Билет 1. Менеджер строчек. Ядро

Реализовать ядро менеджера строчек.

Функциональность

Загрузка строчек из текстового файла.
Запись строчек в текстовый файл.
Предоставление RMI интерфейса к ядру:

- Добавление строки
- Удаление строки
- Список всех строк
- По набору строку предоставление информации о наличии каждой из них.
- Список запросов.

Технологии

I/O
RMI
Collections Framework

Билет 2. Менеджер строчек. Редактор

Реализовать GUI-редактор для менеджера строчек.

Функциональность

Отображения списка строчек
Добавление строчек
Удаление строчек.
Взаимодействие с ядром через RMI-интерфейс:

- Добавление строки
- Удаление строки
- Список всех строк

Технологии

Console I/O
RMI

Примечания

Обратить внимание на эффективную передачу данных по RMI.

Билет 3. Менеджер строчек. Statistics Analyzer

Реализовать GUI для отображения статистики обращений к менеджеру строчек.

Функциональность

Отображение Top-N строчек, для которых были осуществлены запросы.
Вывод статистики в файл в CSV-формате.

Технологии

Console I/O
RMI

Примечания

Обратить внимание на эффективную передачу данных по RMI.

Билет 4. Менеджер строчек. Query Tool

Реализовать GUI, позволяющий делать запросы к ядру менеджера строчек.

Функциональность

Задание запроса в виде набора строчек
Отображение результатов запроса в виде таблицы.
Взаимодействие с ядром через RMI-интерфейс:

- По набору строку предоставление информации о наличии каждой из них.

Технологии

Console I/O
RMI

Билет 5. Исполнитель

Программа должна читать из файла задания в формате

```
<время> <ip> <port>
```

И производить TCP-соединение в указанные моменты времени по указанным адресам.

Программа должна обеспечивать функционирования ядра планировщика заданий.

Функциональность

На одно время могут быть назначены несколько заданий, тогда они должны выполняться в разных потоках.

После выполнения соединения должен выводится результат (удалось/не удалось).

Потоки не должны создаваться для каждого задания.

Технологии

I/O

Net

Multithreading

Билет 6. Устойчивая сортировка

Прочитать скрипт из входного файла или консоли и выполнить его.

Функциональность

В скрипте встречаются следующие команды:

- `add <index> <string>` — добавить строку (не содержащую whitespaces) с заданным индексом.
- `remove <index>` — удалить все строки с заданным индексом.
- `print` — вывести строки, отсортированные по индексу, для строк с одинаковыми индексами — по времени добавления.

Каждая команда задается на отдельной строке.

Формат вывода:

```
<index> <string>
```

Индексы — целые числа в диапазоне -10^9—10^9.

Технологии

I/O

Collections Framework

Билет 7. Отношения классов

Программа должна для классов двух объектов вывести отношение между ними.

Функциональность

Поддерживаемые отношения:

- Совпадение
- Определение в одном пакете
- Один — предок другого
- Общий предок
- Список общих интерфейсов

Классы задаются во входном файле полными именами.

Технологии

Reflection
I/O

Билет 8. File Manager

Программа должна позволять просматривать файлы и управлять ими. Ввод/вывод должен осуществляться с консоли.

Функциональность

Должны поддерживаться команды:

- dir — вывести оглавление текущего каталога
- rm <имя файла> — удалить файл
- cd <директория> — перейти в заданную директорию.
- create <имя файла> — создать файл с заданным именем.
- mkdir <директория> — создать директорию
- rmdir <директория> — удалить директорию (рекурсивно).

Полные сообщения о результатах и ошибках

Технологии

I/O

Билет 9. TCP Proxy

Программа должна перенаправлять TCP-соединения.

При запуске программа читает таблицу перенаправлений из файла формата:

```
<local port> <remote ip-address> <remote port>
```

В процессе работы перенаправления должны осуществлять 10 потоков.

Функциональность

Чтение таблицы перенаправлений.

Реализация перенаправлений.

Технологии

Net

Multithreading

Билет 10. UDP Proxy

Программа должна перенаправлять UDP-датаграммы.

При запуске программа читает таблицу перенаправлений из файла формата:

```
<local port> <remote ip-address> <remote port>
```

В процессе работы перенаправления должны осуществлять несколько потоков.

Функциональность

Чтение таблицы перенаправлений.

Реализация перенаправлений.

Технологии

Net

Multithreading

Билет 11. Автоматический переводчик

Программа читает входной файл, переводит его по словарю и пишет результат в выходной файл.

