Бонусное задание по ПСП.

Выполнили учащиеся:

**группы 972301 – Федоренко Егор,**

**группы 972302 -Барцевич Алина.**

Задание 1. Работа с системой контроля версий GIT.

1. **Скачать и установить систему контроля версий GIT.**

Переходим на сайт <https://desktop.github.com>. Скачиваем desktop приложение для определенной операционной системы (рисунок 1). Необходимо создать личный аккаунт. После создания личного аккаунта, производим вход в приложение.



Рисунок 1 – Приложение GitHub Desktop

1. **Создать локальный репозиторий.**

Войдя в личный аккаунт, можем создать локальный репозиторий.

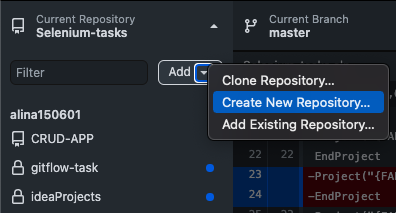


Рисунок 2 – Создание локального репозитория

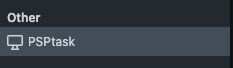


Рисунок 3 – Локальный репозиторий создан

1. **Выложить в репозиторий свой тестовый проект.**

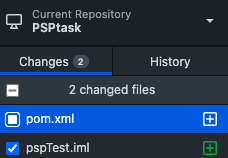


Рисунок 4 – Тестовый проект добавлен

1. **Добавить в тестовый проект новый класс. Изменить существующий код. Выложить в репозиторий.**

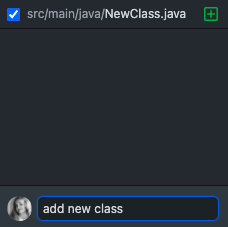


Рисунок 5 – Добавлен новый класс

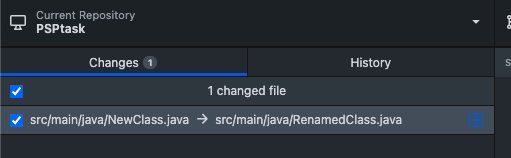


Рисунок 6 – Новый класс переименован

Новый класс выложен в удаленный репозиторий.

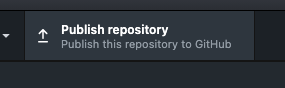


Рисунок 7 – Класс выложен в удаленный репозиторий

1. **Осуществить откат к старой версии выложенного в репозиторий проекта.**

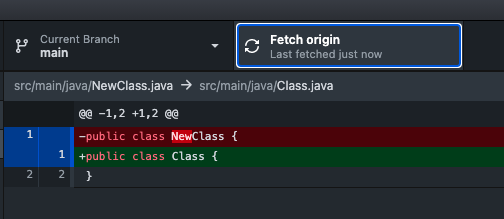


Рисунок 8 – Откат к старой версии через Fetch

1. **Удалить локальную копию проекта и скачать последнюю версию из репозитория.**

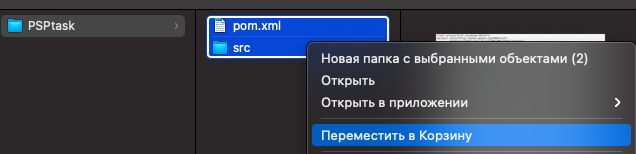


Рисунок 9 – Удаление с локального репозитория проекта



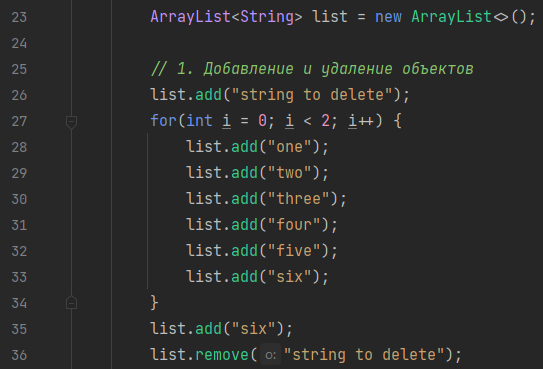
Рисунок 10 – Скачанная версия из удаленного репозитория

Задание 2. Операции над объектами типа String, содержащиеся в коллекции ArrayList.

Основным субъектом всех операций будет переменная list, инициализированная на строке 23.

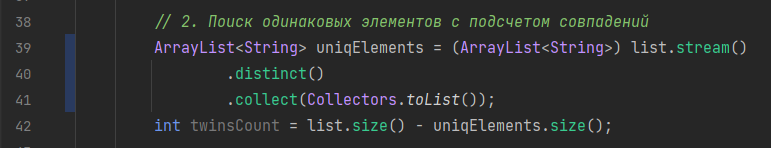
1. **Добавление и удаление объектов.**

Используем методы класса ArrayList add() и remove().



