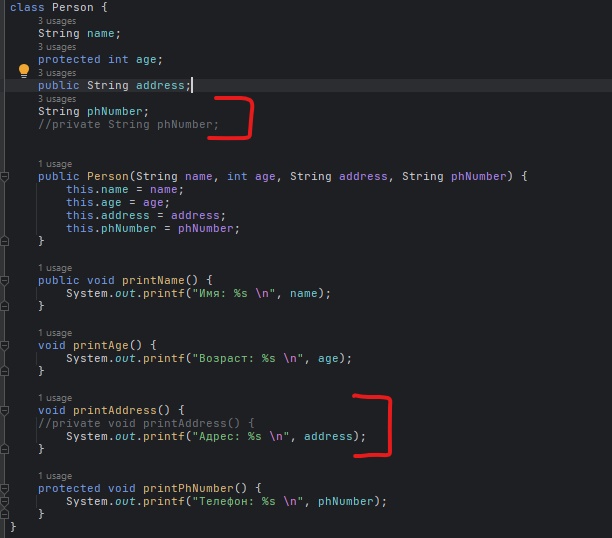


Рисунок 1.6 – исправленный скрипт Person рисунка 9.1

## **Пункт 1.3**

Строчки 6-10 (рисунок 1.7) устанавливают новые значения для переменной age объекта Kate. Так как на age стоит ограничение, по условию от 0 до 110, то значение не меняется.

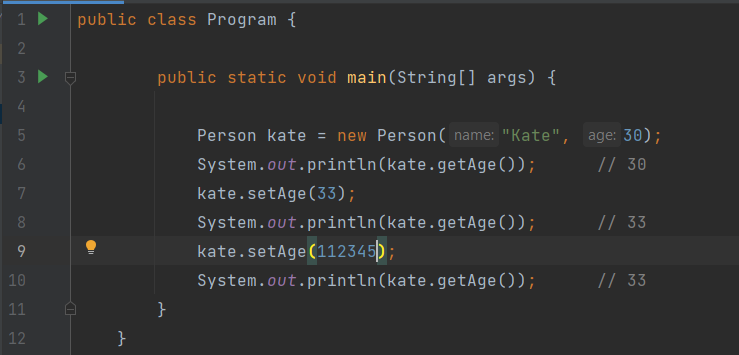


Рисунок 1.7 – скрипт Program рисунка 9.2

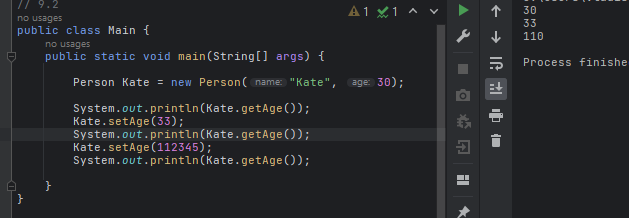


Рисунок 1.8 – скрипт Main рисунка 9.2 и вывод в консоль

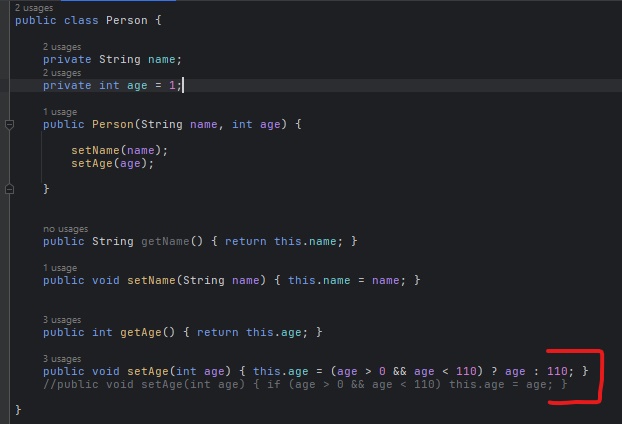


Рисунок 1.9 – исправленный скрипт Person рисунка 9.2

## **Пункт 1.4**

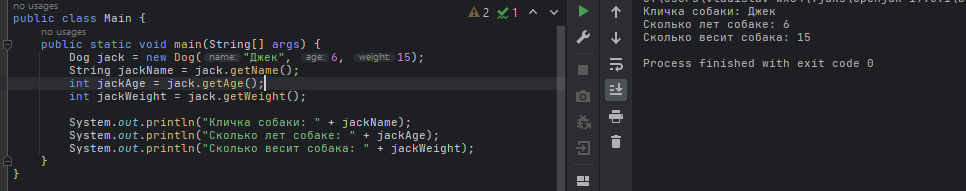


Рисунок 1.10 – скрипт Main

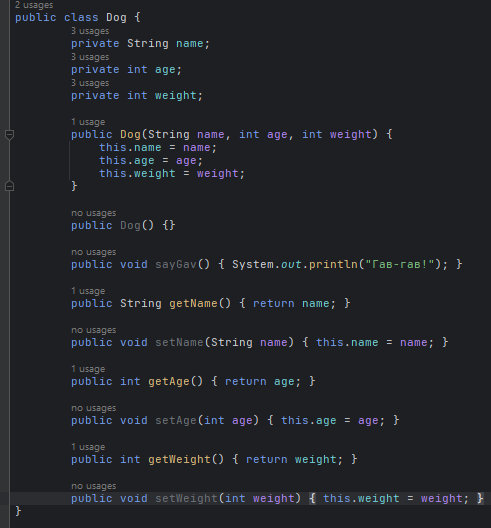


Рисунок 1.11 – скрипт Dog

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

## **Пункт 1.5**

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

Преимущества:

* Читаемость кода и легкость внесения изменений - упрощает работу в большой команде.
* Полный контроль над входящими и исходящими данными - предотвращает допущения ошибок и утечек данных.

