|  |  |  |
| --- | --- | --- |
| Министерство науки и высшего образования РФ Федеральное государственное автономное образовательное учреждение высшего образования  «Пермский государственный национальный исследовательский университет» | | |
|  | Институт компьютерных наук и технологий | |
| **ОТЧЁТ**  по лабораторной работе №6  по дисциплине «Языки программирования»  Вариант 4 | | |
|  | | Работу выполнил  студент группы ПМИ-7,8-2023 2 курса  Семенов И.Н.  «\_\_\_» \_\_\_\_\_\_\_\_2024 г. |
| Работу проверил  Ракина В.Д  «\_\_\_» \_\_\_\_\_\_\_\_\_ 2024 г. |
| Пермь 2024 | | |

СОДЕРЖАНИЕ

[Министерство науки и высшего образования РФ 1](#_Toc185894574)

[Задание 1 4](#_Toc185894575)

[Текст задания 4](#_Toc185894576)

[Алгоритм решения 4](#_Toc185894577)

[Тестирование 4](#_Toc185894578)

[Код программы 5](#_Toc185894580)

[Задание 2 5](#_Toc185894581)

[Текст задания 5](#_Toc185894582)

[.Алгоритм решения 6](#_Toc185894589)

[Тестирование 7](#_Toc185894590)

[Код программы 7](#_Toc185894591)

[Задание 3 8](#_Toc185894592)

[Алгоритм решения 8](#_Toc185894593)

[Тестирование 8](#_Toc185894594)

[Код программы 9](#_Toc185894595)

[Задание 4 10](#_Toc185894596)

[Текст задания 10](#_Toc185894597)

[Алгоритм решения 10](#_Toc185894599)

[Тестирование 11](#_Toc185894600)

[Код программы 11](#_Toc185894601)

# Задание 1

## Текст задания

К строке.

Разработайте класс Entity. Данный класс не является самостоятельной сущностью и предназначен для наследования. Необходимо переопределить его метод приведения к строке так, чтобы поля всех классов-наследников автоматически попадали в строковую форму объекта в виде пар ключ-значение. На рисунке дан пример результата для класса A.

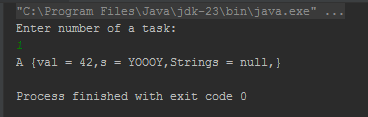
## Алгоритм решения

Переопределяем метод toString. С помощью рефлексии получаем список всех инициализированных полей класса (getDeclaredFields). Создаем мапу для хранения результатов, которая будет хранить пары ключ(строка)-значение(объект).

Проходим по всем полям класса, разрешаем доступ к приватным полям, получаем имя и значение поля, размещаем в мапу.С помощью стрингбилдера собираем результирующую строку, проходя по мапе.

## Тестирование

## C:\Users\Илья\Desktop\lab6\1.2.png



## Код программы

https://github.com/SemenovIlya12/Jab/tree/Lab6

# Задание 2

## Текст задания

## @Invoke.

## Разработайте аннотацию @Invoke, со следующими характеристиками:

## Целью может быть только МЕТОД

## Доступна во время исполнения программы

## Не имеет свойств

## Проаннотируйте какой-либо метод данной аннотацией

@Default.

Разработайте аннотацию @Default, со следующими характеристиками:

* Целью может быть ТИП или ПОЛЕ
* Доступна во время исполнения программы
* Имеет обязательное свойство value типа Class

Проаннотируйте какой-либо класс данной аннотацией.

@ToString.

Разработайте аннотацию @ToString, со следующими характеристиками:

* Целью может быть ТИП или ПОЛЕ
* Доступна во время исполнения программы
* Имеет необязательное свойство valuec двумя вариантами значений: YES или NO
* Значение свойства по умолчанию: YES

Проаннотируйте какой-либо класс данной аннотацией.

@Validate.

Разработайте аннотацию @Validate, со следующими характеристиками:

* Целью может быть ТИП или АННОТАЦИЯ
* Доступна во время исполнения программы
* Имеет обязательное свойство value, типа Class[]

Проаннотируйте какой-либо класс данной аннотацией.

@Two.

