**Java 2020 practice**

Если кому-то нужно почитать про Git

Книжка по Git:

<https://git-scm.com/book/ru/v2>

Практика Git (интерактивные уроки):

<http://gitimmersion.com/lab_01.html>

<https://learngitbranching.js.org/>

**Домашнее задание 1**

1. Скачать и настроить JDK
2. Написать программу, в которой есть две переменные типа Integer со значениями. Значения можно указать в коде программы или ввести с клавиатуры или взять из аргументов.
3. Если первое число больше второго – написать на экран «Число %s больше %s».
4. Если первое число меньше второго – написать на экран «Число %s меньше %s».
5. В любом случае, вывести на экран сумму чисел.

**Домашнее задание 2**

Задание 1. Случайности складываются  
- Создать массив на 10 чисел типа int  
- Заполнить массив случайными числами, используя класс Random и функцию nextInt()  
- Вывести массив на экран одной строкой, разделив элементы пробелом  
- Посчитать сумму всех элементов и вывести на экран  
- Найти минимальный элемент и вывести на экран его индекс в массиве и значение

Задание 2. Случайности расходятся  
- Создать массив на 5 элементов типа int и вывести на экран  
- Разделить на два других - в первом массиве только четные числа, во втором - только нечетные. Оба массива вывести на экран.  
- Найти среднее значение в каждом массива и вывести их на экран

Задание 3. Противоположности притигиваются  
- Создайте два массива на 7 элементов.  
- Используюте функцию Random.nextInt() чтобы полностью заполнить два массива: в первом массиве должно оказаться ровно 7 положительных чисел, во втором - ровно 7 отрицательных.  
Подсказка: используйте цикл while во время генерации случайных чисел и проверку на заполненность массива, прежде, чем записывать туда элемент.  
- Выведите оба массива на экран  
- Создайте новый массив на 7 элементов.  
Значением на каждой позиции является суммой значений на позициях с тем же индексом из положительного и отрицательного массива.

**Домашнее задание 3**

Задание 1. Цифровая персонализация  
- Создать класс Person с двумя полями типа String: firstName, lastName  
- создать один публичный конструктор с аргументами firstName и lastName. В конструкторе инициализировать поля класса значениями из аргументов.  
- Для каждого поля определить только getter'ы (без setter'ов) - getFirstName, getLastName , который возвращают соответствующее поле  
- Создать метод toString() который возвращает человека в формате "Фамилия, Имя"

 в методе main  
- Создать массив типа String на 5 элементов - туда записать какие-нибудь реальные имена.  
- Создать массив типа String на 6 элементов - туда записать какие-нибудь реальные фамилии  
- Создать массив типа Person на 50 элементов - туда записать людей со случайными фамилиями из массивов выше  
- Вывести всех людей из массива в формате "Человек №{индекс в массиве} - Фамилия, Имя"

Задание 2. Случайности - не случайны  
Создать класс SequentialRandom с одним полем типа int.  
Значение поле можно только прочитать, но не изменить (используйте модифиактор private на поле и создайте только getter)  
Значение этого int вычисляется по следующим правилом:  
- Значение для первого созданного объекта - случайное  
- значение для каждого следующего на 1 больше, чем предыдущее  
- У класса есть статический метод resetRandom(), который делает значение следующего созданного объекта - случайным.

Например  
new SequentialRandom() // содержит 5 (случайное)  
new SequentialRandom() // содержит 6 (на 1 больше)  
new SequentialRandom() // содержит 7 (на 1 больше)  
SequentialRandom.resetRandom()  
new SequentialRandom() // содержит 42 (случайное  
new SequentialRandom() // содержит 43) (на 1  больше)

Подсказка: Используйте статическую переменную в классе, чтобы хранить значение предыдущего объекта.

**Домашнее задание 4**

1. Фруктовая неоднозначность  
Создать класс Fruit со свойством "вес" типа int и методом toString  
Создать наследники:  
Apple - с дополнительным свойством цвет (красное, зеленое, желтое) - String  
Orange - с дополнительном свойством толщина кожуры - int от 0 до 20  
Pineapple - с дополнительным свойством - высота хвоста - int, от 5 до 20

переопределить во всех фруктах toString, чтобы выводил все свойства фрукта и тип фрукта (яблоко, апельсин, ананас)

создать метод getRandomFruit - которое возвращает случаный фрукт со случайными характеристиками.  
Создать массив типа Fruit, заполнить его 4 разными случаными фруктами и вывести их на экран

2. Фруктовая абстрактность  
Создать класс AbstractFabric с одним абстрактным методом makeFruit(), который возвращает фрукт.  
Создайте следующие подклассы:

ThinOrangeFactory - которая создает  только апельсины с толщиной кожуры от 1 до 5  
RedApplesFactory - которая создает только красные яблоки  
GreenApplesFactory - которая создает только зеленые яблоки.   
LongPinepplesFactory - которая создает только ананасы с длиной хвоста больше 10

Создайте массив на 10 элементов и поместите туда 10 случайных фабрик одного из 4 описанных выше типов.   
Создайте массив типа Fruit на 20 элементов и заполните его фруктами из всех  фабрик - по 2 фрукта от одной фабрики.