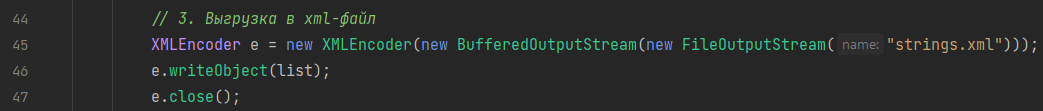
1. **Поиск одинаковых элементов с подсчетом совпадений.**

Воспользуемся Stream API, чтобы с помощью метода distinct() отфильтровать дубликаты. Разница между размером исходной коллекции и размером коллекции уникальных элементов – это количество совпадений.

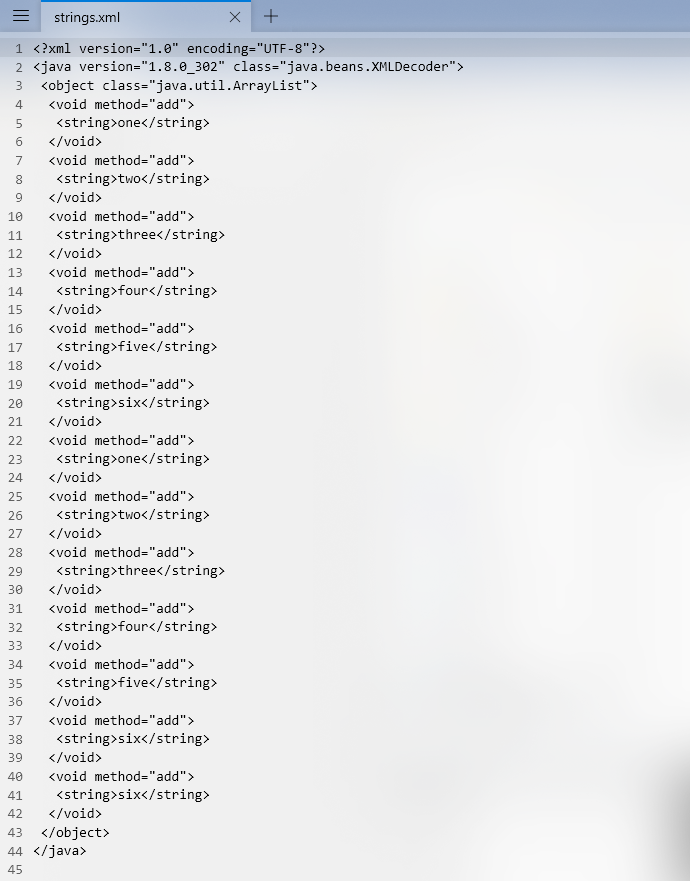


1. **Выгрузка в xml-файл.**

Воспользуемся классов XMLEncoder, чтобы сериализовать объект list. Т.к. encoder инкапсулирует поток вывода, его необходимо закрыть. Это сделано с помощью метода close() на строке 47.

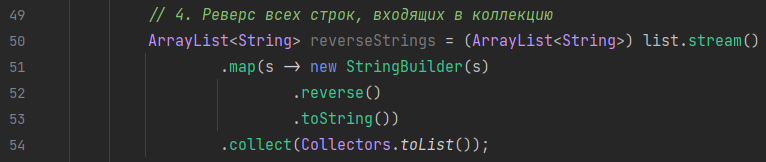


Проверим содержимое файла:



**4.     Реверс всех строк, входящих в коллекцию.**

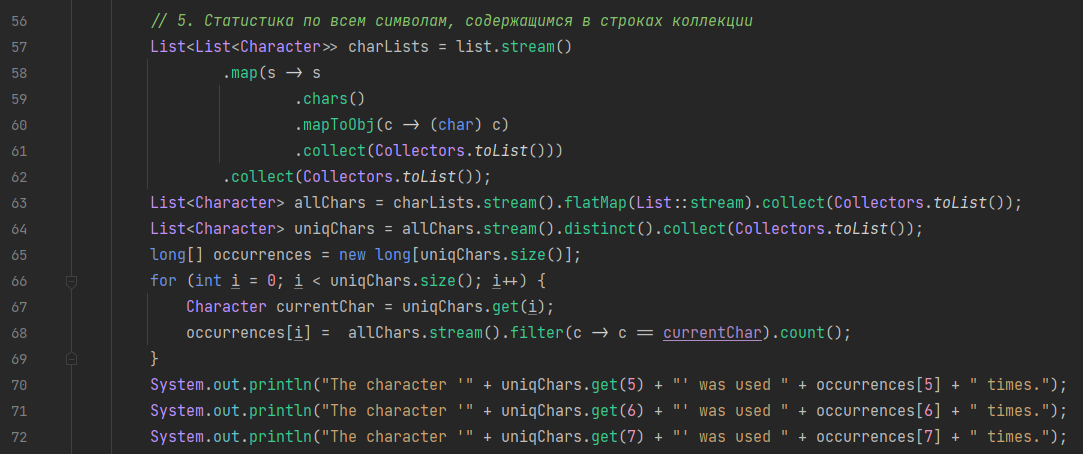
С помощью Stream API проведем над каждой строкой коллекции list следующую операцию: создать экземпляр StringBuilder на основе строки, сделать реверс строки, преобразовать StringBuilder в String.