Разработайте аннотацию @Two, со следующими характеристиками:

* Целью может быть ТИП
* Доступна во время исполнения программы
* Имеет два обязательных свойства: first типа String и second типа int

Проаннотируйте какой-либо класс данной аннотацией.

@Cache.

Разработайте аннотацию @Cache, со следующими характеристиками:

* Целью может быть ТИП
* Доступна во время исполнения программы
* Имеет необязательное свойство value, типа String[]
* Значение свойства по умолчанию: пустой массив

Проаннотируйте какой-либо класс данной аннотацией.

## .Алгоритм решения

Создаем аннотации, для этого создадим класс с типом @interface.

Применяем необходимые свойства для нашего аннотационного класса с помощью

@Retention(RetentionPolicy.RUNTIME) - обеспечивает доступность во время работы программы

@Target(…) – указывает цели аннотации

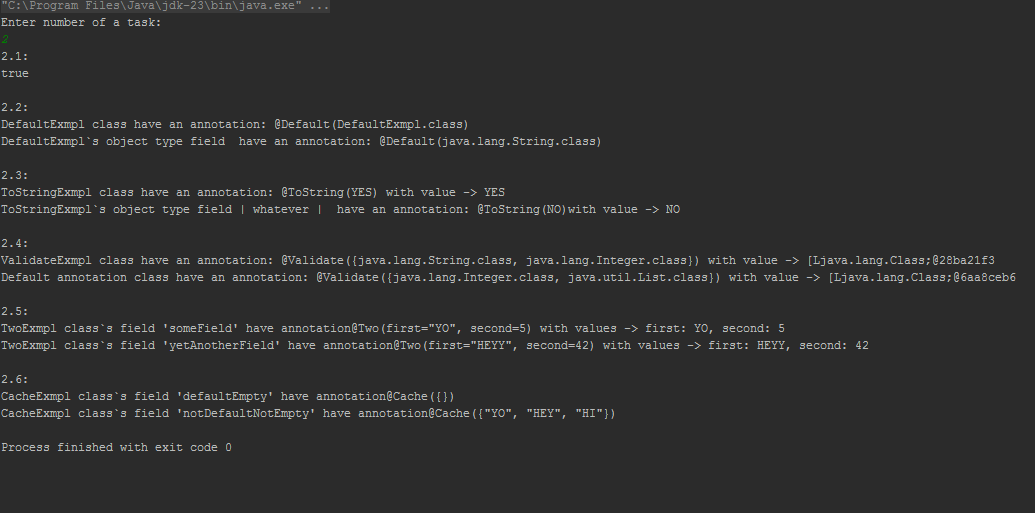
* ElementType.Method – методы
* ElementType.Field – поля
* ElementType.Type – классы, интерфейсы, перечисления, аннотации

Устанавливаем свойства, обязательные/необязательные в теле класса-аннотации.

Для того, чтобы сделать свойство необязательным, необходимо установить значение по умолчанию.

После, аннотируем метод/класс/поле нашей аннотацией и проверяем их наличие.

## Тестирование



## Код программы

Ссылка на githab

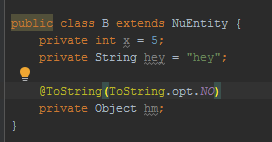
# Задание 3

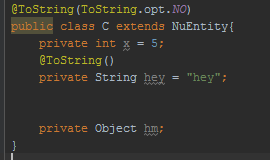
К строке. Измените класс Entity из задачи 1.1. Метод приведения к строке необходимо переписать таким образом, что бы в строковую форму попадали только те поля, которые не помечены аннотацией @ToString (из предыдущей задачи) со значением NO. Поле помеченное как @ToString(YES) или же не помеченное никак должно попадать в строковое представление. В случае если с помощью @ToString(NO) проаннотирован весь класс, то ни одно из его полей не попадает в строковое представление, кроме тех, которые явно помечены как @ToString(YES).