Формат словаря:

```
<слово или выражение> | <перевод слова или выражения>
```

Пример словаря

```
hello | здравствуй
friend | друг
I am | я
…
```

Примечания:

При переводе регистр букв игнорируется.

Если перевода нет в словаре – слово выводится без перевода.

Если есть несколько подходящих вариантов перевода, выбирается вариант, с максимальной длиной левой части.

Функциональность

Чтение словаря.

Чтение входного файла.

Перевод.

Запись выходного файла.

Технологии

I/O

Collections Framework

Билет 12. Bag

Реализуйте коллекцию, представляющую мультимножество (`Bag`).

Функциональность

Реализация интерфейса `Collection`.

При итерации равные элементы должны идти подряд.

Равные элементы должны сохранять identity, а не представляться счетчиками.

Метод `remove` должен удалять любой из элементов, равный аргументу.

Операции доступа (в том числе через итераторы) и изменения должны выполняться асимптотически эффективно.

Технологии

Collections Framework

Generics

Билет 13. LinkedBag

Реализуйте коллекцию, представляющую мультимножество (`LinkedBag`).

Функциональность

Реализация интерфейса `Collection`.

При итерации элементы должны идти в порядке их добавления в коллекцию.

Равные элементы должны сохранять identity, а не представляться счетчиками.

Метод `remove` должен удалять любой из элементов, равный аргументу.

Операции доступа (в том числе через итераторы) и изменения должны выполняться асимптотически эффективно.

Технологии

Collections Framework

Generics

Билет 14. Менеджер плагинов

Программа должна загружать, выгружать и использовать плагины, реализующие интерфейс `Runnable`.

Программа управляется из командной строки следующими командами:

Загрузить плагин
```
load <имя> <класс> <classpath>
```
Запустить плагин
```
run <имя>
```
Выгрузить плагин
```
unload <имя>
```

При попытке загрузки плагина с дублирующимся именем, запуска или выгрузки несуществующего плагина, и прочих проблем должны выводиться сообщения об ошибках.

Функциональность

Загрузка плагина.

Выгрузка плагина.

Запуск плагина.

Технологии

I/O

Reflection

Билет 15. Memorizer

Написать многопоточный класс, вычисляющий функции и кэширующий результаты вычислений (в том числе, многопоточных).

Пример использования

```
Function<String, Integer > f = …;
f = new Memorizer<String, Integer >(f);
f.apply("2+3"); // Производится вычисление
f.apply("3+3"); // Производится вычисление
f.apply("2+3"); // Ответ достается из кэша
```

Функциональность

Поддержка функций с одним аргументом.

Каждое значения должно вычисляться не более одного раза (даже, если оно одновременно запрошено в нескольких потоках).

Применение должно быть прозрачно для пользователя.

Технологии

Concurrency Utilities

Билет 16. Ленивый список

Реализуйте список `LazyList`, который вычисляет свои элементы только при необходимости.

Функция вычисления элементов задается экземпляром интерфейса

```
interface ElementCalculator<T> {
    T calculate(int index);
}
```

Функциональность

Элемент списка должен вычисляться только при доступе к нему.

Каждый элемент списка должен вычисляться не более одного раза.

Класс `LazyList` должен быть thread safe.

Размер списка является неизменяемым и задается при создании списка.

Возможность одновременного вычисления нескольких элементов различными потоками.

Технологии

Collections Framework

Concurrency Utilities

Билет 17. Перекодировка email

Напишите программу, которая дешифрует сообщения, которые были несколько раз перекодированы. При перекодировании, текст считается бинарным файлом, в какой-то кодировке (не обязательно правильной) и перекодируется в какую-то другую кодировку.

Программа должна принимать пять аргументов командной строки:

1. Имя файла, содержащего список поддерживаемых кодировок, по одной на строке.
2. Имя файла, который требуется перекодировать.
3. Имя итогового файла.
4. Кодировка итогового файла.
5. Возможная глубина перекодирования (`D`) — целое число от 1 до 10.

Можно рассчитывать на то, что в исходном сообщении была подстрока `Привет`.

Функциональность

Программа должна раскодировать сообщение и записать его в итоговый файл в указанной кодировке.

При декодировании должны быть опробованы все варианты из не более чем `D` перекодирований.

Если раскодировать сообщение не удалось, программа должна выводить сообщение об ошибке.

Использовать промежуточные файлы запрещается.

Технологии

IO

Exception Handling

Билет 18. Tracing Proxy

Создать `TracingProxy`, который по объекту создает другой объект, реализующий все интерфейсы первого объекта и протоколирующий вызовы всех методов.