**Домашнее задание 5**

Перечисления, интерфейсы и исключения

1. Абстрактный супрематизм

Создать абстрактный класс Figure, описывающий фигуру на двумерной плоскости с декартовыми координатами центра x,y.

В классе объявить абстрактный метод square, который вычисляет площадь фигуры.

В классе объявить и реализовать метод getQuadrant(), который возвращает четверть на координатной плоскости, в которой лежит центр фигуры.

Для описания четверти используйте Enum с четырьями значениями: I,II,III,IV

Создать классы-наследники:

Circle - дополнительный параметр радиус, от 0 до 10

Rectangle - дополнительные параметры ширина и высота, от 0 до 20.

В каждом классе определить метод square, чтобы он правильно считал площадь.

в main'e создать массив на 5 элементов типа Figure, заполните любыми фигурами (не обязательно случайными), выведите все фигуры в формате

"Название фигуры - площадь"

2. Сезонные радости

Создайте enum Season с 4 временами года.

Создайте enum Games с 4 играми.

В каждое время года можно играть ровно в одну игру:

Зима - снежки

Весна - запуск корабликов по ручьям

Лето - собирание и поедание ягод

Осень - измерение глубины луж методом "сапога"

Создайте класс Person. У него есть имя и любимый сезон.

Создайте массив на 10 элементов с людьми со случайными именами и любимыми сезонами.

Для каждого человека выведите любимую игру.

Рекомендации:

в Season определите дополнительное поле - игра для сезона.

**Домашнее задание 6**

Дженерики и коллекции

(пожалуйста, делайте каждое задание в отдельном package'e)

Задание 1. Велосипеды и контейнеры

Создать свой класс MyArrayList, который хранит максимум 10 объектов любого типа (по аналогии с ArrayList).

Если попытаться добавить больше, чем 10 элементов, то должно выкинуться unchecked исключение - MyArrayStoreException.

Используйте обощенный класс для этого (generic) с типом T.

- Реализуйте методы:

1. void add(T) - добавляет элемент в коллекцию

2. boolean remove(T) - возвращает true, если элемент был найден и удален

3. T[] toArray() - возвращает массив из всех текущих элементов

4. int size()

5. T get(int) - взять элемент по индексу.

- Создайте класс Bycicle, определите у него свойства String modelName; int maxSpeed (от 5 до 50км\ч);

Переопределите equals и hashCode, который используют оба свойства велосипеда.

- Создайте 11 случайных велосипедов и попытайтесь их добавить их в новый экземпляр MyArrayList.

Если какой-то велосипед добавить не получилось - то в лог выведите какой велосипед не добавился.

- Проверьте, что все методы работают правильно:

1. Возьмите случайное число от 0 до 50.

2. Удалите из коллекции все велосипеды со скоростью меньше этого случайного числа.

- Теперь создайте новый экземпляр MyArrayList

- Добавьте в него 10 случайных чисел от 1 до 10

- удалите все четные числа

- Выведите оставшиеся числа на экран.

**Домашнее задание 7**

Дженерики и коллекции

1. Коллекционный зоопарк

- Создать абстрактный класс Animal с полем для имени (name - String) и абстрактным методом getFavoriteMeal() - который возвращает любимую еду животного.

- Создать два класса наследника от Animal: WildAnimal и PetAnimal: дикие и домашние животные

- Создать классы для конкретных животных: кошка, собака, волк, лиса. Реализовать метод getFavoriteMeal().

Создать класс Zoo, который состоит из двух коллекций уникальных животных:

Set<WildAnimal> и Set<PetAnimal>

уникальность животного определяется его именем и классом: собака и волк могут быть с одинаковым именем, но двух волков с одинаковым именем быть не может.

- В классе создать два метода для заполнения коллекций животными:

1. метод fillCollectionWithPetAnimals на вход принимает коллекцию домашних животных и добавляет в нее случайное (от 1 до 5) количество кошек или собак.

Тип животного тоже выбирается случайно.

