ДЗ_2

Задание 1

1. Создать новый проект на своем компе
2. Создать новый проект на GitHub и объединить проект локальный с проектом на GitHub
3. Написать программу, которая делает следующее:
    a. спрашивает у пользователя имя (String)
    b. спрашивает у пользователя город проживания (String)
    c. спрашивает у пользователя возраст (int)
    d. спрашивает у пользователя хобби (String)
    e. перед каждым вводом данных, программа должна вывести информацию пользователю с требованием ввода соответствующей информации.
    f. программа выводит красиво оформлено всю информацию о пользователе в трех разных вариантах:

    Вариант 1 (табличный):
    Имя: xxxxxx
    Город: xxxxxx
    Возраст: xxxxxx
    Хобби: xxxxxx

    Вариант 2 (текстовый):
    Человек по имени xxxxxx живет в городе xxxxxx.
    Этому человеку xxxxxx лет и любит он заниматься xxxxxx.

    Вариант 3 (иной):
    xxxxxx - имя
    xxxxxx - город
    xxxxxx - возраст
    xxxxxx - хобби

    ---
4. По ходу решения задачи, пушить изменения на GitHub
5. Финальный рабочий код программы которая соответствует всем подпунктам пункта 3 залить на GitHub

ДЗ_3

Задание 1

Создать новый проект на GitHub и на локальном компе
Написать программу которая:
1. на вход через консоль принимает размер массива
2. на вход через консоль принимает массив чисел *(весь массив вводится в одну строку через пробел)*
3. найти минимальное число в массиве и вывести в консоль (без использования сортировки)
4. найти максимальное число в массиве и вывести в консоль (без использования сортировки)
5. посчитать кол-во повторений числа 5 и вывести в консоль
6. вывести в консоль отсортированный массив

Задание 2 (дополнительное)

Добавить к программе из задания 1 следующее:
7. вывести в консоль максимальное кол-во повторений чисел в массиве
пример 1:
массив 1, 2, 3, 4, 1, 6.
Ответ - 2. Так как число 1 повторяется 2 раза

пример 2:
массив 1, 1, 1, 4, 6, 6.
Ответ - 3. Так как число 1 повторяется 3 раза. А число 6 повторяется 2 раза. Поскольку надо вывести максимум, выводим 3.

пример 3:
массив 2, 3, 3, 5, 5, 6
Ответ - 3. Так как 3 и 5 повторяются по 2 раза, неважно кого из них подсчитывать, цель - вывести максимум. В этом примере максимум повторений чисел является 2 раза.

8. вывести в консоль минимальное кол-во повторений чисел в массиве

Задание 3 (дополнительное)

9. Избавиться от пункта номер один. Пользователь не вводит явно длину массива. Он вводит только содержимого самого массива в одну строку. Программа должна создать массив подходящего размера автоматически и заполнить массив введенными пользователем данными.
пример:
10, 10, 10, 10 тут размер массива 4;
10, 10 тут размер массива 2

ДЗ_3

Задание 1

Написать функцию которая выводит в консоли от 1 до числа X.
Аргументы функции: число X

Например X = 5.
Вывод программы:
1
2
3
4
5

Задание 2

Написать функцию drawRectangle которая рисует в консоли прямоугольник из символов '+'
Аргументы функции: ширина прямоугольника в символах, высота прямоугольника в символах

Например 3 на 2
Вывод программы:
+ + +
+ + +
Задание 3

Перегрузить функцию drawRectangle (задание 2) таким образом, чтобы она могла принимать на вход только 1 параметр (ширина квадрата) и рисовать квадрат с равными сторонами

Например 2
Вывод программы:
+ +
+ +

Например 3
Вывод программы:
+ + +
+ + +
+ + +

Задание 4

Написать функцию getMax которая принимает на вход два аргумента в виде чисел. А возвращает максимальное из этих двух.
Также, необходимо перегрузить эту функцию для работы с разными числовыми типами переменных (int, float)

Задание 5

Решить задачу 1, без использования циклов. Используя рекурсию.

