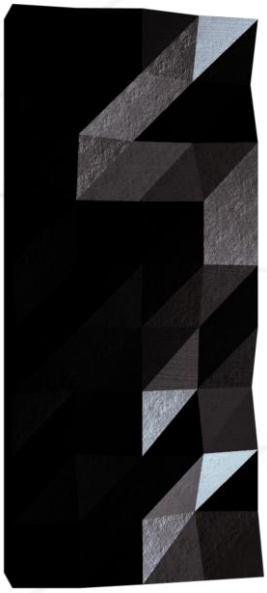

Модуль 03 - Piscine Java

Нитки

Резюме: Сегодня вы узнаете, как использовать основные механизмы многопоточности в Java

Содержание

Глава I

Предисловие

- Любое современное приложение клиент/сервер основано на потоках.
- Потоки реализуют концепцию асинхронных операций, когда несколько слабо связанных между собой задач выполняются "параллельно".
- Многопоточность в приложениях клиент/сервер позволяет перевести некоторые задачи в режим выполнения "back- ground", чтобы клиенту не приходилось ждать ответа сервера. Например, как только вы указали свой e-mail на сайте, сразу же отображается страница, информирующая о том, что сообщение с подтверждением было отправлено на ваш e-mail, независимо от того, сколько времени займет отправка сообщения на ваш e-mail в параллельном потоке.
- Каждый ваш запрос на веб-сайте выполняется в отдельном, независимом потоке на сервере.
- Поведение потоков управляется операционной системой и процессором.
- Поведение потоков недетерминировано. Вы никогда не знаете, какой поток будет запущен в определенный момент, даже если вы перезапустите один и тот же многопоточный код.
- Советы по работе с потоками можно найти в классе Object.
- Нити - любимая тема на собеседованиях с младшими специалистами.

Глава II

Инструкции

- Используйте эту страницу как единственную ссылку. Не слушайте никаких слухов и домыслов о том, как приготовить раствор.
- Теперь для вас существует только одна версия Java - 1.8. Убедитесь, что компилятор и интерпретатор этой версии установлены на вашей машине.
- Вы можете использовать IDE для написания и отладки исходного кода.
- Код чаще читают, чем пишут. Внимательно прочитайте документ, в котором приведены правила форматирования кода. При выполнении каждого задания убедитесь, что вы следуете общепринятым стандартам Oracle
- Комментарии не допускаются в исходном коде вашего решения. Они затрудняют чтение кода.
- Обратите внимание на разрешения ваших файлов и каталогов.
- Чтобы ваше решение было оценено, оно должно находиться в вашем GIT-репозитории.
- Ваши решения будут оценивать ваши товарищи по аквариуму.
- Вы не должны оставлять в своем каталоге никаких других файлов, кроме тех, которые явно указаны в инструкциях к упражнению. Рекомендуется изменить свой .gitignore во избежание несчастных случаев.
- Когда вам нужно получить точный вывод в ваших программах, запрещено выводить предварительно рассчитанный вывод вместо правильного выполнения упражнения.
- У вас есть вопрос? Спросите своего соседа справа. В противном случае попробуйте поговорить с соседом слева.
- Ваше справочное пособие: товарищи / Интернет / Google. И еще кое-что. На любой ваш вопрос есть ответ на Stackoverflow. Научитесь правильно задавать вопросы.
- Внимательно прочитайте примеры. В них могут потребоваться вещи, которые не указаны в предмете.
- Используйте "System.out" для вывода

- И да пребудет с вами Сила!
- Никогда не оставляйте на завтра то, что вы можете сделать сегодня ;)

Глава III

Упражнение 00: Яйцо, курица... или человек?

	Упражнение 00
Яйцо, курица... или человек?	
Входящий каталог : *ex00/*	
Файлы для сдачи : *.java	
Разрешенные функции : Все Рекомендуемые типы и их методы : Object, Thread, Runnable	

Истина рождается в споре - предположим, что каждый поток дает свой ответ. Права та нить, за которой осталось последнее слово.
Необходимо реализовать работу двух потоков. Каждый из них должен вывести свой ответ на экран несколько раз, например, 50:

```
$ java Program --count=50
Egg
Кур
очк
а
Ряб
а
...
Яйцо
```

В этом случае побеждает поток egg. Однако программа также содержит основной поток. Внутри потока выполняется метод public static void main(String args[]). Нам нужно, чтобы в конце выполнения программы этот поток отобразил все свои ответы. Таким образом, окончательный вариант выглядит следующим образом:

```
$ java Program --count=50
Egg
Кур
очк
а
Ряб
а
...
Яй
цо
кур
ино
е
...
Человек
...
...
Человек
```

- Это означает, что программа 50 раз выводит сообщение Human, которое печатает главный поток.