2. метод fillCollectionWithWildAnimals на вход принимает коллекцию диких животных и добавляет в нее случайное (от 3 до 10) количество лис или волков.

Вызвать оба этим метода в конструкторе для заполнения зоопарка.

- создать приватный метод printAnimalsFromCollection, который на вход принимает коллекцию животных и печатает их имена на экран

- создать публичный метод printAllAnimals, который печатает на экран из коллекций с домашними и дикими животными, использую приватный метод

printAnimalsFromCollection

**Домашнее задание 8**

Аннотации и рефлексия

Аннотированная де\сериализация.

Создайте класс, который может записывать любой объект в упрощенном формате JSON:

- Объект начинается с { и заканчивается }

- Свойство это пара "ключ:значение"

- Все свойства отделены друг от друга точкой с запятой

- Все названия свойств в двойных кавычках

- Значениями свойств могут быть только строки ( в двойных кавычках) или числа (с плавающей точкой, без кавычек)

- Создайте аннотацию @JsonName, чтобы определять имя свойства в JSON формате

- Создайте аннотацию @JsonIgnore, чтобы исключить поля из сериализации

Например, класс Person {

@JsonName("name")

String firstName="Vasya";

@JsonName("years")

double age=12

@JsonIgnore

String password;

}

будет выглядеть так

{

"name":"Vasya",

"age":12

}

Создайте класс, который может читать любой объект в упрощенном формате JSON.

например ,deserialize(<пример выше>, Person.class) вернет экземпляр Person{name="Vasya",age=12,password=null}

**Домашнее задание 9**

Потоки и файлы.

Задание 1. Файловый произвол  
- Создате новую папку с названием SE2020\_LESSON9 в любом месте  
- Создайте от 1 до 3 (произвольное число) вложенных папок. Название папки - Directory\_<уровень вложенности>

Пример, какие папки могут получится  
SE2020\_LESSON9/1 - если создалась всего одна папка  
SE2020\_LESSON9/1/2- если создались две папки.  
SE2020\_LESSON9/1/2/3 - если создались все три папки.  
Обратите внимание, что папка 3 - вложена в папку 2, а папка 2 - в папку 1.

- В каждой папке (включая корень SE2020\_LESSON9) создайте 1-3 файла:  
    содержимое каждого файла - 10-200 случайных цифр (цифр, а не чисел).  
    название файла - File\_<номер файла, начиная с 1>

- выведите на экран полный путь до всех файлов  (но не папок, вложенные файлы тоже считаются)) в папке SE2020\_LESSON9  
Совет: Для обхода можно использовать Files.walkFileTree

Задание 2. Фильтрованные атрибуты  
Используйте структуру файлов и папок из Задания 1.

- Выведите полное  название и содержимое самого маленького файла. Если таких файлов несколько - выведите их все  
- Выведите полное название и первые 10 символов самого большого файла.   
- Удалите папку SE2020\_LESSON9 и все вложенные файлы и папки.

**Домашнее задание 10**

.Многопоточность.

1. Создайте коллекцию из трех http ссылок

2. Загрузите содержимое всех трем ссылкам последовательно.

3. Выведите на экран:

- время, которое потребовалось, чтобы загрузить всё содержимое

- суммарный размер содержимого всех трех ссылок в байтах.

Пояснение: 1. Нужно вывести время, которой потратилось на загрузку всех трех ссылок, а не время, которое было потрачено на загрузки одной ссылки. Идея в том, чтобы сравнить последовательную и параллельную загрузку. Параллельная должна быть быстрее (раза в три). 2. Нужно вывести суммарный размер (размер ссылки1 + размер ссылки 2 + размер ссылки 3).  
Идея в том, что вы должно не просто запускать код асинхронно, но потом и иметь возможность дождаться результатов каждой задачи.

4. Теперь загрузите содержимое параллельно, используя три потока, выведите на экран тоже самое (время и размер).

Советы:

- Для хранения содержимого разных ссылок используйте потокобезопасные коллекции:

ConcurrentHashMap,ConcurrentHashSet, CopyOnWriteArrayList, ConcurrentLinkedQueue

- Для получения содержимого используйте new URL().openStream() (не забывайте закрывать поток)

**Домашнее задание 11**

Лямбда-выражения

Задание 1.  
создайте класс Person  с полями  
 имя:String,  
 возраст:int

 Создайте коллекцию людей  
 Отсортируйте коллекцию сначала по имени, выведите на экран  
  Затем - по возрасту, выведите на экран.