**5.     Статистика по всем символам, содержащимся в строках коллекции.**

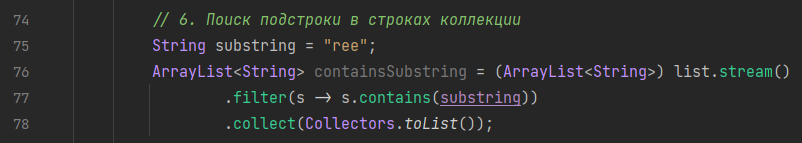
Воспользуемся Stream API, чтобы:

1. Преобразовать каждую строку в коллекцию символов.
2. Преобразовать получившуюся коллекцию коллекций символов в коллекцию символов.
3. Создать коллекцию, где будут отфильтрованы дубликаты, т.е. коллекцию с уникальными символами.
4. Подсчитать количество вхождений каждого символа из коллекции уникальных символов в коллекцию, содержащую все символы.



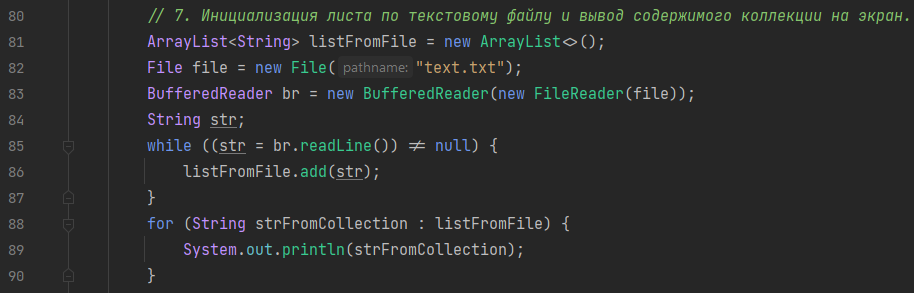
**6.     Поиск подстроки в строках коллекции.**

Воспользуемся Stream API, чтобы выбрать только те строки, которые содержат искомую подстроку.

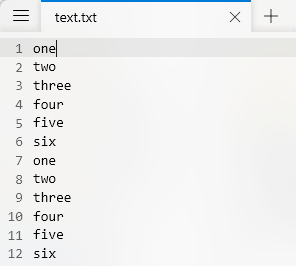
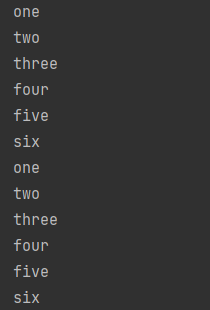


**7.     Инициализация листа по текстовому файлу и вывод содержимого коллекции на экран.**

Воспользуемся классом BufferedReader, который строку за строкой запишет содержимое файла text.txt в переменную-коллекцию listFromFile. Затем выведем в консоль каждую строку этой коллекции.

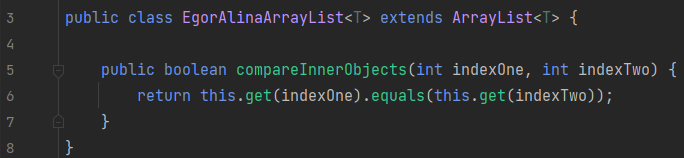


Сравним содержимое файла text.txt и вывод в консоль:

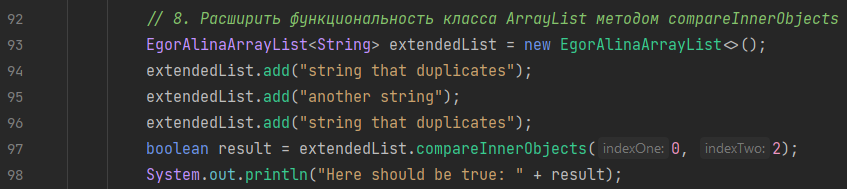
 

**8.     Расширить функциональность класса ArrayList методом compareInnerObjects ( int firstIndex, int secondIndex ).**

Создадим класс, наследующийся от ArrayList, и реализуем в нем нужный метод.



