Практика по Java, часть 2

Пара 1: Задача про Lazy

Юрий Литвинов
yurii.litvinov@gmail.com

10.04.2019г

Правила игры

- Как обычно, куча домашек, две контрольные (и переписывание в конце), баллы и дедлайны, HwProj
  - Будет больше задач на паре
- Система оценивания, скорее всего, останется такой же
- Будет про многопоточные, сетевые, сетевые И многопоточные приложения, пользовательский интерфейс и т.д.

Напоминание про штрафы

Пропущенный дедлайн	-10
Задача на момент дедлайна не реализует все требования условия	пропорционально объёму невыполненных требований
Замечания не исправлены за неделю после их получения или в следующей попытке сдачи	-2
Неумение пользоваться гитом	-2
Проблемы со сборкой (в том числе, забытый org.jetbrains.annotations)	-2
Отсутствие JavaDoc-ов для всех классов, интерфейсов и паблик-методов	-2
Отсутствие описания метода в целом (в том числе, комментарии, начинающиеся с @return)	-1
Слишком широкие области видимости для полей	-2
if (...) return true; else return false;	-2
Комментарии для параметров с заглавной буквы	-0.5

Список может расширяться!

Многопоточное программирование вообще

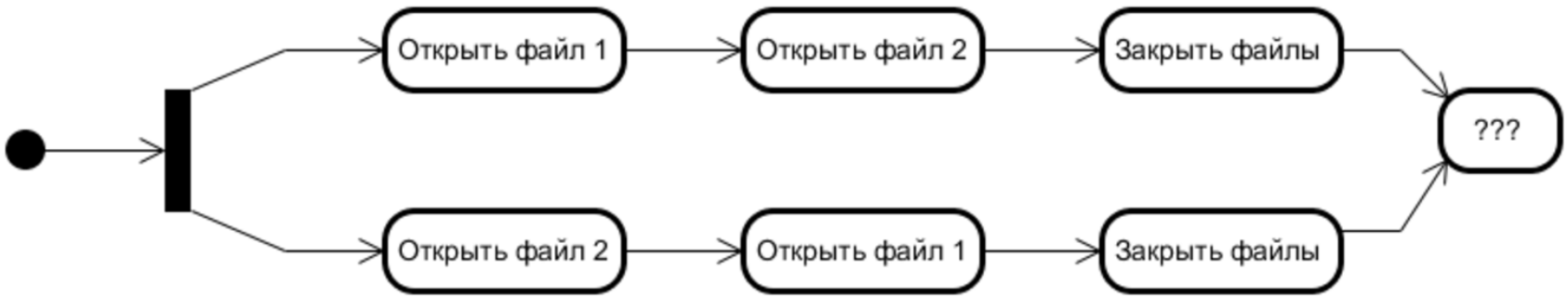
- Плюсы
  - Не вешать пользовательский интерфейс
  - Равномерно распределять вычислительно сложные задачи по ядрам
  - Выполнять одновременно несколько блокирующих операций ввода-вывода
- Минусы
  - Тысяча способов прострелить себе ногу
  - Не всегда многопоточная программа работает быстрее однопоточной

Race condition

Маленький пример на race condition

```
int[] a = new int[1000];
for (int i = 0; i < a.length; ++i) {
    a[i] = 1;
}

int[] result = new int[1];

for (int i = 0; i < 100; ++i) {
    final int localI = i;
    new Thread(() -> {
        for (int j = localI * 10; j <= (localI + 1) * 10 - 1; ++j) {
            result[0] += a[j];
        }
    }).start();
}

Thread.sleep(100);

System.out.println("Result = " + result[0]);
```

Deadlock


Открыть файл 1
Открыть файл 2
Закрыть файлы
???
Открыть файл 2
Открыть файл 1
Закрыть файлы

