Вариант 1

Распространенные операции Stream. Реализуйте:

a. Функцию сложения чисел, то есть int addUp(Stream<Integer> numbers).;
b. Функцию, которая получает исполнителя и возвращает список строк, содержащих имена и место происхождения.
c. Функцию, которая получает альбомы и возвращает список альбомов, содержащих не более трех произведений.

Вариант 2

Итерирование. Перепишите этот код с использованием внутреннего итерирования вместо внешнего.

```
int totalMembers =0;
for (Artist artist : artists){
    Stream<Artist > members = artist.getMembers ();
    totalMembers += members. count ();
}
```

Вариант 3

Вычисление. Взгляните на сигнатуры следующих методов интерфейса Stream. Являются они энергичными или отложенными?

a. boolean anyMatch (Predicate<? super T> predicate) ;
b. Stream<T> limit (long maxSize) ;

Вариант 4

Функции высшего порядка. Являются ли следующще методы Stream функциями высшего порядка? Почему?

a. boolean anyMatch (Predicate<? super T> predicate) ;
b. Stream<T> limit (long maxSize) ;

Вариант 5

Чистые функции. Является ли следующее лямбда-выражение свободным от побочных эффектов, или оно изменяет состояние?

x -> x+1

Рассмотрим пример кода:

AtomicInteger count = new AtomicInteger (0) ;

List<String> origins = album. musicians()

.forEach (musician -> count.incAndGet ();)

Лямбда-выражение, переданное в forEach в данном примере.

Вариант 6

Подчитайте количество строчных букв в строке (подсказка: воспользуйтесь методом chars класса String).

Вариант 7

Пусть дан список строк List<String>. Найдите в нем строку, содержащую максимальное число строчных букв. Чтобы код правильно работал, когда входной список пуст, можете возвращаться объект типа optional<String>.

Вариант 8

Обедающие философы

Пять безмолвных философов сидят вокруг круглого стола, перед каждым философом стоит тарелка спагетти.

Вилки лежат на столе между каждой парой ближайших философов.

Каждый философ может либо есть, либо размышлять.

Философ может есть только тогда, когда держит две вилки — взятую справа и слева.

Взятие каждой вилки и возвращение её на стол являются раздельными действиями, которые должны выполняться одно за другим.

Вариант 9

Напишите код, в котором создаются и запускаются на выполнение потоки S и W. Поток S выполняет переключение с задержкой 1000 миллисекунд из состояния true в состояние false и наоборот. Поток W ожидает состояния true потока S, выводит на консоль обратный отсчет от 30 с задержкой 100 миллисекунд и приостанавливает свое действие, как только поток S переключен в состояние false. Условием завершения работы потоков является достижение отсчета нулевой отметки.

Вариант 10

Всего на автозаправке 4 колонки. Среднее время заправки одной машины {6 мин.}. колонки не простаивают. Через каждые {45 мин.} работы колонки закрываются на техническое обслуживание: первая– на 10 мин, вторая – на 15 мин., третья – на 5 мин., четвертая– на 13 мин. Необходимо разработать многопоточное приложение, которое позволяет сделать следующее:

a) вводятся с клавиатуры все данные, отмеченные в условии задачи как {..};
b) выводится количество автомобилей, которое будет обслужено всеми колонками за {4} часа;
c) выводится количество автомобилей, которое обслужит каждая колонка за {4} часа.

Вариант 11

Дано два потока — производитель и потребитель. Производитель генерирует некоторые данные (в примере — числа). Производитель «потребляет» их.

Два потока разделяют общий буфер данных, размер которого ограничен. Если буфер пуст, потребитель должен ждать, пока там появятся данные. Если буфер заполнен полностью, производитель должен ждать, пока потребитель заберёт данные и место освободится.

Вариант 12

Напишите программу на Java для реализации параллельного веб-сканера, который выполняет одновременный обход нескольких веб-сайтов с использованием потоков.

Вариант 13

Напишите Java-программу, которая создает банковский счет с одновременным пополнением и снятием средств с помощью потоков.

Вариант 14

Напишите Java-программу для демонстрации использования класса ForkJoinPool для параллельного выполнения рекурсивных задач.

Вариант 15

Напишите Java-программу, которая использует интерфейс ScheduledExecutorService для планирования выполнения задач в указанное время или с фиксированной задержкой.

Вариант 16

Напишите Java-программу, которая использует класс Concurrentlink Queue для реализации потокобезопасной очереди.

Вариант 17

Напишите программу, которая каждую секунду отображает на экране данные о времени, прошедшем от начала сессии, а другой её поток выводит сообщение каждые 5 секунд. Предусмотрите возможность ежесекундного оповещения потока, воспроизводящего сообщение, потоком, отсчитывающим время. Не внося изменений в код потока-"хронометра" , добавьте ещё один поток, который выводит на экран другое сообщение каждые 7 секунд.

Вариант 18

В директории лежат входные текстовые файлы, проименованные следующим образом: in_<N>.dat, где N - натуральное число. Каждый файл состоит из двух строк. В первой строке - число, обозначающее действие, а во второй - числа с плавающей точкой, разделенные пробелом.