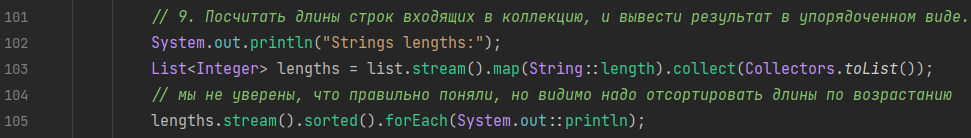
Создадим экземпляр этого класса, добавим в него элементы, среди которых есть два идентичных. Воспользуемся методом.



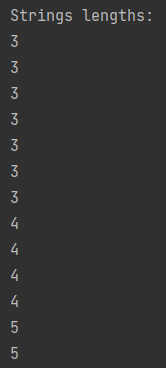
Результат: 

**9.     Посчитать длины срок входящих в коллекцию, и вывести результат в упорядоченном виде.**

Воспользуемся Stream API, чтобы создать коллекцию длин строк из изначальной переменной list. С помощью метода sort() упорядочим получившиеся значения по возрастанию. Затем используем метод forEach(), чтобы вывести каждый элемент в консоль.

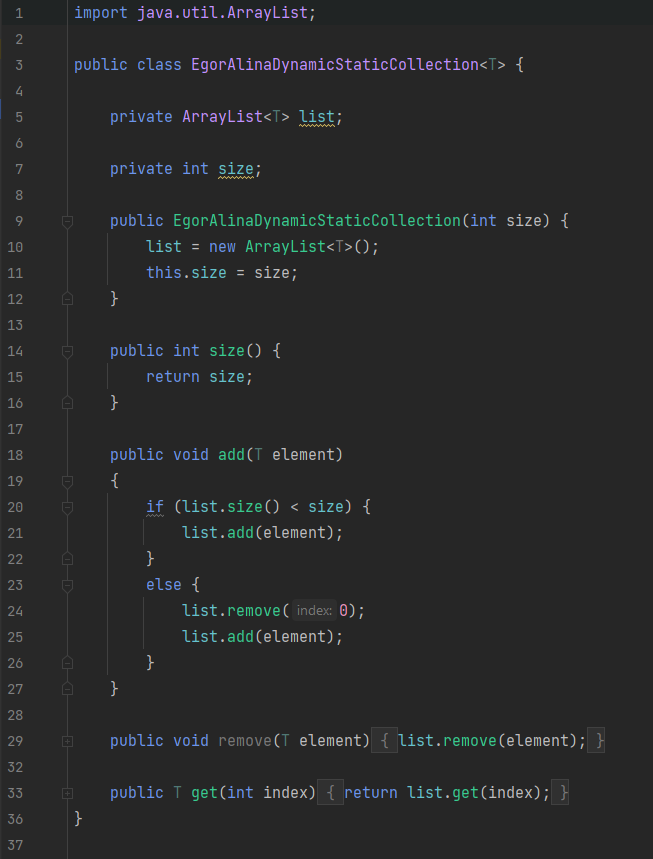


Результат:

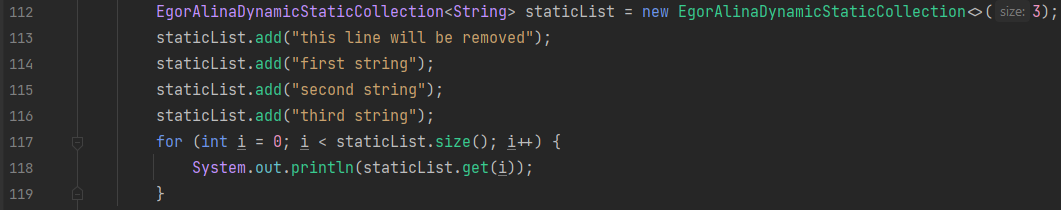


**10.     Реализовать возможность добавления в динамическую коллекцию, как если бы она была статической размерности, т.е. задаем статическую размерность коллекции, пока количество объектов меньше заданной размерности, происходит только добавление объектов, при достижении порогового значения, добавление нового элемента вызывает удаление первого элемента коллекции. Проверить, компилируется ли данный проект, прежде чем записывать его в репозиторий.**

Создадим класс, удовлетворяющий условиям. Для этого в методе add() пропишем логику, проверяющую, можно ли добавить еще один элемент без удаления первого элемента. Если нет – удалить первый элемент.



Создадим экземпляр этого класса, задав размер равным 3. Попробуем добавить четыре элемента. Выведем в консоль все элементы.



В консоли видим результат: первый элемент был удален.