Задание 6

Решить задачу 2, без использования циклов. Используя рекурсию.

Задание 7 (дополнительно)

Написать программу, в которой выполнены все шесть предыдущих заданий. Каждое задание выполняется в отдельной функции. В пределах этой же функции происходит ввод с консоли необходимых данных.
Программа спрашивает у пользователя какую задачу он хочет решить (от 1 до 6).
Затем программа вызывает соответствующую функцию для решения этой задачи.
По окончанию решения задачи, программа спрашивает пользователя, хочет ли он продолжить решать задачи. Если да - опять спрашивает какую задачу

ДЗ_4

Задача 1

Создать пакет соответствующий(название на ваш вкус, но должно быть логично связано с именами классов) пакет
и поместить туда все последующие классы

Класс CarDoor

На прямую к переменным этого класса никто не может, только через методы

Хранит

- состояние двери(открыта/закрыта)
- состояние окна (открыто/закрыто)

Конструктор

- Один без аргументов. Он должен присвоить переменным значения на случай если данных нет.
- Один конструктор принимает оба состояния, двери и окна. Присваивает эти значения переменным внутри объекта.

Методы

- открыть дверь
- закрыть дверь
- открыть/закрыть дверь (если дверь открыта и вызывается эта функция, значит дверь необходимо закрыть и наоборот)
- открыть окно
- закрыть окно
- открыть/закрыть окно(если дверь открыта и вызывается эта функция, значит дверь необходимо закрыть и наоборот)
- Вывести в консоль данные об объекте

Класс CarWheel

На прямую к переменным этого класса никто не может, только через методы

Хранит

- Состояние целостности шины (дробное число от 0-стерта до 1-новая)

Конструктор

- Аналогичный принцип как в классе CarDoor

Методы

- Сменить шину на новую
- Стереть шину на X%
- Получить состояние (return)
- Вывести в консоль данные об объекте

Класс Car

На прямую к переменным этого класса никто не может, только через методы

Хранит

- дата производства (неизменна после создания объекта)
- тип двигателя
- максимальная скорость машины (если она новая)
- время разгона до 100км/ч
- пассажировместимость
- кол-во пассажиров внутри в данный момент
- текущая скорость
- массив колес
- массив дверей

Конструктор

- Нет пустого конструктора. Так как есть поля в классе, которые нельзя изменять после создания объекта. Например дата производства.
- Конструктор с датой производства.
- Конструктор со всеми полями, кроме массива колес и массива дверей.

Методы

- Изменить текущую скорость
- Посадить 1 пассажира в машину
- Высадить 1 пассажира
- Высадить всех пассажиров

- Получить дверь по индексу
- Получить колесо по индексу
- Снять все колеса с машины
- Установить на машину X новых колесу (вдобавок к имеющимся, то есть если было 4 колеса, после вызова метода с X аргументом равным трем, колес будет 4+3=7)
- Вычислить текущую возможную максимальную скорость (Скорость машины вычисляется так. Максимальная скорость новой машины множиться на самое стёртое колесо в машине. Максимальная скорость равна 0 если в машине нет ни одного пассажира, так как некому ее вести)
- Вывести в консоль данные об объекте (все поля и вычисленную максимальную скорость в зависимости от целостности колес и наличия водителя)

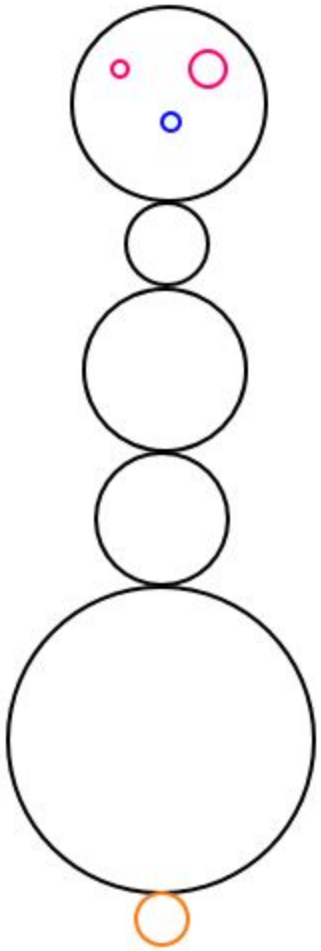
Задание 2 (дополнительное)