У каждого `TracingProxy` должна быть глубина (целое число). Если глубина больше нуля, то значения, возвращаемые методами должны оборачиваться в `TracingProxy` с глубиной на единицу меньше. Если глубина равна нулю, то возвращаемые значения передаются как есть.

Функциональность

Оборачивание исходного объекта с указанием глубины.
Оборачивание возвращаемых значений.
Прозрачность для исключений.
Правильная обработка примитивных типов.
Протоколирование на консоль:

- вызова метода, с указанием его имени и значений аргументов;
- возврата из метода и возвращаемого значения метода (если есть);
- завершение метода по исключению;

Технологии

Reflection

Билет 19. Обработка файла

Программа должна прочитать входной файл, заменить каждый символ табуляции на четыре пробела и вывести результат в тот же файл.

Функциональность

При сбое программы в том числе, из-за питания компьютера, исходный файл не должен быть потерян.

Технологии

I/O
Exception handling

Билет 20. Очередь заданий

Написать программу, которая создает два потока, взаимодействующих через очередь сообщений. Первый поток должен добавлять значения в очередь в случайные моменты времени, а второй доставать их из очереди, через заданные промежутки времени.

Функциональность

Потоки должны выводить на консоль информацию о добавлении в очередь и изъятии из нее.
Очередь, ни в какой момент времени, не должна содержать более 10 элементов.

Технологии

Multithreading
Collections Framework
Не использовать Concurrency Utilities

Билет 21. Тараканьи бега

Программа должна эмулировать тараканьи бега. В бегах участвуют N тараканов, которые бегут M этапов.

Каждый таракан представляется отдельным потоком, который сортирует случайный массив из 100000 целых чисел. За этап, таракану, закончившему первым начисляется N очков, второму N-1 очков и так далее.

Очки за отдельные этапы суммируются. Следующий этап начинается сразу же, когда все тараканы закончат предыдущий.

Функциональность

Реализация классов «таракана» и «судьи» (по одному классу).

После завершения каждого этапа судья должен выводить статистику по этапу и по уже пройденным этапам.

Технологии

Multithreading (без Concurrent Utilities)

Билет 22. Обобщенные матрицы

Реализуйте матрицы, на основе произвольных элементов. Тип элемента матрицы должен являться параметром типа.

Функциональность

Чтение матрицы из файла.
Сложение матриц.
Умножение матриц.
Транспонирование.
Запись матрицы в файл.

Технологии

Generics
I/O

Билет 23. Chat Client

Сделать приложение-чат. Чат должен подсоединяться к серверу по сети, передавать и получать сообщения.

При запуске клиента в командной строке ему передается имя пользователя, которое должно быть сообщено серверу при подключении.

Полученные сообщения должны отображаться в консоли с префиксом в виде имени пользователя, написавшего сообщение и символа «>».

Клиент должен корректно работать с русским языком.

Функциональность

Подсоединение к серверу.
Передача сообщений
Прием сообщений

Технологии

Networking
I/O

Билет 24. Chat Server

Сделать чат-сервер. К серверу по сети должны присоединяться клиенты, передающие и получающие сообщения.

Передаваемые сообщения должны транслироваться всем клиентам.

При подключении клиента, должно передаваться имя пользователя. Если пользователь с таким именем активен, то подключение запрещается.

Функциональность

Подсоединение к клиенту.
Прием сообщений
Передача сообщений

Технологии

Networking
I/O

Билет 25. TODO List

Сделать приложение, реализующее список задач.

Функциональность

Загрузка списка задач из файла.
Запись списка задач в файл.
Операции с задачами

- добавление задачи;
- удаление задачи;
- пометка задачи как выполненной.

Просмотр задач:

- всех;
- выполненных;
- невыполненных.

Интерактивный ввод-ввод осуществляется с консоли.

Технологии

I/O
Collections Framework

Билет 26. Markdown2HTML

Написать программу, преобразующую Markdown-разметку (http://en.wikipedia.org/wiki/Markdown) в (X)HTML.

Функциональность

Поддержка ввода данных из файла и стандартного ввода.
Поддержка вывода данных в файл и стандартный вывод.
Поддержка указания кодировки входного и выходного файлов (возможной различной).

Технологии

I/O
Collections Framework

Билет 27. Wiki2HTML

Написать программу, преобразующую Wiki-разметку в (X)HTML.

Функциональность

Поддержка ввода данных из файла и стандартного ввода.
Поддержка вывода данных в файл и стандартный вывод.
Поддержка указания кодировки входного и выходного файлов (возможной различной).

