

Java 25. Горячие JEP'ы

Бесконечный цикл докладов на Joker/JPoint

Java 25. Горячие JEP'ы



юбилейное

На манеже всё те же



Андрей Кулешов



Григорий Кошелев



Вадим Цесько

На манеже всё те же



Андрей Кулешов



Григорий Кошелев

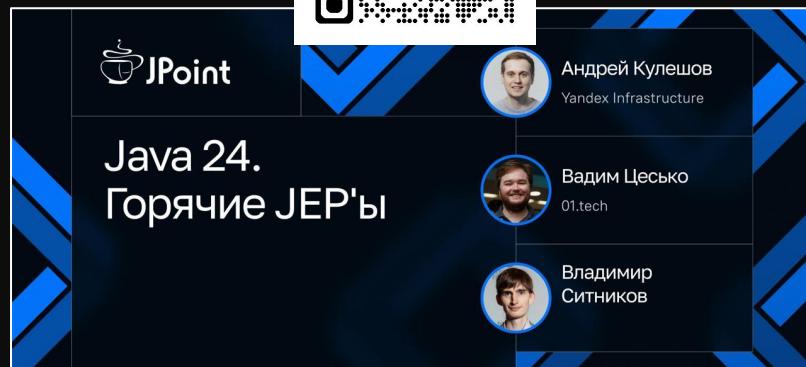
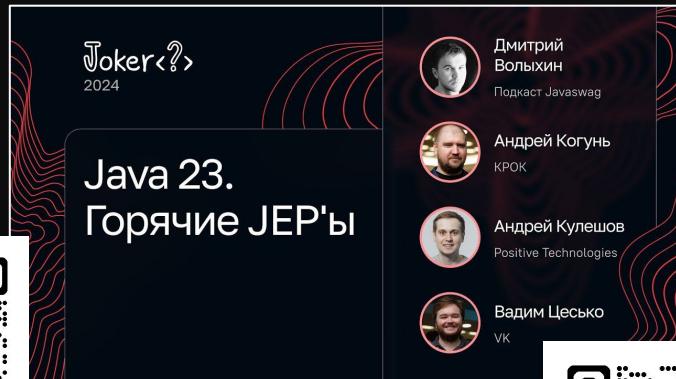
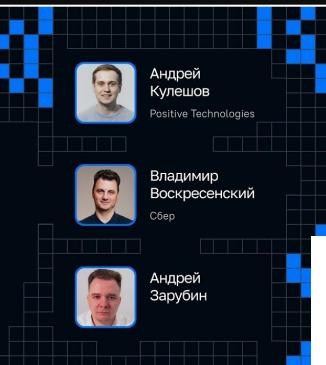
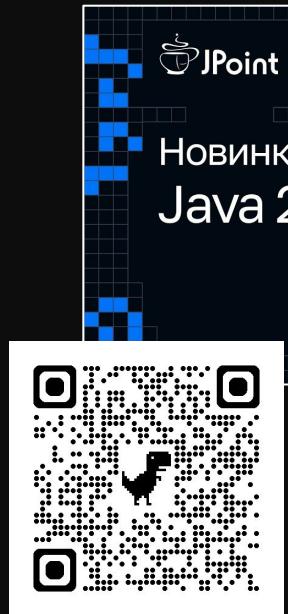


Вадим Цесько

+ благодарности

Андрею Когуню, Владимиру Ситникову и Диме Волыхину

В предыдущих сериях



О ЧЁМ ГОВОРИМ

Oracle Java SE Support Roadmap^{*†}

Release	GA Date	Premier Support Until	Extended Support Until
8 (LTS)**	March 2014	March 2022	December 2030****
9 - 10 (non-LTS)	September 2017 - March 2018	March 2018 - September 2018	Not Available
11 (LTS)	September 2018	September 2023	January 2032****
12 - 16 (non-LTS)	March 2019 - March 2021	September 2019 - September 2021	Not Available
17 (LTS)	September 2021	September 2026****	September 2029****
18 - 20 (non-LTS)	March 2022 - March 2023	September 2022 - September 2023	Not Available
21 (LTS)	September 2023	September 2028****	September 2031****
22 (non-LTS)	March 2024	September 2024	Not Available
23 (non-LTS)	September 2024	March 2025	Not Available
24 (non-LTS)	March 2025	September 2025	Not Available
25 (LTS)	September 2025	September 2030****	September 2033****
26 (non-LTS)***	March 2026	September 2026	Not Available
27 (non-LTS)***	September 2026	March 2027	Not Available

О чём говорим

Oracle Java SE Support Roadmap*†

Release	GA Date	Premier Support Until	Extended Support Until
8 (LTS)**	March 2014	March 2022	December 2030****
9 - 10 (non-LTS)	September 2017 - March 2018	March 2018 - September 2018	Not Available
11 (LTS)	September 2018	September 2023	January 2032****
12 - 16 (non-LTS)	March 2019 - March 2021	September 2019 - September 2021	Not Available
17 (LTS)	September 2021	September 2026****	September 2029****
18 - 20 (non-LTS)	March 2022 - March 2024	September 2022 - September 2023	Not Available
21 (LTS)	September 2023	September 2028****	September 2031****
22 (non-LTS)	March 2024	September 2024	Not Available
23 (non-LTS)	September 2024	March 2025	Not Available
24 (non-LTS)	March 2025	September 2025	Not Available
25 (LTS)	September 2025	September 2030****	September 2033****
26 (non-LTS)***	March 2026	September 2026	Not Available
27 (non-LTS)***	September 2026	March 2027	Not Available



JDK Project

Installing
Contributing
Sponsoring
Developers' Guide
Vulnerabilities
JDK GA/EA Builds

Mailing lists
Wiki · IRC
Mastodon
Bluesky
Bylaws · Census
Legal

Workshop

JEP Process
Source code
GitHub
Mercurial

Tools
Git
jtreg harness

Groups
(overview)
Adoption
Build
Client Libraries
Compatibility &

The goal of this long-running Project is to produce a series of open-source reference implementations of the Java SE Platform, as specified by JSRs in the Java Community Process. The Project ships a feature release every six months according to a strict, time-based model, [as proposed](#).