Создать консольный пользовательский интерфейс. В котором пользователя программа будет спрашивать какое действие выполнить и с какими параметрами.
Кол-во различных действий = кол-ву функций в ДЗ.

ДЗ_5

Задание 1

Создать оконное приложение на JavaFX
Необходимо нарисовать снеговика исходя из пожеланий пользователя. Снеговик состоит из кругов.
Каждый последующий круг находиться над предыдущим, касаясь его(!). Если круг будет налазить на другой круг, либо между ними будет пробел - значит снеговик отрисован не верно. Радиусы кругов рандомизированы.
Цвета кругов должны быть рандомизированы.
На голове снеговика(верхний круг) из кругов должен быть отрисован нос и 2 глаза (меньше головы).
Круги на теле должны быть рандомизированы. Нет последовательности на уменьшение или увеличение.



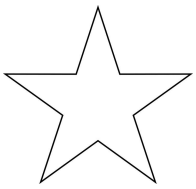
Данные вводятся через консоль:

1. кол-во кругов
2. мин. радиус круга
3. макс. радиус круга

Дополнительные задания

1. Убрать ввод с консоли и заменить его на UI элементы (кнопки и поля для ввода)
2. Добавить кнопку - покрасить все круги в красный. Она красит все ранее нарисованные круги в красный цвет.
3. Добавить кнопку - Gradient. Она красит все круги начиная от нижнего к верхнему серыми оттенками. К примеру нижний круг полностью черный, а каждый круг выше становиться более светлым. И в конечном счете верхний круг снеговика должен быть светло серым.

Задание 2

Создать оконное приложение на JavaFX
Необходимо отрисовать правильную пятиконечную звезду исходя из пожеланий пользователя. 

Данные вводятся через консоль:

1. X центра звезды
2. Y центра звезды
3. радиус звезды (расстояние от центра звезды до любого ее конца)

ДЗ_6

Задание 1

Существуют различные MP3 плееры. Каждый из них чем-то отличается от другого. В задаче рассмотрим несколько видов плееров начиная от дешевых к дорогим. Суть задания заключается в создании классов, которые будут описывать все эти плееры. В главном классе программы необходимо создать объекты каждого класса и продемонстрировать их работу.

Плееры работают с таким понятием как “**песня**” и “**плейлист**”.

Песню будем рассматривать как строку, которая состоит из названия песни. К примеру вот как выглядит одна песня
String song1 = “The Best Song”;

Плейлист выглядит как массив песен(строк).
String[] playlist = new String[] {“The best song”, “Good song”, “Super Song”};

Проигрывая песню, плееру необходимо вывести в консоль - “Playing: “ + songName;
Например вот такой вывод должен получиться при проигрывании песни “The best song”:
Playing: The best song

Код логики методов не должен дублироваться в классах!

Плееры не могут иметь больше возможностей, чем указано!
Каждый новый плеер == новый класс

Каждый плеер имеет метод **public void playSong**
Проигрывает первую имеющуюся песню на плеере.

Некоторые плееры имеют метод **public void playAllSongs()**
Проигрывает все песни на плеере

Плеер 1.
Имеет final цену(задается в конструкторе) и геттер
Имеет только 1 песню (нельзя объявить эту переменную в классе этого плеера)
playSong Может проиграть песню.

Плеер 2.
Имеет final цену(задается в конструкторе) и геттер
Имеет только 1 песню (нельзя объявить эту переменную в классе этого плеера)
playSong Пытаясь проиграть песню выдает ошибку в консоль - “error”.

