Завдання 5
Stream API

У Java 8 з'явився Stream API, який дозволяє працювати з вмістом колекцій, як з потоком значень. Тобто для кожного з елементів колекції виконується множина визначених проміжних операції, а у кінці викликається деякий термінальний метод, що повертає фінальний результат. Даний підхід є популярний при роботі з масивам та аналізом великих даних та застосовується у машинному навчанні.

У даному завданні пропонується самостійно реалізувати частину даного API.

В архіві знаходиться шаблон проекту, який містить інтерфейси та набір класів, які необхідно реалізувати.

Інструкція:

1. Скачайте архів з шаблоном проекту *Streams* та розархівуйте його
2. Додайте його в локальний *Git* репозиторій
    a. Перейдіть в командному рядку до директорії *Streams*
    b. Виконайте команди:
    > git init
    *> git add **
    > git commit -m "Streams initial commit"
3. Створіть на своєму GitHub новий репозиторій з іменем за наступним шаблоном apps19<surname>-hw5
4. Знову перейдіть в директорію *Streams* і виконайте наступну команду
    *> git remote add origin https://github.com/**<user_name>/apps19<surname>-hw5.git***
5. Виконайте команду
    > git push -u origin master

Після цього може відкрити проект в IDE і почати виконувати завдання

Важливо:

- не змінюйте структуру шаблоного проекту та не видаляйте файли *checkstyle.xml, pom.xml*
- при коміті проекту в локальний репозиторій (та на GitHub) необхідно щоб виключно були закомічені: *src/, checkstyle.xml, pom.xml*. Директорію *target/,* яка буде створена під час компіляції комітити не треба.

Завдання:

1. Реалізувати *наступні методи класу AsIntStream по роботі з потоком (stream) цілочисельних даних:*

    static IntStream of(int... values)
    створює початковий потік на основі масиву цілих чисел

    - Double average()

середнє значення чисел в потоці. Термінальній метод. IllegalArgumentException - if empty

- Integer max()/min()

максимальне / мінімальне значення числа в потоці. Термінальній метод. IllegalArgumentException - if empty

- long count()

кількість значень (елементів) в потоці. Термінальній метод.

- Integer sum()

сума всіх значень в потоці. Термінальній метод. IllegalArgumentException - if empty

- int[] toArray()

повертає потік у вигляді масиву. Термінальній метод.

- void forEach(IntConsumer action)

для кожного значення з потоку виконує операцію зазначену в реалізації IntConsumer. Даний метод є термінальним і нічого не повертає

- IntStream filter(IntPredicate predicate)

для кожного значення з потоку перевіряє його на предмет чи задовольняє воно умові в реалізації IntPredicate, якщо так - повертає його в результуючий потік, якщо ні - викидає

- IntStream map(IntUnaryOperator mapper)

застосовує до кожного зі значень потоку реалізацію IntUnaryOperator і повертає його в результуючий потік

- IntStream flatMap(IntToIntStreamFunction func)

застосовує до кожного зі значень потоку реалізацію IntToIntStreamFunction, яка на основі кожного зі значення створює новий потік значень, які потім об'єднуються в один результуючий потік (см. http://java.amitph.com/2014/02/java-8-streams-api-intermediate.html)

- int reduce(int identity, IntBinaryOperator op)

виконує згортку значень потоку в ціле число, початкове значення задається identity, функція згортки - в реалізації IntBinaryOperator. Термінальній метод.

2. *Шаблон Maven-проекту (не мод.ифікуйте pom.xml та checkstyle.xml):* Streams.zip

В даному проекті вже є шаблони класів і невеликий тест, які повинен почати проходити після реалізації частини методів.

При реалізації можна використовувати стандартні колекції з Java, але ***не можливості Java Stream API****.*

При реалізації ***спробуйте*** *реалізувати функціональність проміжних методів (filter, map, flatMap) таким чином, щоб визначені у них операції по виконанню дій над значеннями потоку* ***не виконувались до виклику одного з термінальних методів****. Це буде відповідати підходу "лінивих обчислень". Таку реалізацію* ***можна*** *зробити використовуючи* ***Iterator pattern*** *з попереднього завдання. Також такий підхід дозволить не створювати зайвих проміжних масивів/колекцій.*

Необхідно написати модульні тести на всі методи класу AsIntStream

3. Аналогічно до попереднього завдання має бути створена нова Build Job на системі **CI Hudson/Jenkins** з іменем ***appS19<surname>-hw5*** та виконані вимоги до якості коду та його покриття тестами
4. Помилки, які виявив дженкінс будуть також враховані =)