Действия могут быть следующими:

1 - сложение

2 - умножение

3 - сумма квадратов

Необходимо написать многопоточное приложение, которое выполнит требуемые действия над числами и сумму результатов запишет в файл out.dat. Название рабочей директории передается в виде аргумента рабочей строки. В реализации приветствуется использование полиморфизма и паттернов проектирования.

Вариант 19

Разработайте многопоточное приложение, выполняющее вычисление произведения матриц A (m×n) и B (n×k). Элементы cij матрицы произведения C = A×B вычисляются параллельно p однотипными потоками. Если некоторый поток уже вычисляет элемент cij матрицы C, то следующий приступающий к вычислению поток выбирает для расчета элемент cij+1, если j<k, и ci+1k, если j=k. Выполнив вычисление элемента матрицы-произведения, поток проверяет, нет ли элемента, который еще не рассчитывается. Если такой элемент есть, то приступает к его расчету. В противном случае отправляет (пользовательское) сообщение о завершении своей работы и приостанавливает своё выполнение. Главный поток, получив сообщения о завершении вычислений от всех потоков,

выводит результат на экран и запускает поток, записывающий результат в конец файла-протокола. В каждом потоке должна быть задержка в выполнении вычислений (чтобы дать возможность поработать всем потокам). Синхронизацию потоков между собой организуйте через критическую секцию или мьютекс.

Вариант 20

Задача в терминах C++. При нажатии кнопки "Start" диалоговое приложение запускает консольное приложение. Последующие нажатия кнопки "Start" должны привести к созданию в консольном приложении N новых рабочих потоков, где N - значение из поля с числовым счетчиком.

Нажатие кнопки "Stop" должно привести к закрытию последнего созданного рабочего потока в консольном приложении, а в случае отсутствия рабочих потоков - к его завершению. Консольное приложение также должно завершиться и при завершении диалогового. Диалоговое приложение должно отслеживать, закрыл ли пользователь консольное приложение самостоятельно. Следующее после закрытия консоли нажатие "Start" возобновляет рабочий цикл.

После запуска консоли выпадающий список содержит строки "Все потоки" и "Главный поток", по мере создания новых потоков в него добавляются строки, содержащие их номера (разумеется, при удалении потоков удаляются и соответствующие им строки). Корректировка выпадающего списка осуществляется лишь после получения подтверждения от консоли об успешном создании потоков.

Взаимодействие между приложениями реализовать с помощью объектов ядра "события" (events).

Добавить в приложения передачу информации с помощью файла, отображаемого в память (memory mapped file). Реализовать транспортные функции в виде динамически подключаемой библиотеки с неявной загрузкой.

При нажатии кнопки "Send" диалоговое приложение пересылает консоли текст, введенный в поле ввода. Выпадающий список задает поток-адресат. При выборе строки "Все потоки" сообщение отправляется всем потокам, каждый поток считывает информацию независимо.

Главный поток выводит полученный текст на стандартный вывод (stdout), рабочие потоки создают текстовые файлы с именами <свой номер>.txt и дописывают в них полученный текст. Файл, отображаемый в память, защищается от совместного использования с помощью объекта синхронизации mutex.

Все действия подтверждаются сообщениями от консоли.

Вариант 21

Суть алгоритма поликлиники заключается в том, что при выборе потоком себе номера, он не обращается к общему счетчику, а выбирает наибольший среди тех, что имеются у других потоков. При определении следующей очереди, так же как и в предыдущем алгоритме, ее получает поток с наименьшим номером.

Вариант 22

Необходимо реализовать многопоточное приложение, которое решает следующую задачу:

Есть два типа пользователя (два типа потока). Один - Хозяин, имеет в своем арсенале список вещей (Вещь: цена и вес), второй - Вор, имеет рюкзак (Рюкзак: предельный вес, который может в себя вместить). Поток Хозяина выполняет работу по выкладыванию вещей в квартиру. Поток Вора - забирает вещи из квартиры. При этом Вор должен забрать такие вещи, чтобы их ценность была максимальной и вес их должен быть меньше предельного веса, который может поместиться в рюкзак.

Объектные модели:

1. Вещь; атрибуты: вес, ценность
2. Хозяин; атрибуты: Вещи; действия: внести вещи в квартиру
3. Рюкзак; атрибуты: предельный вес
4. Вор; атрибуты: рюкзак. Действия: сложить вещи в рюкзак.

Ограничения:

1. Если работает поток Хозяина, то вор не должен класть вещи в рюкзак.
2. Если работает Вор, то Хозяин не может войти в квартиру

Возможные ограничения системы:

1. Хозяев может быть 1..n.
2. потоки Хозяев БЕЗ взаимной блокировки: несколько хозяев могут выкладывать вещи в квартиру одновременно
3. Воров может быть 1..m.
4. Потоки Воров со ВЗАИМНОЙ блокировкой: воровать одновременно может только 1 вор."