Java Core
Финальный проект

Базовое финальное задание
- Описание задания
- Детали технической реализации
- Входные данные
- Выходной файл-отчет
- Документация
- Сохранение проекта

Усложненное финальное задание. Версия 1
Усложненное финальное задание. Версия 2
Использование в реальных кейсах
Что изучается и повторяется во время выполнения задания
Критерии проверки

Базовое финальное задание

Описание задания

Создать программу для выполнения денежного перевода с одного счета на другой. Предварительно в программе должен находится файл с номерами счетов и суммами на них. При запуске программа должна ожидать ввода информации из консоли.
При выборе функции парсинга программа должна парсить все подходящие файлы из каталога ‘input’ и перенести распаршеные файлы в другой каталог ‘archive’. В результате парсинга файлов программа должна сформировать/обновить файл-отчет и обновить информацию в файле с номерами счетов и суммами до актуальных.

Детали технической реализации

При запуске программа ожидает ввод информации:

в консоль введено 1 - вызов операции парсинга файлов перевода из input,
в консоль введено 2 - вызов операции вывода списка всех переводов из файла-отчета.

Программа должна обрабатывать файлы формата txt.
Если в каталоге есть файлы другого формата, то программа должна их пропускать и не обрабатывать. Предусмотреть различные кейсы и реализовать проверки.
Например, в каталоге нет ни одного подходящего файла, отрицательная сумма перевода, невалидные номер счетов и другие. Предусмотреть другие возможные случаи возникновения и обработки ошибок и исключений. Предусмотреть сохранение Java классов по “слоям”. Слоем будем считать просто каталог/package в исходном коде проекта. Например, model слой - каталог, в котором будут находится классы для модели данных (класс для счета и другие); exception слой - каталог, в котором будут находиться классы для работы с исключениями; Название каталогов(слоев) и их количество - на усмотрение студента.

Входные данные

Формат входных файлов: txt.
Файлы должны содержать поля:

- номер счета с (10 цифр XXXXX-XXXXX);
- номер счета на (10 цифр XXXXX-XXXXX);
- сумма для перевода (только целые числа).

Входной файл может содержать любое количество других полей, но во время обработки файла нужно получать информацию только из трех вышеописанных.
Студент является ответственным за создание входных файлов.

Выходной файл-отчет

Выходной файл-отчет должен содержать список обработанных операций с указанием файла, статуса операции, даты и время операции.
Пример выходного файла отчета:

дата-время | файл_1 | перевод с XXXXX-XXXXX на YYYYY-YYYYY 500 | успешно обработан
дата-время | файл_1 | перевод с XXXXX-XXXXX на YYYYY-YYYYY -100 | ошибка во время обработки, неверная сумма перевода

Выходной файл-отчет должен быть формата txt.
Выходной файл отчет генерируется после вызова операции обработки файлов-переводов. Выходной файл-отчет может быть сохранен в любом месте.

Документация

Создать диаграмму классов с описанием зависимостей одних классов от других.

Сохранение проекта

Проект должен быть сохранен в https://github.com/.
В проекте обязательно должен быть заполнен Readme файл.
Желательно заполнять Readme файл на английском языке.

Усложненное финальное задание. Версия 1

Выполнить базовую версию финального задания.
Сумма для перевода может быть с дробной частью.
Добавить дополнительную функциональность:

1. Для вывода истории обработанных записей из файла отчета по датам с ... по...

Усложненное финальное задание. Версия 2

Выполнить усложненное финальное задание версии 1.
Добавить загрузку результата парсинга в базу данных. Предварительно создать таблицу для хранения этой информации.

Использование в реальных кейсах

Данная программа представляет собой простую реализацию сервиса для переводов, что является очень частым практическим кейсом в реальных проектах.
Также задания такого вида часто используются как тестовые задания во многих компаниях.

Что изучается и повторяется во время выполнения задания

При выполнении данного задания студент повторяет весь пройденный материал Java Core и использует основные конструкции языка Java.
В частности, во время выполнения задания студент будет использовать управляющие конструкции, циклы, классы для работы с файлами и датами, коллекции, регулярные выражения, исключения и другие инструменты языка.
Сохранение информации в выходной файл и сохранение информации об ошибках можно рассматривать как подход к логированию.
В более сложных версия финального задания студент учится работать с базой данных (создавать базу данных, создавать таблицы и правильно хранить данные в таблицах).
Студент учится создавать документацию к проекту и строить диаграммы классов.

Критерии проверки

Для успешной защиты проекта достаточно демонстрации успешного выполнения базовой версии программы.
Не следует обращать внимание на ошибки при обработки “поломанных” файлов или файлов с неполными данными и других “ловушек”.