Для сортировки используйте Collecitons.sort(collection, comparator)  
компаратор задайте в виде лямбда выражения

Задание 2.Расширяемый калькулятор  
Создать класс Calculator, который может выполнять операции на двумя числами.  
определить метод calculate(String operationName, Double number1, Double number2);  
   
Операции могут быть разными и хранятся внутри калькулятора.

Определите интерфейс   
interface Operation {   
       Double doOperation(Double number1, Double number2) throw ArithmeticException;  
 }  
   
 в классе калькудятор создайте метод, который добавляет новыую операцию в калькулятор:  
 void addOperation(String operationName, Operation implementation);  
   
 Добавьте в калькулятор операции:  
 Умножение  
 Деление  
 Сложение  
 Вычитание  
 Возведение в степень  
 Извлечение корня степени n  
   
 Для опредления тела операции используйте лямбда выражения  
 Например:  
 Calculator calc = new Calculator();  
 calc.add("sum", (a,b) -> a+b);  
 calc.calculate("sum",2,3);// выводит 5

Задание 3. Потоковые лекции

Создайте класс Lection с полями : название, дата проведения. Описывает лекцию по какой-то дисциплине в определенную дату.

В качестве даты можете использовать LocalDate, используя конструктор LocalDate.of

Создайте класс Student с полями имя (строка) и список посещенных курсов в формате Set<Lection>.

Создайте List<Student> students и заполните его 10 студентами, которые посетили 5 разных курсов (матанализ,философия, английкий язык, история, физкультура).

Количество посещений придумайте самостоятельно.

Теперь проведем аналитику:

Для всех заданий используй students.stream() и операции с потоками.

1. Выведите список студентов, которые хоть раз посещали матанализ.

2. Выведите статистику посещений для каждого студентам в формате: имя - количество посщенных лекций.

3. Выведите название дисциплин, имеющих наибольшее количество посещений.

Если два разных студента посещают одну лекцию в один день, то это считается как два посещения.

4. Выведите имена студентов, которые посетили наибольшее количество лекций в день.

5. Выведите статистику по курсам в формате:

название курсов - количество разных студентов, которые посетили хотя бы одно занятие. (т.е. в лучше случае это будет 10)

Обратите внимание, что большинство заданий не предполагает "однострочного" решения цепочкой stream'ов.

Где-то придется заводить временные переменные или методы.

не забывайте, что логика по работе лежит не только в классе Stream, но еще Collectors, в частности, полезным может оказаться

grouppingBy

**Домашнее задание 12**

Как установить Maven:

1. Скачать архив с сайта <https://maven.apache.org/download.cgi>
2. Распаковать куда угодно
3. Добавить в PATH <https://maven.apache.org/install.html>
4. Убедитесь, что в PATH прописан путь JAVA\_HOME к JDK   
   Проверьте, что все работает - наберите в консоли mvn –version
5. Может понадобиться настроить прокси:

<https://maven.apache.org/guides/mini/guide-proxies.html>

Задание 12. Фреймворки.

1. Создайте новый Maven проект из архетипа maven-archetype-quickstart

http://maven.apache.org/archetypes/maven-archetype-quickstart/

в pom.xml добавьте зависимость от артефакта groupId = org.apache.commons, artifactId = commons-lang3,version = 3.10

версию можно не указывать, тогда Maven будет использовать последнюю доступную версию, но это считается небезопасным.

commons-lang3 - это библиотека общего назначения, включает в себя функции для работы со строками, числами,объектами, случайными строками и т.п.

- В сгенерированном проекте определите класс ShopItem с полями:

Категория - строка

Цена - double

Наименование - строка

Количество - int

ID - строка

ID - это строка из 25 символов (английские буквы или цифры)

- переопределите equals и hashCode, используя org.apache.commons.lang.builder.EqualsBuilder и HashCodeBuilder

Подсказка: при генерации equals и hashCode в IDEA можно выбрать шаблон, чтобы использовались эти классы

- Создайте класс Shop и добавьте 10 товаров. Для генерации ID используйте класс RandomStringUtils из commons-lang

- Создайте класс ItemSorters и определите в нем методы, отвечающие за сравнение двух объектов

int sortByCategory(ShopItem, ShopItem)

int sortByTitle(ShopItem, ShopItem)

int sortByPrice(ShopItem, ShopItem)

int sortByQuantity(ShopItem, ShopItem)

в классе Shop создайте коллекцию для хранения компараторов :

List<Comparator<ItemShop>>

заполните это коллекцию, использую MethodReference (ссылайтесь на методы из класса ItemSorters)

- Выведите все товары из списка в формате, отсортированную, используя случайный компаратор

ID Категория Наименование Цена Остаток

Длинные значения (больше 10 символов) полей сократите, используя StringUtils.abbreviate