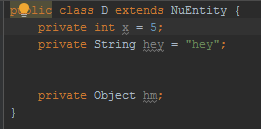
## Алгоритм решения

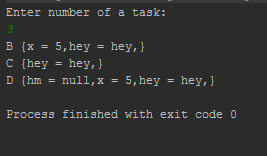
Получаем аннотацию ToString на уровне класса, если аннотация есть и ее значение равно NO, устанавливаем флаг exclude на true. В цикле для каждого поля получаем аннотацию на уровне поля, инициализируем флаг include – он будет решать, включать ли поле в строковое представление или нет. Include будет иметь значение тернарного оператора, если флаг exclude true – тогда включаем только те поля, которые имеют значение поля YES, если флаг exclude false – тогда исключаем поля с значением NO.

## Тестирование









## Код программы

Ссылка на githab

# Задание 4

## Текст задания

## Напишите тест, используя фреймворк juntil, к методу из задания 3. Пояснения к тесту и объекту тестирования отобразить в комментариях кода и в отчете

## Алгоритм решения

Создаем отдельный класс для тестирования. в котором у нас будут хранится сами тесты. Сделаем несколько схожих экземпляров классов, на котором будем проводить тесты, они будут иметь одинаковые поля, но разные аннотации: B – поле hm аннотировано с значением NO, С - Весь класс аннотирован с значением NO, поле hey аннотировано с значением по умолчанию. D – обычный класс без аннотаций.

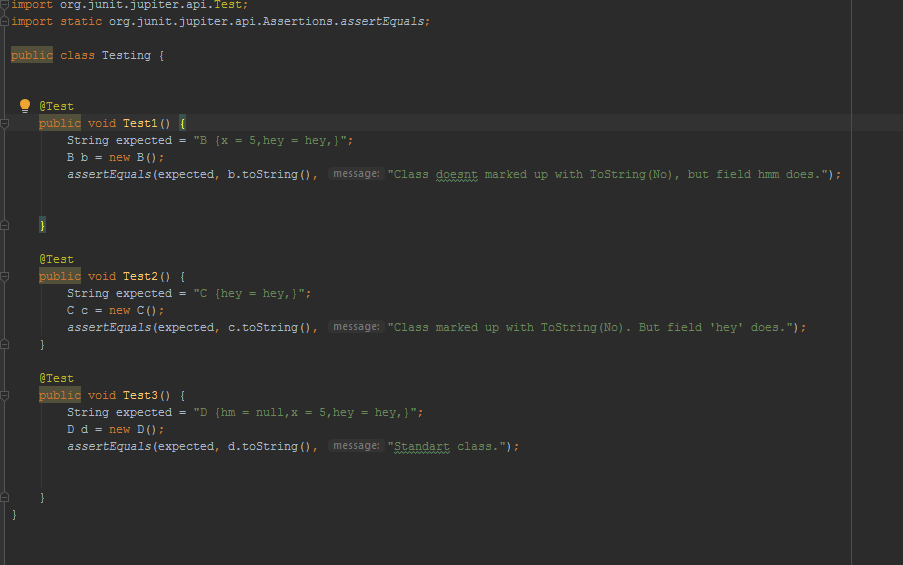
Тесты создаются с помощью аннотации @Test.

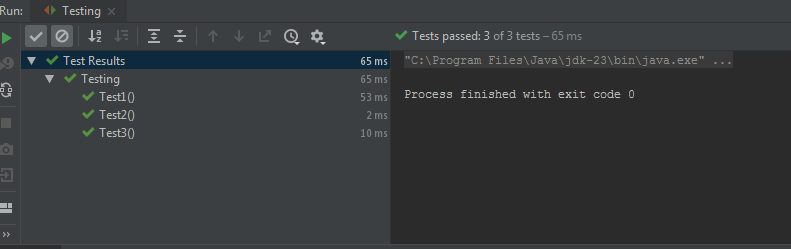
Проверка осуществляется посредством сравнения ожидаемого результата с полученным в итоге тестирования. Используем assertEquals – метод принимает ожидаемый результат – в нашем случае это результирующая строка, и сам метод, который подвергается проверке – в нашем случае, это метод toString.

Тесты:

1. Ожидаемый результат: "B {x = 5,hey = hey,}", метод b.toString()
2. Ожидаемый результат: “C {hey = hey,}", метод c.toString()
3. Ожидаемый результат: "D {hm = null,x = 5,hey = hey,}", метод d.toString()

## Тестирование





## Код программы

Ссылка на githab