Releases

- 26 (in development)
- 25 (GA 2025/09/16)
- 24 (GA 2025/03/18)
- 23 (GA 2024/09/17)
- 22 (GA 2024/03/19)
- 21 (GA 2023/09/19)
- 20 (GA 2023/03/21)
- 19 (GA 2022/09/20)
- 18 (GA 2022/03/22)
- 17 (GA 2021/09/14)
- 16 (GA 2021/03/16)
- 15 (GA 2020/09/15)
- 14 (GA 2020/03/17)
- 13 (GA 2019/09/17)
- 12 (GA 2019/03/19)
- 11 (GA 2018/09/25)
- 10 (GA 2018/03/20)

Resources

- Development list: [jdk-dev](#)
- Repository: <https://github.com/openjdk/jdk/>
- [Group, Area, & Project Leads](#)

JDK Project

Installing
Contributing
Sponsoring
Developers' Guide
Vulnerabilities
JDK GA/EA Builds

Mailing lists
Wiki · IRC
Mastodon
Bluesky

Bylaws · Census
Legal

Workshop

JEP Process

Source code
GitHub
Mercurial

Tools

Git
jtreg harness

Groups
(overview)
Adoption
Build
Client Libraries
Compatibility &

The goal of this long-running Project is to produce a series of open-source reference implementations of the Java SE Platform, as specified by JSRs in the Java Community Process. The Project ships a feature release every six months according to a strict, time-based model, [as proposed](#).

Releases

- 26 (in development)
- 25 (GA 2025/09/16)
- 24 (GA 2025/03/18)
- 23 (GA 2024/09/17)
- 22 (GA 2024/03/19)
- 21 (GA 2023/09/19)
- 20 (GA 2023/03/21)
- 19 (GA 2022/09/20)
- 18 (GA 2022/03/22)
- 17 (GA 2021/09/14)
- 16 (GA 2021/03/16)
- 15 (GA 2020/09/15)
- 14 (GA 2020/03/17)
- 13 (GA 2019/09/17)
- 12 (GA 2019/03/19)
- 11 (GA 2018/09/25)
- 10 (GA 2018/03/20)

Resources

- Development list: [jdk-dev](#)
- Repository: <https://github.com/openjdk/jdk/>
- [Group, Area, & Project Leads](#)

КТО УЖЕ
ПЕРЕШЁЛ?

Что внутри?

12 “стабильных”

2678 коммитов

1 экспериментальный

2 превью ушли на
следующий круг

2 новых превью

1 инкубатор

JDK 21 vs JDK 25

Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#) or [learn more about diff comparisons](#).



base: jdk-21-ga ▾



compare: jdk-25-ga ▾

-o Commits 5,000+

Files changed 0

101 contributors



34 JEP'a

Краткий повтор:
НЕ включая Java 25

Краткий повтор: Hotspot

Улучшения всех GC – G1 / ZGC / Shenandoah: Region Pinning, Generational Modes

Краткий повтор: Hotspot

Улучшения всех GC – G1 / ZGC / Shenandoah: Region Pinning, Generational Modes

AOT Загрузка и линковка классов

Краткий повтор: Hotspot

Улучшения всех GC – G1 / ZGC / Shenandoah: Region Pinning, Generational Modes

AOT Загрузка и линковка классов

Избавление от пиннинга виртуальных потоков

Краткий повтор: сам язык

Unnamed переменные:

- for (Order _ : orders)
- var _ = q.remove();
- catch (NumberFormatException _) {
- try (var _ = ScopedContext.acquire()) {
- case RedBall _ → process(ball);

Краткий повтор: библиотеки

Class-File API для работы с байткодом

Краткий повтор: библиотеки

Class-File API для работы с байткодом

FFM API для работы с нативным кодом и памятью

Краткий повтор: библиотеки

Class-File API для работы с байткодом

FFM API для работы с нативным кодом и памятью

Stream.of(1,2,3,4,5).gather(/* MyGatherer */).toList()

Краткий повтор: библиотеки

Class-File API для работы с байткодом

FFM API для работы с нативным кодом и памятью

Stream.of(1,2,3,4,5).gather(/* MyGatherer */).toList()

Криптография: от квантово-устойчивых алгоритмов до API для Key Derivation Functions (KDFs)

Краткий повтор: инструменты

JavaDoc'и теперь можно писать с markdownом внутри:

```
/// - [the `java.base` module](java.base/)
```

Краткий повтор: инструменты

JavaDoc'и теперь можно писать с markdownом внутри:

```
/// - [the `java.base` module](java.base/)
```

Запуск программ из нескольких файлов: \$ java MyProg.java

Краткий повтор: инструменты

JavaDoc'и теперь можно писать с markdownом внутри:

```
/// - [the `java.base` module](java.base/)
```

Запуск программ из нескольких файлов: `$ java MyProg.java`

JDK становится компактнее на 25% за счет выиливания JMOD-файлов

**Краткий повтор:
идём в сторону
отказа от
32-битных
портов JDK**



```
$ diff Java24 Java25
```

```
--- Java 24
```

```
+++ Java 25
```

- Всё ждем Project Valhalla
- + Зато ждём быстрее
- Новые JEPы, которые вы не попробуете
- + 1 Vector API еще раз Incubator

Дисклеймер: некоторые ЕРы **без** изменений



01

Сразу стабильные
новые JEРы

03

Следующее
Preview

02

Из Preview в
стабильные JEРы

04

Experimental
Incubator

01

Сразу стабильные
новые JEPы

03

Следующее
Preview

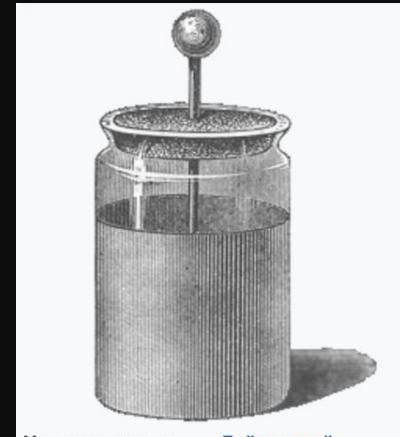
02

Из Preview в
стабильные JEPы

04

Experimental
Incubator

Что это?