Плеер 3.
Имеет final цену(задается в конструкторе) и геттер
Имеет плейлист
playSong Может проиграть первую песню
playAllSongs может проиграть все песни

Плеер 4.
Имеет final цену(задается в конструкторе) и геттер
Имеет плейлист
playSong Может проиграть **последнюю** песню
playAllSongs может проиграть все песни

Плеер 5.
Имеет final цену(задается в конструкторе) и геттер
Имеет плейлист
playSong Может проиграть первую песню
playAllSongs может проиграть все песни с конца плейлиста в начало

Плеер 6.
Имеет final цену(задается в конструкторе) и геттер
Имеет плейлист
playSong Может проиграть первую песню
playAllSongs может проиграть все песни
Имеет метод **public void shuffle() -** перемешивает все песни в плейлисте местами

Задание 2 (дополнительное)

Научить плееры отрисовывать свои интерфейсы, чтобы пользователь мог в графическом представлении работать с плеером, без консоли.
Добавить к классам плееров метод **public void show(Pane root)**.
Метод должен уметь отрисовать в окно все функции плеера. Не забываем о наследование и переопределении методов при создании этого метода.

Добавить визуализацию проигрывания песен и кнопок(функций плееров) в любом удобном для вас виде, но не консольном.

Задание 3 (дополнительное high-level)

Самостоятельно нагуглить и разобраться как проигрывать mp3 файлы.
Создать новый главный класс программы - **RealPlayer** и инициализировать там метод main который будет запускать плеер с графическим интерфейсом на JavaFX.
Научить плееры пользоваться mp3 файлами, а вместо строк выводить теперь настоящие названия песен.

ДЗ_7

Задание 1

Создать базовый класс для цветов. От него наследуется 3 класса-цветка:

- роза,
- ромашка,
- тюльпан

Создать класс **FlowerStore** который продает букеты цветов через функцию **sell**. Эта функция принимает 3 числа. Кол-во роз, кол-во ромашек, кол-во тюльпанов. А вернуть должна один массив цветов, в котором будут храниться цветы. Один объект в массив == один цветок в букете. Последовательность расстановки цветов в букете не имеет значения.
Добавить метод **sellSequence** который работает так же как и метод sell, принимает 3 числа(кол-во цветов) но теперь цветы в букете должны идти чередуясь: роза, ромашка,

тюльпан. Если для последовательности не хватает цветков, продолжать выводить по той же последовательности но пропуская цветки которые лишние.
Например 5 розы, 3 ромашки, 1 тюльпан:
роза, ромашка, тюльпан, роза, ромашка, роза, ромашка, роза, роза

В главном классе программы продемонстрировать работу метода sell и sellSequence
Вывести по букету в консоль через запятую в одну строчку.
По'ля String name у классов-цветов нет

Задание 2

Добавить к классам-цветам поле - цена. Роза стоит 100, ромашка 70, тюльпан 45.
Добавить к классу FlowerStore кошелек. Который хранит сколько в магазине денег. После продажи букета - пополнять кошелек магазина на сумму проданных цветов.

Задание 3

Написать класс FlowersLoader. Этот класс имеет статическую функцию - **load** которая принимает путь к файлу, а возвращает массив объектов цветов (букет).
Необходимо гарантировать чтобы объект этого класса создать было нельзя.

Задание 4

Написать класс FlowersSaver. Этот класс имеет статическую функцию - **save** которая принимает путь к файлу и массив цветов (букет). Должна сохранить цветы в файл.
Необходимо гарантировать чтобы объект этого класса создать было нельзя.

ДЗ_8

Нельзя использовать Java коллекции при выполнении заданий!
Примеры с урока тут

Задание 1 - ArrayList

Написать свой класс MyArrayList как аналог классу ArrayList.
Можно использовать 1 массив для хранения данных.

