Lazy

```
public interface Lazy<T> {

    T get();
}
```

- При первом вызове метода *Lazy.get*, результат получается из вызова *Supplier.get()*
- Повторные вызовы *Lazy.get* должны возвращать тот же объект, который вернулся в п.1
- В однопоточном режиме вычисление должно запускаться не более одного раза
- Если работают несколько потоков, возможность повторного запуска вычисления определяется реализацией

LazyFactory

Необходимо создать класс *LazyFactory*, возвращающий в трех разных методах разные реализации *Lazy*.

- Для однопоточного режима
- Для многопоточного режима; вычисление не должно производиться > 1 раза (см. Singleton)
- 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)

Рекомендуемый шаблон сигнатуры методов: *public static <T> Lazy<T> createLazy(Supplier<T>)*

Требования:

- Maven/Gradle проект
- Ограничение по памяти на каждый *Lazy*-объект: не больше двух ссылок
- Тесты
  - однопоточные
  - многопоточные

Примечания:

- Помните, что *Supplier.get* может вернуть *null*.

Срок: **15.02.2016 23:59**

Формат сдачи

- Каждое задание выполняете в отдельной ветке в репозитории на GitHub
- Создаете pull request ветки в master этого же репозитория
- Тема PR: Java02. ДЗ 0_, <фамилия> <имя>, подгруппа _
- В комментарии упоминаете username преподавателя
- Посылаете письмо преподавателю с такой же темой с ссылкой на pull request

Штрафы

- Несоблюдение формата сдачи
- Неотформатированный код
- Повторение предыдущих ошибок
- Количество попыток