Лейденская банка

Материал из Википедии — свободной энциклопедии

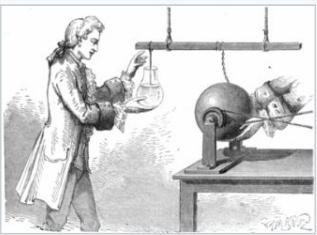
[править код]

Лейденская банка — первый электрический конденсатор, изобретённый голландским учёным Питером ван Мушенбруком и его учеником Кюнеусом в 1745 году в Лейдене. Параллельно и независимо от них сходный аппарат под названием «медицинская банка» изобрёл немецкий учёный Эвальд Юрген фон Клейст.

Содержание [скрыть]

- 1 Устройство
- 2 Значение
- 3 Наши дни
- 4 Галерея
- 5 Ссылки

Устройство [править | править код]



Обнаружение лейденской банки в лаборатории Мушенброка. Статическое электричество, создаваемое электростатическим генератором вращающейся стеклянной сферы, проводилось цепью через подвесной стержень к воде в стакане,

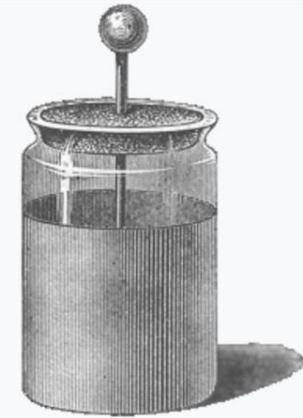
Согласно энциклопедии Ф. А. Брокгауза и И. А. Ефрана, «этот конденсатор имеет форму банки, то есть цилиндра с более или менее широким горлом или же просто цилиндра, обыкновенно стеклянного. Банка оклеена внутри и снаружи листовым оловом (наружная и внутренняя обкладки) примерно до 2/3 её высоты и прикрыта деревянной крышкой. Банка может не иметь внутренней обкладки, но тогда в ней должна быть жидкость, например вода; банка может не иметь и внешней обкладки, но в таком случае при заряджении надо её обхватить ладонями; такова и была банка в первоначальном виде, когда её устроил (1745) голландский физик Мушенбрук и когда впервые испытал удар от разряда банки лейденский гражданин Кюнеус».

Лейденская банка позволяла накапливать и хранить сравнительно большие заряды, порядка микрокулона.

Значение [править | править код]

Изобретение лейденской банки стимулировало изучение электричества, в частности,

Лейденская банка



Названо в честь Лейденский университет
Первооткрыватель Эвальд Юрген фон Клейст
Дата открытия (изобретения) 1745

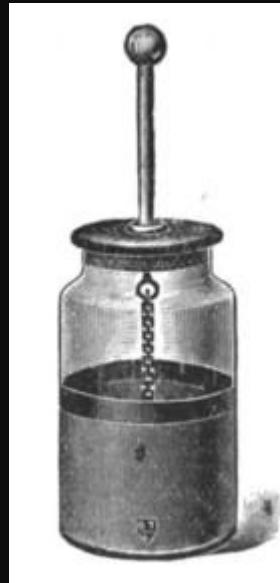
Медиафайлы на Викискладе



01

JEP 514: Ahead-of-Time Command-Line Ergonomics

Новые стабильные JEPы



В предыдущих сериях, [483](#): Ahead-of-Time Class Loading & Linking

- Запускаем ~~түся, работяги~~, собираем статистику и конфигурацию АОТ:

```
java -XX:AOTMode=record -XX:AOTConfiguration=app.aotconf -cp app.jar com.example.App
```

- Создаём АОТ кэш с информацией о линковке:

```
java -XX:AOTMode=create -XX:AOTConfiguration=app.aotconf -XX:AOTCache=app.aot -cp app.jar
```

В предыдущих сериях, [483](#): Ahead-of-Time Class Loading & Linking

- Запускаем ~~түсініктер~~, собираем статистику и конфигурацию АОТ:

```
java -XX:AOTMode=record -XX:AOTConfiguration=app.aotconf -cp app.jar com.example.App
```

- Создаём АОТ кэш с информацией о линковке:

```
java -XX:AOTMode=create -XX:AOTConfiguration=app.aotconf -XX:AOTCache=app.aot -cp app.jar
```

- Запускаем приложение с вышесозданным кэшом:

```
java -XX:AOTCache=app.aot -cp app.jar com.example.App
```

- *"If cache is unavailable or incompatible, JVM issues a warning and continues."*

514: Ahead-of-Time Command-Line Ergonomics

- Запускаем ~~туся, работяги~~, собираем статистику и конфигурацию АОТ:

```
java -XX:AOTMode=record -XX:AOTConfiguration=app.aotconf -cp app.jar com.example.App
```

- Создаём АОТ кэш с информацией о линковке:

```
java -XX:AOTMode=create -XX:AOTConfiguration=app.aotconf -XX:AOTCache=app.aot -cp app.jar
```

- Запускаем приложение с вышесозданным кэшом:

```
java -XX:AOTCache=app.aot -cp app.jar com.example.App
```

- “If cache is unavailable or incompatible, JVM issues a warning and continues.”

514: Ahead-of-Time Command-Line Ergonomics

```
java -XX:AOTCacheOutput=app.aot -cp app.jar com.example.App  
или JDK_AOT_VM_OPTIONS
```

- Запускаем приложение с вышесозданным кэшом:

```
java -XX:AOTCache=app.aot -cp app.jar com.example.App
```

- “If cache is unavailable or incompatible, JVM issues a warning and continues.”