Методы

- add(T value) добавляет элемент в конец
- remove(int index) удаляет элемент под индексом
- clear() очищает коллекцию
- size() возвращает размер коллекции
- get(int index) возвращает элемент под индексом

Задание 2 - LinkedList

Написать свой класс MyLinkedList как аналог классу LinkedList.
Нельзя использовать массив. Каждый элемент должен быть отдельным объектом-посредником(Node - нода) который хранит ссылку на прошлый и следующий элемент коллекции(двусвязный список).

Методы

- add(T value) добавляет элемент в конец
- remove(int index) удаляет элемент под индексом
- clear() очищает коллекцию
- size() возвращает размер коллекции
- get(int index) возвращает элемент под индексом

Задание 3 - Queue

Написать свой класс MyQueue как аналог классу Queue и LinkedList.
FIFO (first-in-first-out)
Можно делать либо с помощью Node либо с помощью массива.

Методы

- add(T value) добавляет элемент в конец
- remove(int index) удаляет элемент под индексом
- clear() очищает коллекцию
- size() возвращает размер коллекции
- peek() возвращает первый элемент в очереди (FIFO)
- poll() возвращает первый элемент в очереди и удаляет его из коллекции

Задание 4 - Stack

Написать свой класс MyStack как аналог классу Stack.
LIFO (last-in-first-out)
Можно делать либо с помощью Node либо с помощью массива.

Методы

- push(T value) добавляет элемент в конец
- remove(int index) удаляет элемент под индексом
- clear() очищает коллекцию
- size() возвращает размер коллекции
- peek() возвращает первый элемент в стеке (LIFO)
- pop() возвращает первый элемент в стеке и удаляет его из коллекции

Задание 5 - HashMap

Написать свой класс MyHashMap как аналог классу HashMap.
Нужно делать с помощью односвязной Node.
Не может хранить две ноды с одинаковых ключами одновременно.

Методы

- put(T key, K value) добавляет пару ключ + значение
- remove(T key) удаляет пару по ключу
- clear() очищает коллекцию
- size() возвращает размер коллекции
- get(T key) возвращает значение(K value) по ключу

ДЗ_9

Задание 1

Ваш друг открыл лавочку с фруктами. Но к сожалению он не справляется со всей новой информацией которая поступает. Он попросил вас разработать систему которая ему с этим поможет.

Потребуется создать класс фрукта. Фрукт хранит:

- вид (клубника, яблоко, груша и т.д.) храниться в Enum. Придумать до 10 видов.
- срок годности (10 дней, 20 дней и т.д.)
- дата поставки
- цена

И класс - торговая лавка. Он будет отвечать за хранение информации о складе лавки.

Необходимо принимать поставки фруктов и дополнять базу данных склада лавки новой информацией. **void addFruits(String pathToJsonFile)** - метод принимает путь к файлу внутри которого находится json с фруктами и датой поставки. Один файл - одна поставка. Базу данных во время работы программы можно хранить в коллекции.

Пример одной поставки

```
{
	"fruits": [{
				"type": "Banana",
				"shelfLife": 10,
				"date": "10/07/2017",
				"price": 100
		},
		{
				"type": "Apple",
				"shelfLife": 5,
				"date": "10/07/2017",
				"price": 70
		}
	]
}
// в этой поставке 1 яблоко и 1 банан
```

Нужен метод **void save(String pathToJsonFile)** который сохранит всю информацию со склада лавки в json файл.
Аналогичный метод **void load(String pathToJsonFile)** который удаляют текущую информацию из коллекции и загружает новую из сохраненной версии.

В главном классе программы проверьте все методы на работоспособность. Создайте объект класса-лавки, дайте ему несколько поставок с разными данными и датами. Затем сохраните информацию со склада и проверьте полученный json файл.
Также, попробуйте загрузить информацию о складах лавки на прямую из полученного json файла.