Очень маленький пример на deadlock

```
Thread.currentThread().join();
```

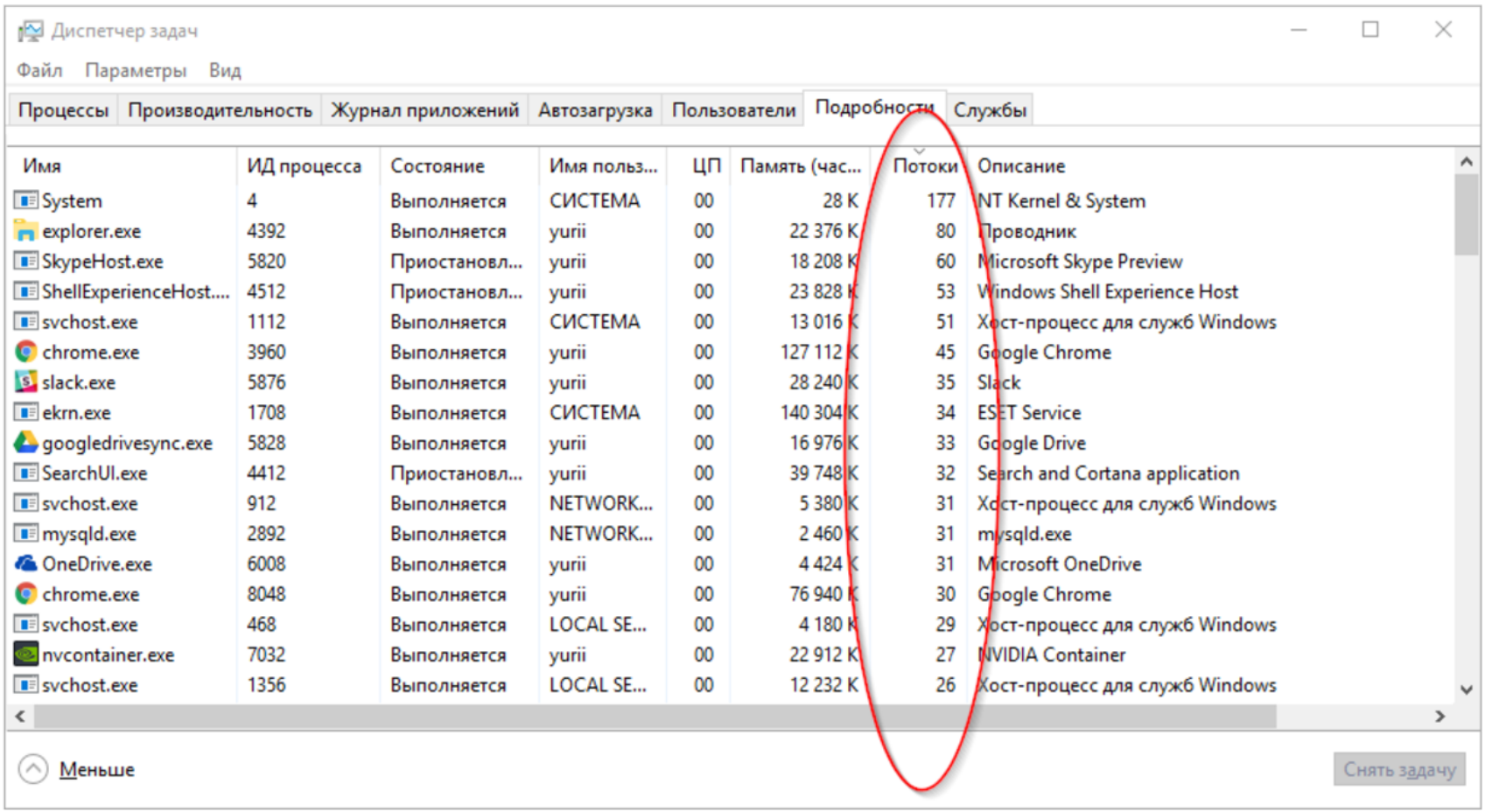
Пример, потоки в Windows

- Thread Kernel Object (~1240 байт)
- Thread environment block (TEB) (4 Кб)
- User-mode stack (1 Мб)
- Kernel-mode stack (24 Кб)

Ещё для каждой dll-ки, загруженной для процесса при старте или остановке потока вызывается DllMain с параметрами DLL_THREAD_ATTACH и DLL_THREAD_DETACH

Квант времени — 20-30 мс, после чего происходит *переключение контекстов*

Как делать не надо



Задача на пару, многопоточный Lazy

Реализовать следующий интерфейс, представляющий ленивое вычисление:

```
public interface Lazy<T> {
    T get();
}
```

- Объект *Lazy* создаётся на основе вычисления (представляемого объектом *Supplier*)
- Первый вызов *get()* вызывает вычисление и возвращает результат
- Повторные вызовы *get()* возвращают **тот же** объект, что и первый вызов
- Вычисление Supplier.get() должно запускаться не более одного раза для однопоточного случая и для “простого” многопоточного случая

LazyFactory

Создавать объекты надо не вручную, а с помощью класса *LazyFactory*, который должен иметь два метода с сигнатурами вида *public static <T> Lazy<T> create...Lazy(Supplier<T>)*, возвращающих три разные реализации *Lazy<T>*:

- Простая версия с гарантией корректной работы в однопоточном режиме (без синхронизации)
- Гарантия корректной работы в многопоточном режиме; вычисление не должно производиться более одного раза
  - Что-то наподобие многопоточного синглтона
- Для многопоточного lock-free режима
  - вычисление может производиться > 1 раза, но при этом Lazy.get всегда должен возвращать один и тот же объект (см. AtomicReference/AtomicReferenceFieldUpdater)

При этом

- Ограничение по памяти на каждый *Lazy*-объект: не больше двух полей
- *Supplier.get* вправе вернуть *null*
- Тесты
  - Однопоточные, на разные хорошие и плохие случаи
  - Многопоточные, на наличие гонок
- Делать в командах по два человека

Баллы

- 1 балл за однопоточную реализацию
- 2 балла за многопоточную реализацию с блокировками
- 2 балла за многопоточную реализацию с lock-free
- 1 балл за многопоточные тесты на гонки (включая требование, что вычисление производится один раз)
- -1 балл, если потоки неоправданно мешают друг другу
- -1 балл за потенциальные проблемы с порядком операций