514: Ahead-of-Time Command-Line Ergonomics

А ведь думали про **-XX:AOTMode=record+create**

- Запускаем приложение с вышесозданным кэшом:

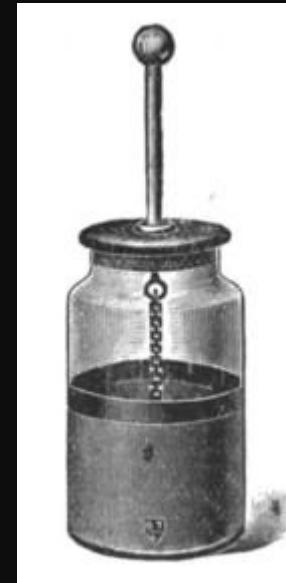
```
java -XX:AOTCache=app.aot -cp app.jar com.example.App
```

- “If cache is unavailable or incompatible, JVM issues a warning and continues.”

01

JEP 515: Ahead-of-Time Method Profiling

Новые стабильные JEPы



515: Ahead-of-Time Method Profiling

- Запускаем ~~түся, работяги~~, собираем статистику и строим AOT кеш:

```
java -XX:AOTCacheOutput=app.aot -cp app.jar com.example.App
```

- Запускаем приложение с вышесозданным кэшом:

```
java -XX:AOTCache=app.aot -cp app.jar com.example.App
```

- Развиваем достигнутый успех!

Теперь ещё и данные профилирования методов -- можно сразу запускать JIT

515: Ahead-of-Time Method Profiling

```
public class HelloStreamWarmup {  
  
    static String greeting(int n) {  
        var words = List.of("Hello", "" + n, "world!");  
        return words.stream()  
            .filter(w → !w.contains("0"))  
            .collect(Collectors.joining(", "));  
    }  
  
    public static void main(String... args) {  
        for (int i = 0; i < 100_000; i++)  
            greeting(i);  
        // "Hello, world!"  
        System.out.println(greeting(0));  
    }  
}
```

515: Ahead-of-Time Method Profiling

```
public class HelloStreamWarmup {  
  
    static String greeting(int n) {  
        var words = List.of("Hello", "" + n, "world!");  
        return words.stream()  
            .filter(w → !w.contains("0"))  
            .collect(Collectors.joining(", "));  
    }  
  
    public static void main(String... args) {  
        for (int i = 0; i < 100_000; i++)  
            greeting(i);  
        // "Hello, world!"  
        System.out.println(greeting(0));  
    }  
}
```

- 900 классов
- 30 горячих методов
- 90 мс → 73 мс

515: Ahead-of-Time Method Profiling

```
public class HelloStreamWarmup {  
  
    static String greeting(int n) {  
        var words = List.of("Hello", "" + n, "world!");  
        return words.stream()  
            .filter(w → !w.contains("0"))  
            .collect(Collectors.joining(", "));  
    }  
  
    public static void main(String... args) {  
        for (int i = 0; i < 100_000; i++)  
            greeting(i);  
        // "Hello, world!"  
        System.out.println(greeting(0));  
    }  
}
```

- 900 классов
- 30 горячих методов
- 90 мс → 73 мс

515: Ahead-of-Time Method Profiling

```
public class HelloStreamWarmup {  
  
    static String greeting(int n) {  
        var words = List.of("Hello", "" + n, "world!");  
        return words.stream()  
            .filter(w → !w.contains("0"))  
            .collect(Collectors.joining(", "));  
    }  
  
    public static void main(String... args) {  
        for (int i = 0; i < 100_000; i++)  
            greeting(i);  
        // "Hello, world!"  
        System.out.println(greeting(0));  
    }  
}
```

- 900 классов
- 30 горячих методов
- 90 мс → 73 мс

515: Ahead-of-Time Method Profiling

```
public class HelloStreamWarmup {  
  
    static String greeting(int n) {  
        var words = List.of("Hello", "" + n, "world!");  
        return words.stream()  
            .filter(w → !w.contains("0"))  
            .collect(Collectors.joining(", "));  
    }  
  
    public static void main(String... args) {  
        for (int i = 0; i < 100_000; i++)  
            greeting(i);  
        // "Hello, world!"  
        System.out.println(greeting(0));  
    }  
}
```

- 900 классов
- 30 горячих методов
- 90 мс → 73 мс

01

JEP 518: JFR Cooperative Sampling

Новые стабильные JЕРы

518: JFR Cooperative Sampling

Проблема

- Метаданные для stack frames валидны только в safepoints
- Но не хотим **safepoint bias**
- Поэтому **эвристики** и **возможные падения**

518: JFR Cooperative Sampling

Проблема

- Метаданные для stack frames валидны только в safepoints
- Но не хотим **safepoint bias**
- Поэтому **эвристики** и **возможные падения**

Решение

- Запоминаем PC и stack pointer вне safepoint
 - Работает **быстрее**
- Кладём sample request в thread local очередь
- В safepoint разгребаем очередь, реконструируем стеки и **эмитим JFR иVENTы**
 - Код **проще**

01

JEP 520: JFR Method Timing & Tracing

Новые стабильные JEPы

520: JFR Method Timing & Tracing

```
$ java -XX:StartFlightRecording:  
      jdk.MethodTrace#filter=java.util.HashMap::resize,filename=resize.jfr
```

520: JFR Method Timing & Tracing

```
$ java -XX:StartFlightRecording:  
      jdk.MethodTrace#filter=java.util.HashMap::resize,filename=resize.jfr  
  
$ jfr print --events jdk.MethodTrace --stack-depth 20 resize.jfr
```