Задание 2

Добавить к классу-лавке метод способный сказать какие продукты испортятся к заданной дате **List<Fruit> getSpoiledFruits(Date date)**

И метод который возвращает список готовых к продаже продуктов
List<Fruit> getAvailableFruits(Date date)

В главном классе программы проверьте работоспособность этих методов.

Задание 3

Перегрузить имеющиеся методы **spoiledFruits** и **availableFruits**.
На прием еще одного параметра - вид фрукта
List<Fruit> getSpoiledFruits(Date date, Type type)
List<Fruit> getAvailableFruits(Date date, Type type)
Работают как и прежде, но теперь фильтруют только по заданному типу фрукта

Добавить метод который возвращает продукты которые были доставлены в заданную дату **List<Fruit> getAddedFruits(Date date)** и его переопределенная версия - **List<Fruit> getAddedFruits(Date date, Type type)**

Задание 4 (дополнительно)

Необходимо учитывать продукты которые были проданы. Для этого добавим метод **void sell(String pathToJsonFile)**. Метод принимает путь к файлу с джейсоном который хранит записи о клиентах который хотели купить продукты в заданный день.

Если продукты продукты присутствуют на складе в заданном количестве - сделка происходит и товары удаляются со склада, а на счет лавки зачисляются деньги за продукты и продукты со склада удаляются.
В противном случае сделка не происходит и клиент уходит ни с чем, а продукты остаются не тронутыми.

Необходимо добавить числовое значение moneyBalance которое хранит текущий баланс денег лавки. Должен сохраняться и загружаться при вызовах методов save и load.

```
{
	"clients": [{
				"name": "Вася",
				"type": "Apple",
				"count": 100
			},
			{
				"name": "Джон",
				"type": "Apple",
				"count": 500
			},
			{
				"name": "Джон",
				"type": "Banana",
				"count": 1
			}
	]
}
// Вася хочет купить 100 яблок, а Джон хочет купить 500 яблок и 1 банан.
```

Задание 5 (дополнительно high-level)

После выполненных задач, ваш друг разбогател и открыл много лавок с фруктами. Теперь он просит вас помочь с менеджерингом всех сразу.

Необходимо создать класс Company. Который имеет внутри себя

- коллекцию лавок
- moneyBalance - баланс компании

Методы

- void save(String pathToJsonFile) сохраняет всю информацию компании
- void load(String pathToJsonFile) загружает всю информацию компании
- геттер лавки по индексу из коллекции
- int getCompanyBalance() возвращает сумму балансов всех лавок
- List<Fruit> getSpoiledFruits(Date date)
  работает также, но в масштабах компании (по всем лавкам)
- List<Fruit> getAvailableFruits(Date date)
  работает также, но в масштабах компании (по всем лавкам)
- List<Fruit> getAddedFruits(Date date)
  работает также, но в масштабах компании (по всем лавкам)
- List<Fruit> getSpoiledFruits(Date date, Type type)
  работает также, но в масштабах компании (по всем лавкам)

- List<Fruit> getAvailableFruits(Date date, Type type)
  работает также, но в масштабах компании (по всем лавкам)
- List<Fruit> getAddedFruits(Date date, Type type)
  работает также, но в масштабах компании (по всем лавкам)

ДЗ_9

Все задания поддать тестированию при котором многократно и очень быстро нажимаются одни и те же или разные кнопки в интерфейсах программ. Программы должны работать корректно и не крешиться при этом.

Задание - библиотека 1

Написать программу которая считывает с консоли число peopleCount и maxAmount.
Затем создает peopleCount потоков. Каждый поток - человек, хочет войти в библиотеку.
Библиотека имеет лимит кол-ва людей которые одновременно могут в ней находиться - maxAmount.
Каждый поток должен писать в консоль что он(человек) делает.