Глава IV

Упражнение 01 : Яйцо, курица, яйцо, курица...

	Упражнение 01
Яйцо, курица, яйцо, курица...	
Входящий каталог : *ex01/*	
Файлы для сдачи : *.java	
Разрешенные функции : Все Рекомендуемые типы и их методы : Object, Thread, Runnable Ключевые слова : Synchronized	

Давайте организуем спор. Теперь каждый поток может дать свой ответ только после того, как это сделает другой поток. Предположим, что поток egg всегда отвечает первым.

```
$ java Program --count=50
Egg
Курино
е
яйцо
Курино
е
яйцо
Курино
е
яйцо
...
```

Примечание:

- Для решения этой задачи мы рекомендуем изучить принцип работы модели Producer-Consumer

Глава V

Упражнение 02 : Реальная многопоточность

	Упражнени е 02
Реальная многопоточность	
Входящий каталог : *ex02/*	
Файлы для сдачи : *.java	
Разрешенные функции : Все Рекомендуемые типы и их методы : Object, Thread, Runnable Ключевые слова : Synchronized	

Старайтесь использовать многопоточность по назначению: распределять вычисления по программе.

Предположим, что имеется массив целочисленных значений. Ваша цель - вычислить сумму элементов массива, используя несколько потоков "суммирования". Каждый поток вычисляет определенную секцию внутри массива. Количество элементов в каждой секции постоянно, за исключением последней (ее размер может отличаться в большую или меньшую сторону).

Массив каждый раз генерируется случайным образом. Длина массива и количество потоков передаются в качестве аргументов командной строки.

Чтобы убедиться, что программа работает правильно, мы должны начать с вычисления суммы элементов массива стандартным методом.

Максимальное количество элементов массива - 2,000,000. Максимальное количество потоков не больше текущего количества элементов массива. Максимальное значение модуля каждого элемента массива - 1 000. Все данные гарантированно достоверны.

Пример работы программы (каждый элемент массива равен 1):

```
$ java Program --arraySize=13 --threadsCount=3
Сумма: 13
Нить 1: от 0 до 4 сумма равна 5
Нить 2: от 5 до 9 сумма равна 5
Нить 3: от 10 до 12 сумма равна 3
Сумма по нитям: 13
```

Примечание:

- В приведенном выше примере размер последней секции суммирования, используемой третьим потоком, меньше других.
- Нити могут выдавать результаты работы непоследовательно.

Глава VI

Упражнение 03: Слишком много нитей...

	Упражнение 03
Слишком много ниток...	
Входящий каталог : *ex03/*	
Файлы для сдачи : *.java	
Разрешенные функции : Все Рекомендуемые типы и их методы : Object, Thread, Runnable Ключевые слова : Synchronized	

Предположим, что нам нужно загрузить список файлов из сети. Некоторые файлы загружаются быстрее, а другие - медленнее.

Для реализации этой функциональности мы, очевидно, можем использовать многопоточную загрузку, где каждый поток загружает определенный файл. Но что делать, если файлов слишком много? Большое количество потоков не может быть запущено одновременно. Поэтому многие из них будут ждать.

Кроме того, следует помнить, что постоянное создание и завершение потоков - это очень затратная операция, которой следует избегать. Логичнее запустить сразу N потоков, и когда один из них закончит загрузку файла, он сможет заняться следующим файлом в очереди.

Нужно создать файл files_urls.txt (имя файла должно быть явно указано в коде программы), в котором указать список URL-адресов файлов, которые нужно, например, скачать:

1 https://i.pinimg.com/originals/11/19/2e/11192eba63f6f3aa591d3263fdb66bd5.jpg
2 https://pluspng.com/img-png/balloon-hd-png-balloons-png-hd-2750.png
3 https://i.pinimg.com/originals/db/a1/62/dba162603c71cac00d3548420c52bac6.png
4 https://pngimg.com/uploads/balloon/balloon_PNG4969.png
5 http://tldp.org/LDP/intro-linux/intro-linux.pdf

Пример работы программы:

```
$ java Program.java --threadsCount=3
Thread-1 start скачать файл номер 1
Thread-2 start скачать файл номер 2
Thread-1 finish скачать файл номер 1
Thread-1 start скачать файл номер 3
Thread-3 start скачать файл номер 4
Thread-1 finish скачать файл номер 3
Thread-2 finish скачать файл номер 2
Thread-1 start скачать файл номер 5
Thread-3 finish скачать файл номер 4
Thread-1 finish скачать файл номер 5
```

Примечания:

- Вывод, созданный реализованной программой, может отличаться от иллюстрации.
- Каждый файл загружается только один раз одним потоком.
- Программа может содержать "бесконечный цикл" без условия выхода (в этом случае программу можно завершить, прервав процесс).