520: JFR Method Timing & Tracing

```
$ java -XX:StartFlightRecording:  
      jdk.MethodTrace#filter=java.util.HashMap::resize,filename=resize.jfr  
  
$ jfr print --events jdk.MethodTrace --stack-depth 20 resize.jfr  
  
$ java -XX:StartFlightRecording:method-timing=:<clinit> ...  
$ jfr view method-timing clinit.jfr
```

520: JFR Method Timing & Tracing

```
$ java -XX:StartFlightRecording:  
      jdk.MethodTrace#filter=java.util.HashMap::resize,filename=resize.jfr  
  
$ jfr print --events jdk.MethodTrace --stack-depth 20 resize.jfr  
  
$ java -XX:StartFlightRecording:method-timing=:<clinit> ...  
$ jfr view method-timing clinit.jfr  
  
$ jccmd <pid> JFR.start method-timing=@jakarta.ws.rs.GET
```

520: JFR Method Timing & Tracing

```
// Можно и через RemoteRecordingStream поверх JMX  
  
settings.put("jdk.MethodTrace#enabled", "true");  
settings.put("jdk.MethodTrace#stackTrace", "true");  
settings.put("jdk.MethodTrace#threshold", "1 ms");  
settings.put("jdk.MethodTrace#filter", "com.foo.Bar");
```

520: JFR Method Timing & Tracing

- Не нужны никакие агенты
- JFR обычно обещает оверхед <1%, но **не в этом случае**
- Timing оценивает min/avg/max
- Фильтры по имени класса и/или метода

520: JFR Method Timing & Tracing

- Не нужны никакие агенты
- JFR обычно обещает оверхед <1%, но **не в этом случае**
- Timing оценивает min/avg/max
- Фильтры по имени класса и/или метода
 - Нет интерфейсов
 - Нет сигнатур
 - Нет wildcards

520: JFR Method Timing & Tracing

- Не нужны никакие агенты
- JFR обычно обещает оверхед <1%, но **не в этом случае**
- Timing оценивает min/avg/max
- Фильтры по имени класса и/или метода
 - Нет интерфейсов
 - Нет сигнатур
 - Нет wildcards
 - Но возможна **бесконечная рекурсия**

ДОКЛАД

JDK

Performance

17.10 / 16:30 – 17:15 (UTC+3)

JDK Flight Recorder в 2025-ом

Зал 2

RU



JDK Flight Recorder – не новая технология, претерпевшая несколько перерождений за свою историю. Очередной релиз JDK 25 также не обошел стороной JDK Flight Recorder и принес с собой новые возможности.

Развитие JDK Flight Recorder идет динамично, но часто остается ниже радара большинства разработчиков.

В докладе подведу итог тем возможностям, которые JDK Flight Recorder накопил к 2025 году, включая возможности API, работу из командной строки и широкий набор событий JVM. Ну и, конечно, разберем новые возможности, которые принес нам JDK 25 – такие как возможность трассировки методов, создания контекстных событий и новые механизмы сэмплирования.

Спикеры



Алексей Рагозин

Приглашенные эксперты



Иван Углынский

Huawei

01

Сразу стабильные
новые JEPы

02

Из Preview в
стабильные JEPы

03

Следующее
Preview

04

Experimental
Incubator

02

JEP 503: Remove the 32-bit x86 Port

Из Preview в стабильные JEPы



503: Remove the 32-bit x86 Port

- Поддержку Windows 32-bit x86 уже дропнули в JDK 24 (JEP 479)
- Windows 10 End of Life на этой неделе
- Debian планирует дропнуть 32-bit x86 вот-вот

503: Remove the 32-bit x86 Port

- Поддержку Windows 32-bit x86 уже дропнули в JDK 24 (JEP 479)
- Windows 10 End of Life на этой неделе
- Debian планирует дропнуть 32-bit x86 вот-вот
- ***Unblock new features that require platform-specific support from having to implement 32-bit x86 fallbacks***

503: Remove the 32-bit x86 Port

- Поддержку Windows 32-bit x86 уже дропнули в JDK 24 (JEP 479)
- Windows 10 End of Life на этой неделе
- Debian планирует дропнуть 32-bit x86 вот-вот
- *Unblock new features that require platform-specific support from having to implement 32-bit x86 fallbacks*
- R.I.P.

02

JEP 506: Scoped Values

Из Preview в стабильные JEPы

506: Scoped Values

```
private static final ScopedValue<FrameworkContext> CONTEXT =
    ScopedValue.newInstance();

void serve(Request request, Response response) {
    var context = createContext(request);
    ScopedValue.where(CONTEXT, context).run(
        () -> Application.handle(request, response));
}

public PersistedObject readKey(String key) {
    var context = CONTEXT.get();
    var db = getDBConnection(context);
    db.readKey(key);
}
```

506: Scoped Values

- One-way transmission of unchanging data
- Lower space and time costs
- Автоматическое наследование в StructuredTaskScope

506: Scoped Values

- One-way transmission of unchanging data
- Lower space and time costs
- Автоматическое наследование в StructuredTaskScope

ПОЧТИ БЕЗ ИЗМЕНЕНИЙ

- Но только ScopedValue.orElse() больше не принимает null

506: Scoped Values

Доклад



ThreadLocal устарел? Детальное сравнение со ScopedValue

С появлением виртуальных потоков ThreadLocal перестал быть единственным решением для передачи контекста – в Java 21 появился ScopedValue. Но когда что использовать? Что быстрее и занимает меньше памяти?



Александр Маторин

Сбер

RU



JDK

Languages

Performance

02

JEP 519: Compact Object Headers

Из Preview в стабильные JEPы

519: Compact Object Headers

```
$ java -XX:+UseCompactObjectHeaders ...
```

- 96-128 бит → 64 бит
 - Из них 4 бита ушли в Valhalla
- Не больше 4M классов
 - Compressed class pointers подрезали 32 бит → 22 бит
- Много бенчмарков с заметным профитом -- пробуйте!

