Национальный исследовательский университет

Высшая школа экономики

Дисциплина: Конструирование программного обеспечения

Контрольное домашнее задание за 4 модуль

Выполнено студентом:

группы 142-2

Смилянским А.А.

## Задача:

Предложить самостоятельно и реализовать средствами JDK8 пример обработки коллекции (коллекций) с применением потокового API (java.util.streams) с использованием промежуточных и терминальных (завершающих) операций и оценить время выполнения предложенной обработки при ее реализации разными средствами.

1. Последовательными потоками

2. Параллельными потоками

3. С использованием явных (внешних) циклов (в стиле старых JDK (<8))

4. Иными доступными средствами (fork/join framework и др.)

## Реализация:

Мной было решено сделать симулятор жизни офиса. Существует некоторый контейнер работников (офис), куда могут принимать новых работников, увольнять старых, а также, главное, назначать им выполнять некоторые задания. Заданий может быть несколько на каждого работника (они выстраиваются в очередь). Сами задания наследуются от абстрактного класса, где описаны основные методы. То есть, независимо от конкретного задания, оно должно быть выполнено всеми 4-ымя способами с измерением времени работы каждого. Данные было решено взять одни, чтобы не вводить много данных в программе, они подгружаются сами. Существует класс логера, для логирования из любого места программы. Существует класс для измерения времени из любого места программы. Существует класс подгрузки данных. За данные было решено взять с открытого репозитория анализа данных данные об алкогольных напитках (предварительно обработанные, без лишних данных). Существует .txt файл, подгружаемый при старте программы, откуда берутся основные сообщения программы.

\*Фишки (Features):

* Log.java - логирование из любого места программы
* MessageManager.java – подгрузка основных сообщений программы, для хранения их всех в одном файле с озможностью замены их не меняя кода (аналог Android strings, res … файлам)
* NicknameManager.java – генератор никнеймов с подгрузкой префикса и корня имени из файлов.
* TimeManager.java – получение данных о времени в любой момент времени из любого места программы.
* Task.java – абстрактный класс, представляющий собой задачу, которую может выполнить рабочий 4-ымя способами указанными в задании. Наследуясь от него можно быстро и эффективно создавать новые задачи.

## Формат данных:

WINE,CANADA,198267,3.0,COPPER MOON - MALBEC,30.99  
WINE,CANADA,305375,4.0,DOMAINE D'OR - DRY,32.99

1. класс напитка – string
2. страна производитель - string
3. id - string
4. ёмкость - double
5. название - string
6. цена - double

## Описание выполненных задач:

Основных task 2, но их можно дополнять быстро и эффективно: FindValues (поиск значений в контейнере для алкоголя), Sort (сортировка контейнера по выбранному аргументу).

После выполнения задания, работник покажет работу, это можно отключить кнопочкой «не показывать результат».

|  |  |  |
| --- | --- | --- |
| № | Наличие использования | Место выполнения |
| 1 | Коллекций из java.util.\* (в зависимости от количества), преобразование их в java.util.Stream | Sort, FindValues, Task |
| 2 | Промежуточные и терминальные (завершающие) операции |
| 3 | Оценка времени выполнения предложенной обработки при ее реализации разными средствами (см. перечень необходимого из п. 2). |
| 4 | Функциональных интерфейсов (Predicate, Supplier, …) – необходимо предоставить свои Lambda выражения, которые реализуют эти интерфейсы, присвоить Lambda переменной типа функционального интерфейса, а затем использовать эту переменную в своем демонстрационном коде | FindValues, Sort |
| 5 | Функциональные интерфейсы собственной разработки, которые помечены аннотацией @FunctionalInterface | SimpleLog |
| 6 | Графического интерфейса (JavaFX либо Java Swing) с примерами использования лямбд | Main, SortPain, FindPane |
| 7 | Ссылки на методы, также с использованием this и super | AlcoholContainer, Sort |
| 8 | Ссылки на конструкторы | Sort |
| 9 | Default-методы | SimpleUI |
| 10 | Static-методы в интерфейсах | SimpleUI |
| 11 | Демонстрация собственных реализаций accumulator и combiner в следующих методах (в том числе, при использовании parallelStream):  - Optional<T> reduce(BinaryOperator<T> accumulator);  - T reduce(T identityVal, BinaryOperator<T> accumulator);  - <U> U reduce(U identity, BiFunction<U, ? super T, U> accumulator, BinaryOperator<U> combiner); | FindValues |
| 12 | Реализация своих mapFunc для методов mapTo\* (mapToInt,…) | FindValues |
| 13 | Демонстрация  <R> R collect ( Supplier<R> target, BiConsumer<R, ? super T> accumulator, BiConsumer <R, R> combiner);  с использованием собственных accumulator и combiner | Sort, FindValues |
| 14 | Ant-cборки проекта (с исполняемым jar’ом) , либо Maven- проекта, организованного в среде IntelliJ IDEA | Build.xml |
| 15 | JUnit – тестов разработанного программного кода | Tests |
| 16 | Javadoc – комментариев с автоматической генерацией документации для разработанного кода | docs |