- пришел ко входу в библиотеку
- ждет входа в библиотеку (происходит только если нет места на момент прихода к библиотеке)
- вошел в библиотеку
- читает книгу (поток должен делать это действие рандомное кол-во времени от 1 до 5 секунд)
- вышел из библиотеки

Задание - библиотека 2

Надо добавить в программу дверь - вход/выход в библиотеку.
В дверь может одновременно проходить 1 поток. Время прохождения через дверь потока = 0.5 секунды).
Появляются дополнительные действия со стороны поток которые они должны писать в консоль:

- подошел к двери с улицы
- подошел к двери изнутри
- проходит через дверь внутрь

- проходит через дверь наружу
- прошел через дверь внутрь
- прошел через дверь наружу

Задание - геометрия в движении 1

Создать JavaFX приложение которое имеет одну кнопку и большую пустую область. Кнопка - «Multi Threads» генерирует от 3 до 10 прямоугольников рандомного цвета и размера. Выставляет все прямоугольники на пустую плоскость окна. Прямоугольники могут накладываться друг на друга. Каждый прямоугольник имеет отдельный поток который двигает прямоугольник в одном из 4 направлений

1. Лево-вверх (x—, y—)
2. Право-вверх (x++, y—)
3. Лево-вниз (x—, y++)
4. Право-вниз (x++, y++)

При столкновением прямоугольника со стенкой окна программы, прямоугольник отбивается под соответствующем углом (это будет всегда +или- 90 градусов)

Задание - геометрия в движении 2

- Добавить в приложение вторую кнопку
- Кнопка - «Single Thread», она создает один дополнительный поток(вместо множества потоков по одному на прямоугольник) который рассчитывает движение каждого прямоугольника. При этом не блокируя main поток.

Задание - геометрия в движении 3 (дополнительно)

- Добавить в приложение третью кнопку
- Кнопка - «Optimal Threads» она создает столько потоков, сколько комп на котором программа запускается имеет ядер. Каждый поток делит с другими равное кол-во прямоугольников для оптимального распределения нагрузки.
- Добавить в приложение CheckBox - «Collide with each other» если true значит прямоугольники должны уметь сталкиваться друг с другом и оба отбиваться на 90 градусов. В случае если они наложились друг на друга, надо их отодвинуть до позиции касания друг к другу. В противном случае они зависнут в постоянном цикле столкновений друг с другом.
- Добавить два текстовых поля для ввода минимального макс. и мин. кол-ва генерируемых прямоугольников.

Задание - геометрия в движении 4 (дополнительно - high level)

- Добавить к прямоугольникам: круги и равносторонние треугольники
- Добавить настройки генерации фигур:
    - Мин макс ширина прямоугольника
    - Мин макс высота прямоугольника
    - Мин макс радиус круга
    - Мин макс радиус вписанной в треугольник окружности
    - Мин макс кол-во прямоугольников
    - Мин макс кол-во кругов
    - Мин макс кол-во треугольников

ДЗ_10

Код с урока тут

Задание 1

Пользователь вводит с консоли числа A и B.
Программа спрашивает пользователя какое математическое действие надо применить к этим числам:

1. +
2. -
3. *
4. /
5. %
6. ==
7. >
8. <

Действие пользователь вводит в виде соответствующего символа
После чего программа в отдельном потоке выполняет действие.
Результат в консоль должен вывести main поток
Решить задачу без использования глобальных переменных

Задание 2

Написать функцию которая создает массив интов размером size. Числа в массиве идут по возрастанию от 1 до size.

С помощью этой функции создайте массив с size равным 80 000 000.

Подсчет

- для каждого элемента массива подсчитать result = sin(x) + cos(x), где x - итый элемент массива. Вывести в консоль сумму всех result для всего массива. Распараллельте эту логику на потоки для ускорения вычислений.

Пользователю надо ввести N в консоль. N это кол-во раз, сколько надо повторить *Подсчет*. Одновременно в программе может быть запущено только вычисление одного *Подсчета*. Но при этом саму итерацию подсчета нужно параллелить.

Программа должна во время одного запуска отработать в двух режимах. И подсчитывать время которое она затратила на каждый из режимов работы:

- Режим Thread. Программа для работы создает Thread-ы.
- Режим Thread Pool. Программа использует один thread pool единожды созданный.