519: Compact Object Headers



519: Compact Object Headers

Доклад



32 GB хватит всем

Расскажу, как некоторые старые (compressed oops и compressed class pointers) и новые (compact object headers) настройки влияют на реальное использование памяти, и что делать, если ваше приложение приближается к границе в 32 GB.



Олег Естехин
Yandex Cloud

RU



JVM

Performance

02

JEP 521: Generational Shenandoah

Из Preview в стабильные JEPы

521: Generational Shenandoah

```
$ java -XX:+UseShenandoahGC -XX:ShenandoahGCMode=generational ...
```

521: Generational Shenandoah

```
$ java -XX:+UseShenandoahGC -XX:ShenandoahGCMode=generational ...
```

Вспомним конкурентов из JEP 404:

- C4 от Azul **не** в open-source и **не** поддерживает compressed oops
- Generational ZGC из 21 **не** поддерживает compressed oops
- Тем временем, большинство heap'ов << 32 ГБ

Профит на куче бенчмарков -- пробуйте!

521: Generational Shenandoah



02

JEP 511: Module Import Declarations

Из Preview в стабильные JEPы

511: Module Import Declarations

```
import module java.xml;  
import module java.desktop;  
import module java.se; // JShell
```

511: Module Import Declarations

```
import module java.xml;  
import module java.desktop;  
import module java.se; // JShell
```

- Транзитивное тоже импортируется
 - `java.se` → `java.base`
- Готовимся к конфликтам при использовании неоднозначностей
 - "*The shadowing behavior of import declarations matches their specificity*"

511: Module Import Declarations

```
import module java.xml;  
import module java.desktop;  
import module java.se; // JShell
```



- Транзитивное тоже импортируется
 - `java.se` → `java.base`
- Готовимся к конфликтам при использовании неоднозначностей
 - "*The shadowing behavior of import declarations matches their specificity*"
- Сделаем Java ~~great again~~ первым языком программирования?

511: Module Import Declarations



02

JEP 512: Compact Source Files and Instance Main Methods

Из Preview в стабильные JEPы

Как писали деды

```
import java.util.List;

public class Speaker {

    public static void main() {
        var speakers = List.of("Andrew/Alexey", "Gregory", "Vadim");
        for (var name : speakers) {
            System.out.println(name + " is talking");
        }
    }
}
```



512: Compact Source Files and Instance Main Methods

```
void main() {  
    var speakers = List.of("Andrew/Alexey", "Gregory", "Vadim");  
    for (var name : speakers) {  
        IO.println(name + " is talking");  
    }  
}
```

512: Compact Source Files and Instance Main Methods (**diff**)

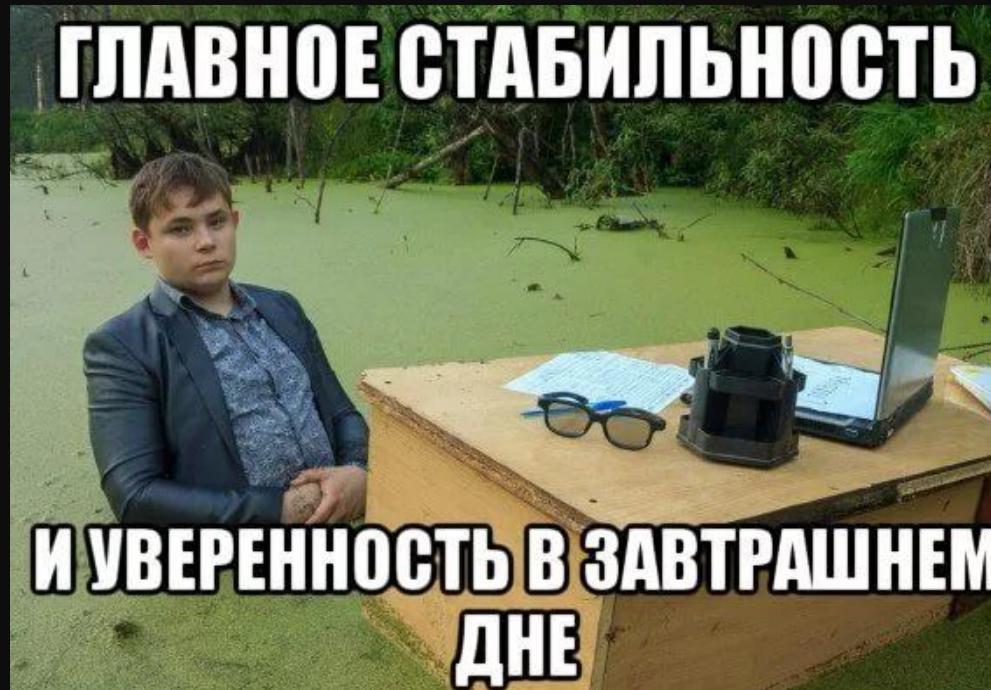
```
import module java.base; // Implicitly
//import static java.lang.Io.*;

void main() {
    var speakers = List.of("Andrew/Alexey", "Gregory", "Vadim");
    for (var name : speakers) {
        Io.println(name + " is talking");
    }
}
```

JEP 477: Implicitly Declared Classes and Instance Main Methods

JEP 495: Simple Source Files and Instance Main Methods

JEP 512: Compact Source Files and Instance Main Methods



02

JEP 513: Flexible Constructor Bodies

Из Preview в стабильные JEPы

02

JEP 513: Flexible Constructor Bodies

Гибкие конструкторские тела

Из Preview в стабильные JEPы



Как писали деды

```
class Pensioner extends Person {  
  
    Pensioner(boolean sex, int age) {  
        super(sex, age); // Potentially unnecessary work  
        if (sex && age < 60 || !sex && age < 65)  
            throw new IllegalArgumentException("Wrong age");  
    }  
  
}
```



