Тесты к курсу «Введение в программирование»

Ветка: **master**

Georgiy Korneev f122b8d45b Redundant file removed **месяцев назад: 7**

artifacts	f0481490a8 Compiled tests updated	месяцев назад: 7
java	f122b8d45b Redundant file removed	месяцев назад: 7
README.md	29d996e847 Homework 8 modification 2 added	месяцев назад: 7

README.md

Тесты к курсу «Введение в программирование»

Условия домашних заданий

Домашнее задание 9. WebCrawler

Условия:

- Ваш класс должен иметь имя `crawler.SimpleWebCrawler`
  - Класс должен реализовывать интерфейс WebCrawler
  - Конструктор класса должен принимать Downloader, используемый для загрузки страниц и картинок
- Прохождение боссов является не обязательным, по позволяет получить дополнительные баллы
  - Тестовый босс

Исходный код тестов:

- Обычные: WebCrawlerTest.java
  - Кэш
- Для боссов: WebCrawlerBossTest.java
- Для экспериментов: CachingDownloader.java

Домашнее задание 8. Markdown to HTML

Модификации

- *Mark*
  - Добавьте поддержку `~выделения цветом~`: `<mark>выделения цветом</mark>`
  - Исходный код тестов
  - Откомпилированные тесты
- *Underline*
  - Добавьте поддержку `++подчеркивания++`: `<u>подчеркивания</u>`
  - Исходный код тестов
  - Откомпилированные тесты

Исходный код тестов: Md2HtmlTest.java

Откомпилированные тесты: Md2HtmlTest.jar

Домашнее задание 7. Разметка

Модификации

- *HTML*
  - Дополнительно реализуйте метод `toHtml`, геререрующий HTML-разметку:
    - выделеный текст окружается тегом `em`;
    - сильно выделеный текст окружается тегом `strong`;
    - зачеркнутый текст окружается тегом `s`.
  - Исходный код тестов
- *TeX*
  - Дополнительно реализуйте метод `toTeX`, геререрующий TeX-разметку:
    - выделеный текст заключается в `\emph{` и `}`;
    - сильно выделеный текст заключается в `\textbf{` и `}`;
    - зачеркнутый текст заключается в `\textst{` и `}`.
  - Исходный код тестов

Исходный код тестов:

- Исходный код тестов

Домашнее задание 6. Подсчет слов++

Модификации

- *LineIndex*
  - В выходном файле слова должны быть упорядочены в лексикографическом порядке
  - Вместо номеров вхождений во всем файле надо указывать `<номер строки>:<номер в строке>`
  - Класс должен иметь имя `WordStatLineIndex`
  - Исходный код тестов
  - Откомпилированные тесты

Исходный код тестов:

- WordStatIndexTest.java
- WordStatIndexChecker.java

Откомпилированные тесты: WordStatIndexTest.jar

Домашнее задание 5. Быстрый реверс

Модификации

- *Минимум*
  - Рассмотрим входные данные как (не полностью определенную) матрицу, вместо каждого числа выведите минимум из чисел в его столбце и строке.
  - Класс должен иметь имя `ReverseMin`
  - Исходный код тестов
  - Откомпилированные тесты
- *Сумма*
  - Рассмотрим входные данные как (не полностью определенную) матрицу, вместо каждого числа выведите сумму чисел в его столбце и строке.
  - Класс должен иметь имя `ReverseSum`
  - Исходный код тестов
  - Откомпилированные тесты

Исходный код тестов:

- ReverseFastTest.java
- ReverseChecker.java

Откомпилированные тесты: ReverseFastTest.jar

Домашнее задание 4. Подсчет слов

Модификации

- *Words*
  - В выходном файле слова должны быть упорядочены в лексикографическом порядке
  - Класс должен иметь имя `WordStatWords`
  - Исходный код тестов
  - Откомпилированные тесты
- *Count*
  - В выходном файле слова должны быть упорядочены по возрастанию числа вхождений, а при равном числе вхождений – по порядку первого вхождения во входном файле
  - Класс должен иметь имя `WordStatCount`
  - Исходный код тестов
  - Откомпилированные тесты

Исходный код тестов:

- WordStatInputTest.java
- WordStatChecker.java

Откомпилированные тесты: WordStatInputTest.jar

Домашнее задание 3. Сумма чисел в файле

Модификации

- *Hex*
  - На вход подаются десятичные и шестнадцатеричные числа
  - Шестнадцатеричные числа имеют префикс `0x`
  - Ввод регистронезависим
  - Класс должен иметь имя `SumHexFile`
  - Исходный код тестов
  - Откомпилированные тесты
- *Abc*
  - На вход подаются числа, представленные буквами. Нулю соответствует буква `a`, единице – `b` и так далее
  - Ввод регистронезависим
  - Класс должен иметь имя `SumAbcFile`
  - Исходный код тестов
  - Откомпилированные тесты

Исходный код тестов:

- SumFileTest.java
- SumFileChecker.java

Откомпилированные тесты: SumFileTest.jar

Домашнее задание 2. Реверс

Модификации

- *Транспозиция*
  - Рассмотрим входные данные как (не полностью определенную) матрицу, выведите ее в транспонированном виде
  - Класс должен иметь имя `ReverseTranspose`
  - Исходный код тестов
  - Откомпилированные тесты
- *Максимум*
  - Рассмотрим входные данные как (не полностью определенную) матрицу, вместо каждого числа выведите максимум из чисел в его столбце и строке.
  - Класс должен иметь имя `ReverseMax`
  - Исходный код тестов
  - Откомпилированные тесты

Исходный код тестов:

- ReverseTest.java
- ReverseChecker.java

Откомпилированные тесты: ReverseTest.jar

Домашнее задание 1. Сумма чисел

Модификации

- *DoubleSpace*
  - Входные данные являются числами с формате с плавающей точкой
  - Класс должен иметь имя `SumDoubleSpace`
  - Числа разделяются пробелами-разделителями
  - Исходный код тестов
  - Откомпилированные тесты
- *BigIntegerDigit*
  - Входные данные помещаются в тип BigInteger
  - Класс должен иметь имя `SumBigInteger`
  - Числа имеют вид `[знак]цифры`
  - Исходный код тестов
  - Откомпилированные тесты

Для того, чтобы протестировать исходную программу:

1. Скачайте откомпилированные тесты (SumTest.jar)
2. Откомпилируйте `Sum.java`
3. Проверьте, что создался `Sum.class`
4. В каталоге, в котором находится `Sum.class` выполните команду `java -jar <путь к SumTest.jar>`
   - Например, если `SumTest.jar` находится в текущем каталоге, выполните команду `java -jar SumTest.jar`

Исходный код тестов:

- SumTest.java
- SumChecker.java
- Базовые классы