Режим Thread и Thread Pool не могут работать одновременно друг с другом. Только по очереди.

Задание 3

Сохранить в программе множество из 50 ссылок в интернет на картинки.

Написать JavaFX приложение в котором есть одна кнопка - “Обновить”
И 25 картинок в одном окне в виде матрицы (5 на 5). Картинки беруться рандомно из массива ссылок.
После нажатия на кнопку “Обновить”, картинки должны быть еще раз рандомно взяты из множества ссылок, загружены и отображены в окне.
При этом нельзя хранить на жестком диске картинки. Каждый раз надо загружать их из интернета.

Задание 4 (дополнительно)

Дополнить задание 3.

Добавить ProgressBar поверх картинки которая находиться в состоянии загрузки.

Задание - текстовый редактор 1

Создать JavaFX приложение в котором есть текстовое поле для ввода пути к файлу.
Кнопка Load которая загружает файл в отдельном потоке.
И большое текстовое поле куда выводиться содержимое файла.
Пользователь должен иметь доступ редактировать содержимое файла прямо в окне программы.
Последним элементом программы является кнопка Save которая сохраняет файл по указанному пути. Делает это в отдельном потоке

Задание - текстовый редактор 2

Для решения этой задачи необходимо выполнить первую задачу.
Надо добавить в программу:

1. Текстовое поле для ввода числа
2. Кнопку Fibonacci которая берет число из текстового поля, назовем его X.
3. Затем в отдельном потоке начинает искать(нельзя хранить заранее созданные числа) число Фибоначчи под номером X.
4. Все числа Фибоначчи до числа Фибоначчи под номером X должны быть записаны в файл, путь к которому указан в текстовом поле (см. Задание 1).
5. При процессе поиска числа Фибоначчи программа должна реагировать на действия пользователя. Перетаскивать мышкой по экрану, редактироваться текстовые поля. Другими словами - не должна зависать.

Задание - текстовый редактор 3 (дополнительно)

Программа должна уметь работать с значением X >= 500.000.000.000.000.000.000
Также надо добавить кнопку Cancel. Который останавливает любой из процессов (Loading, Saving, Searching X).
Подсказка: Гугл по запросу «BigInteger»

Задание - текстовый редактор 4 (дополнительно - high level)

Добавить в окно программы:

1. ProgressBar который должен соответствовать прогрессу Загрузки, Сохранения файла и Поиску числа Фибоначчи
2. Текст поверх прогресс бара который отображает текущее состояние программы и % завершенности
3. Список состояний:
    1. Ready - пользователь ничего не делает, программа в ожидании
    2. Saving - программа сохраняет данные в файл
    3. Loading - загрузка данных из файла
    4. Editing - состояние когда пользователь редактирует загруженный файл. Это состояние происходит только тогда, когда пользователь вносит изменения в файл внутри большого текстового поля где отображается содержимое файла. Автоматический выход из этого состояния через 3 секунды после последнего изменения текста в этом поле. Если во время редактирования работает какой-то другой процесс типа загрузки, сохранения, поиска Фибоначчи и т.д. Приоритет отдается тем состоянием. Состояние Editing имеет самый низкий приоритет.
    5. X searching - поиск числа Фибоначчи.
4. Всплывающее окно которое появляется при ошибках программы.
5. Возможные ошибки:
    1. Файл для загрузки не найден
    2. Не удалось сохранить в файл
    3. Поиск X остановлен (кнопкой Cancel)
    4. Сохранение остановлено (кнопкой Cancel)
    5. Загрузка остановлена (кнопкой Cancel)
    6. Неверное значение X (если значение не целое число)
    7. Надо что бы выводило подробную информацию об ошибке:
        1. Число должно быть целым (если оно дробное)
        2. Значение должно быть числовым (если текст)

Этим списком ошибок можно не ограничиваться и найти еще, если получиться.