Как писали деды-оптимизаторы



```
class Pensioner extends Person {  
  
    private static int verifyAge(boolean sex, int age) {  
        if (sex && age < 60 || !sex && age < 65)  
            throw new IllegalArgumentException("Wrong age");  
        return age;  
    }  
  
    Pensioner(boolean sex, int age) {  
        super(sex, verifyAge(sex, age));  
    }  
}
```

513: Flexible Constructor Bodies

```
class Pensioner extends Person {  
  
    Pensioner(boolean sex, int age) {  
        if (sex && age < 60 || !sex && age < 65)  
            // Now fails fast!  
        throw new IllegalArgumentException("Wrong age");  
  
        super(sex, age);  
    }  
  
}
```

513: Flexible Constructor Bodies

До вызова `super()/this()` нельзя:

- Явно или неявно использовать `this` и `super`
 - Даже для чтения полей и вызовов методов объекта

513: Flexible Constructor Bodies

До вызова `super()/this()` нельзя:

- Явно или неявно использовать `this` и `super`
 - Даже для чтения полей и вызовов методов объекта

Но можно:

- Присваивать значения неинициализированным полям

513: Flexible Constructor Bodies

До вызова `super()/this()` нельзя:

- Явно или неявно использовать `this` и `super`
 - Даже для чтения полей и вызовов методов объекта

Но можно:

- Присваивать значения неинициализированным полям

В байткоде так можно было уже много лет.

513: Flexible Constructor Bodies



02

JEP 510: Key Derivation Function API

Из Preview в стабильные JEPы

510: Key Derivation Function API

- Идём к Post-Quantum Cryptography
- Через Hybrid Public Key Encryption (HPKE)
 - KEM API из JEP 452 был первым шагом
 - KDF API -- следующий шаг ← мы здесь
 - ...
- См. DSL в `javax.crypto.KDF`
- Позже планируют Argon2 KDF

510: Key Derivation Function API

- Идём к Post-Quantum Cryptography
- Через Hybrid Public Key Encryption (HPKE)
 - KEM API из JEP 452 был первым шагом
 - KDF API -- следующий шаг ← мы здесь
 - ...
- См. DSL в `javax.crypto.KDF`
- Позже планируют Argon2 KDF
- И заодно отрефакторить TLS 1.3

510: Key Derivation Function API



510: Key Derivation Function API

Доклад



Квантовые технологии: на пути к практическим задачам

Сегодня квантовые технологии – одно из наиболее бурно развивающихся направлений. Уже сегодня квантовые компьютеры начинают решать практические задачи в области оптимизации и машинного обучения, квантовые коммуникации внедряются в телекоммуникационные системы, а квантовые сенсоры используются для задач, например, в области биомедицины. Понимание возможностей квантовых технологий – это шаг в понимании будущего мира.



Алексей Федоров
Российский квантовый центр / МИСИС

01

Сразу стабильные
новые JEРы

02

Из Preview в
стабильные JEРы

03

Следующее
Preview

04

Experimental
Incubator

03

JEP 502: Stable Values (будущие Lazy Constants)

Новинка Preview

502: Stable Values (будущие Lazy Constants)

Тезис: отложенная, но
гарантированная
неизменяемость объектов

502: Stable Values (будущие Lazy Constants)

```
class OrderController {  
  
    private final StableValue<Logger> logger = StableValue.of();  
  
    Logger getLogger() {  
        return logger.orElseSet(() -> Logger.create(OrderController.class));  
    }  
}
```

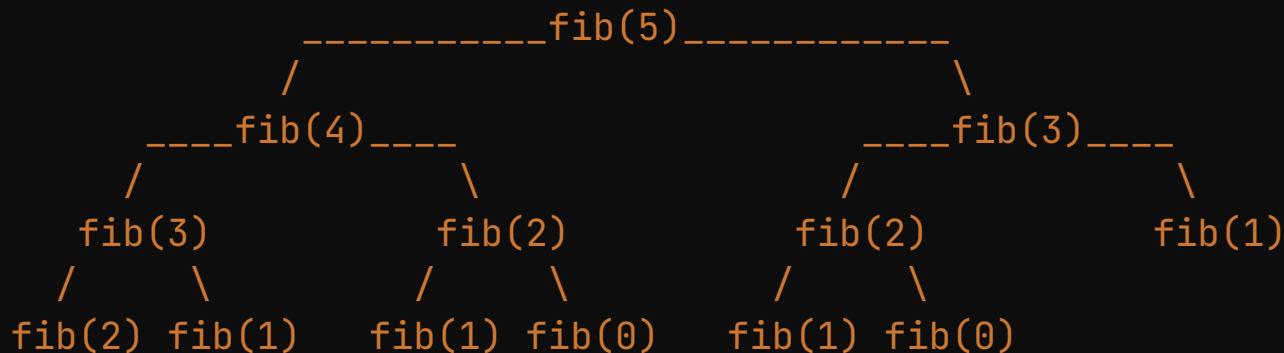
502: Stable Values

```
class Dependents {  
  
    static class Foo {}  
  
    static class Bar {  
        Bar(Foo foo) {}  
    }  
  
    private static final Supplier<Foo> FOO =  
        StableValue.supplier(Foo::new);  
  
    private static final Supplier<Bar> BAR =  
        StableValue.supplier(() -> new Bar(FOO.get()));  
}
```

502: Stable Values

```
class Fibonacci {  
  
    private static final int MAX_SIZE_INT = 46;  
  
    private static final IntFunction<Integer> FIB =  
        StableValue.intFunction(MAX_SIZE_INT, Fibonacci::fib);  
  
    public static int fib(int n) {  
        return n < 2  
            ? n  
            : FIB.apply(n - 1) + FIB.apply(n - 2);  
    }  
}
```

502: Stable Values



If there are circular dependencies in a dependency graph, a stable value will eventually throw an `IllegalStateException` upon referencing elements in a circularity.