Технологии

I/O
Collections Framework

Билет 28. Сетевые крестики-нолики

Написать программу, которая позволит играть в крестики-нолики на поле 3x3 по сети.

Функциональность

Установка соединения между игроками.
Возможность поведения нескольких партий подряд.
Интерактивный ввод-вывод на консоль (включая отображение текущего состояния поля).

Технологии

I/O
Collections Framework

Билет 29. Sort

Напишите аналог утилиты `sort`.

Функциональность

Поддерживаемые опции:

- `--ignore-leading-blanks`
- `--dictionary-order`
- `--ignore-case`
- `--general-numeric-sort`
- `--ignore-nonprinting`
- `--numeric-sort`
- `--reverse`

Ввод данных с консоли или входного файла.

Исходные данные могут **не поместиться** в памяти.

Технологии

I/O

Collections Framework

Билет 30. Parallel MergeSort

Реализуйте параллельную версию сортировки слиянием.

Функциональность

Сортировка данных с естественным и внешним порядком.
Указание числа потоков при запуске.
Не более одного дополнительного массива.
Программа, демонстрирующая работу сортировки.

Технологии

Generics
Multithreading

Билет 31. Parallel QuickSort

Реализуйте параллельную версию быстрой сортировки.

Функциональность

Сортировка данных с естественным и внешним порядком.
Указание числа потоков при запуске.
Не более одного дополнительного массива.
Приложение для демонстрации сортировки.

Технологии

Generics
Multithreading

Билет 32. Java Shell

Реализуйте аналог оболочки (`sh`/`bash`/`cmd`) на Java.

Функциональность

Консольный ввод-вывод.

Запуск программ с аргументами, например: “`java –cp . Hello.java`».

Перенаправление из/в файл, между программами, например: «`input > sort –r | uniq –c > output`».

Условное исполнение программ, например: «`test–f file && truncate -s 0 file || touch file`».

Перенаправления и условное исполнение должны быть реализованы средствами Java.

Технологии

I/O

Multithreading

Билет 33. Протоколирование

Реализуйте набор классов для ведения протоколов.

Функциональность

Создайте интерфейс `Logger` с методами:

- `log(String message)`
- `log(String message, Throwable cause)`

Создайте две реализации: `FileLogger`, лог в текстовый файл и `HTMLLogger`, записывающий лог в HTML файл.

Создайте класс `CompositeLogger`, содержащий набор логгеров и передающий входящие вызовы им всем.

Добавьте возможность задать специфичный формат записи лога. Например, включать в нее текущее время или дату.

Добавьте возможность задать важность события: `debug`, `info`, `warning` или `error` и реализуйте возможность фильтровать сообщения, уровень важности которых не ниже заданного.

Протоколирование должно безошибочно и эффективно работать в многопоточной среде.

Сделайте программу, иллюстрирующую работу созданных классов.

Технологии

I/O

Multithreading

Билет 34. Конвертация файла

Программа должна позволять выбрать два файла, прочитать первый из них в одной кодировке и вывести в другой кодировке.

Функциональность

Выбор файлов должен быть осуществлен в использованием `JFileChooser`.

Поддерживаемые кодировки:

- KOI8-R
- Cp1251
- Cp866
- UTF8
- UTF16

Технологии

I/O

Swing

Система Лейтнера

Реализуйте интервальное повторение переводов слов на основе системы Лейтнера.

Функциональность

- Во входном файле даны пары слово (словосочетание) — перевод.
- Слова распределены на 10 корзин. Вероятность вытащить каждое слово из корзины n пропорциональна 1.5^n.
- Если пользователь ввел правильный перевод, то слово перемещается в корзину с номером на единицу больше (или остается в 10 корзине). Если пользователь ошибся, то слово перемещается в первую корзину.
- Если пользователь ввел пустой ответ, то программа должна завершиться, и при повторном запуске продолжить с того же места (с сохранением распределения слов по корзинам).
- В любой момент пользователь может посмотреть распределение слов по корзинам.
- Поддержка русского и английского языков

Технологии

- I/O
- I18n и I10n

Мат ладьёй и королём

Реализуйте программу, ставящую мат пользователю в позиции вида "ладья и король против короля".

Функциональность

- Исходное положение фигур задаётся в качестве аргументов командной строки.
- Программа играет за белых и делает первый ход.
- Вывод текущего положения до и после хода пользователя.
- Пользователь играет за чёрных. Ходы вводятся в алгебраической нотации.
- Поддержка русского и английского языков.

Технологии

- I/O
- I18n и I10n