## Пример работы:

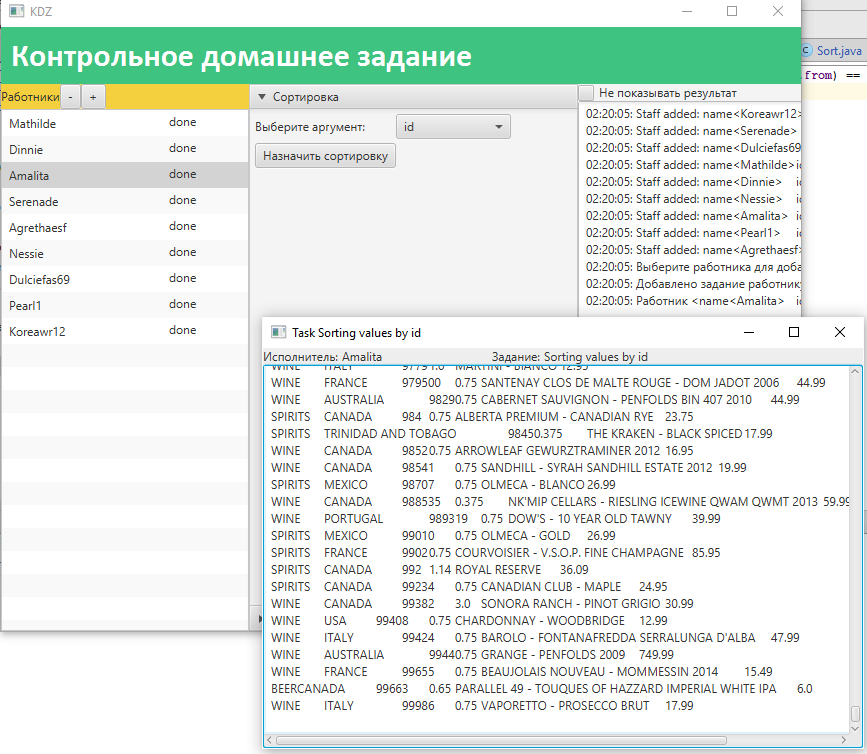


рис. 1 Выполненная сортировка по id

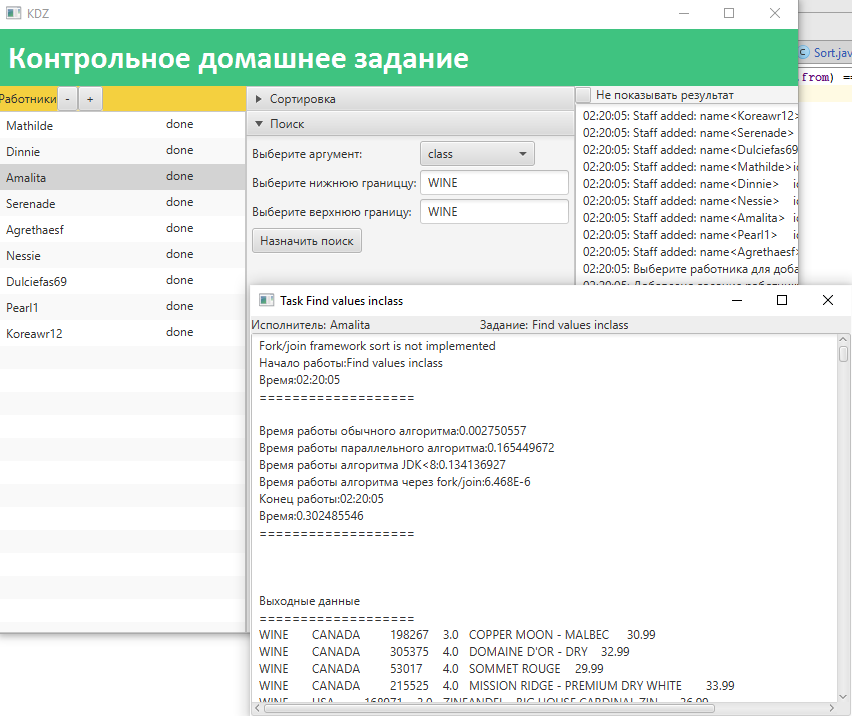


Рис. 2. Выполненный поиск по классу