Недостатки:

* Сложность использования чужих решений. Чужие методы могут приводить к багам или не приводить к желаемому результату.

# **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().

## **Пункт 1.1**

Ошибка №1

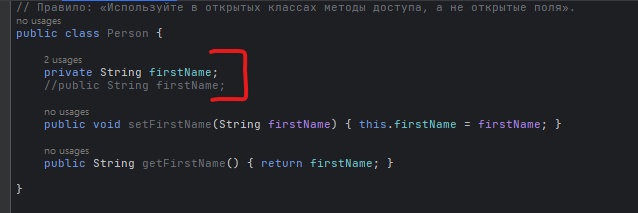


Рисунок 2.1 – исправленный скрипт Person

Ошибка №2

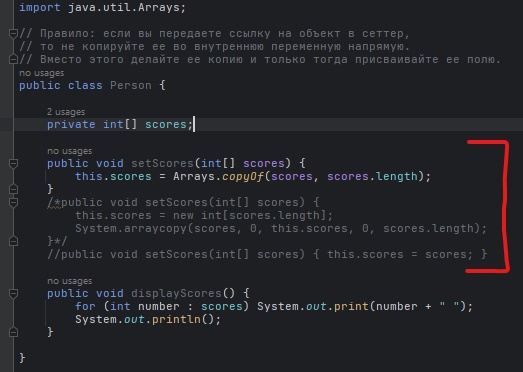


Рисунок 2.2 – исправленный скрипт Person

Ошибка №3

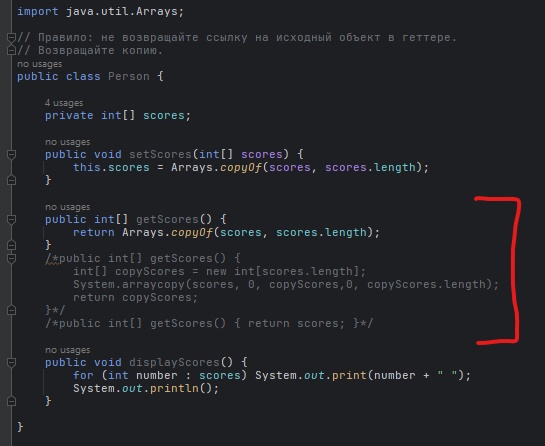


Рисунок 2.3 – исправленный скрипт Person

## **Пункт 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:

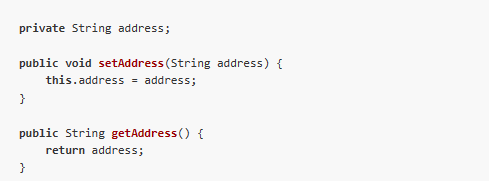


Рисунок 2.4 – геттеры и сеттеры для String

### **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:

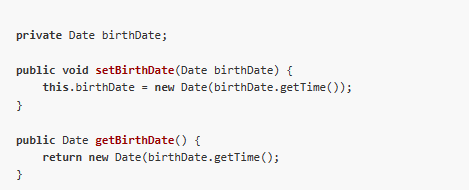


Рисунок 2.5 – геттеры и сеттеры для объектов типа Date

## **Пункт 1.3**

Собственный класс Vector2D (рисунок 2.7) с использованием метода clone().

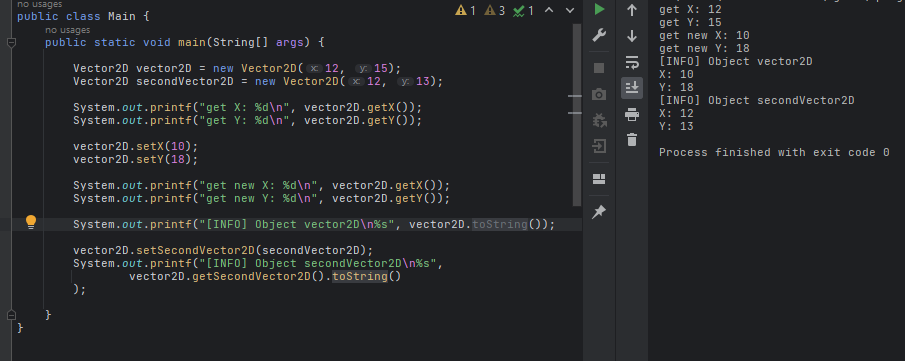


Рисунок 2.6 – скрипт Main и вывод в консоль



Рисунок 2.7 – скрипт Vector2D