502: Stable Values

- Отложная иммутабельность
- Stable Value, Supplier, Function, IntFunction, List, Map
- Thread safety
- Constant-folding optimizations
 - Elide all future reads

502: Stable Values

- Отложная иммутабельность
- Stable Value, Supplier, Function, IntFunction, List, Map
- Thread safety
- Constant-folding optimizations
 - Elide all future reads
 - Но только если static final StableValue<T>

502: Stable Values

- Отложная иммутабельность
- Stable Value, Supplier, Function, IntFunction, List, Map
- Thread safety
- Constant-folding optimizations
 - Elide all future reads
 - Но только если static final StableValue<T>
 - Где бенчмарки, Лебовски?!

502: Stable Values

- Отложная иммутабельность
- Stable Value, Supplier, Function, IntFunction, List, Map
- Thread safety
- Constant-folding optimizations
 - Elide all future reads
 - Но только если static final StableValue<T>
 - Где бенчмарки, Лебовски?!
- *In the long term we intend to limit the reflection API so that all instance final fields can be trusted, as part of the broader shift toward integrity by default*

03

JEP 505: Structured Concurrency

Следующее Preview

Как писали деды

```
Response handle() throws ExecutionException, InterruptedException {  
    Future<String> user = executor.submit(() → findUser());  
    Future<Integer> order = executor.submit(() → fetchOrder());  
    String theUser = user.get();    // Join findUser  
    int theOrder = order.get();    // Join fetchOrder  
    return new Response(theUser, theOrder);  
}
```



505: Structured Concurrency

```
Response handle() throws InterruptedException {  
  
    try (var scope = StructuredTaskScope.open()) {  
  
        Subtask<String> user = scope.fork(() -> findUser());  
        Subtask<Integer> order = scope.fork(() -> fetchOrder());  
  
        scope.join(); // Join subtasks, propagating exceptions  
  
        // Both subtasks have succeeded, so compose their results  
        return new Response(user.get(), order.get());  
    }  
}
```

505: Structured Concurrency

```
Response handle() throws InterruptedException {  
    try (var scope = StructuredTaskScope.open()) {  
  
        Subtask<String> user = scope.fork(() → findUser());  
        Subtask<Integer> order = scope.fork(() → fetchOrder());  
  
        scope.join(); // Join subtasks, propagating exceptions  
  
        // Both subtasks have succeeded, so compose their results  
        return new Response(user.get(), order.get());  
    }  
}
```

HOBEO



505: Structured Concurrency

- По сравнению с CompletableFuture
 - Blocking paradigm -- **после успешного join()** всё доступно
 - **Нет thread leaks** при ошибках
 - **Отмена** при таймаутах/дедлайнах
 - Обработка **interrupt()**

505: Structured Concurrency

- По сравнению с CompletableFuture
 - Blocking paradigm -- **после успешного join()** всё доступно
 - **Нет thread leaks** при ошибках
 - **Отмена** при таймаутах/дедлайнах
 - Обработка **interrupt()**
- Похоже на **иерархию супервизоров** в Erlang/Akka

505: Structured Concurrency

- По сравнению с CompletableFuture
 - Blocking paradigm -- **после успешного join()** всё доступно
 - **Нет thread leaks** при ошибках
 - **Отмена** при таймаутах/дедлайнах
 - Обработка **interrupt()**
- Похоже на **иерархию супервизоров** в Erlang/Akka
- Новый **StructureViolationException**

03

JEP 507: Primitive Types in Patterns,
instanceof, and switch

Следующее Preview, вы всё видели

507: Primitive Types in Patterns, instanceof, and switch

```
switch (x.getStatus()) {  
    case 0 → "okay";  
    case 1 → "warning";  
    case 2 → "error";  
    case int i → "unknown status: " + i;  
}
```

```
sealed interface JsonValue {  
    record JsonString(String s) implements JsonValue { }  
    record JsonNumber(double d) implements JsonValue { }  
    record JsonObject(Map<String, JsonValue> map) implements JsonValue { }  
}
```

```
var json = new JsonObject(Map.of(  
    "name", new JsonString("John"),  
    "age", new JsonNumber(30))  
);
```

```
if (json instanceof JsonObject(var map) &&  
    map.get("age") instanceof JsonNumber(double a)) {  
    int age = (int) a;  
}
```

```
sealed interface JsonValue {  
    record JsonString(String s) implements JsonValue { }  
    record JsonNumber(double d) implements JsonValue { }  
    record JsonObject(Map<String, JsonValue> map) implements JsonValue { }  
}
```

```
var json = new JsonObject(Map.of(  
    "name", new JsonString("John"),  
    "age", new JsonNumber(30))  
);
```

```
if (json instanceof JsonObject(var map) &&  
    map.get("age") instanceof JsonNumber(double a)) {  
    int age = (int) a;  
}
```

```
sealed interface JsonValue {  
    record JsonString(String s) implements JsonValue { }  
    record JsonNumber(double d) implements JsonValue { }  
    record JsonObject(Map<String, JsonValue> map) implements JsonValue { }  
}
```

```
var json = new JsonObject(Map.of(  
    "name", new JsonString("John"),  
    "age", new JsonNumber(30))  
);
```

```
if (json instanceof JsonObject(var map) &&  
    map.get("age") instanceof JsonNumber(double a)) {  
    int age = (int) a;  
}
```

```
sealed interface JsonValue {  
    record JsonString(String s) implements JsonValue { }  
    record JsonNumber(double d) implements JsonValue { }  
    record JsonObject(Map<String, JsonValue> map) implements JsonValue { }  
}
```

```
var json = new JsonObject(Map.of(  
    "name", new JsonString("John"),  
    "age", new JsonNumber(30))  
);
```

```
if (json instanceof JsonObject(var map) &&  
    map.get("age") instanceof JsonNumber(double a)) {  
    int age = (int)a;  
}
```

```
sealed interface JsonValue {  
    record JsonString(String s) implements JsonValue { }  
    record JsonNumber(double d) implements JsonValue { }  
    record JsonObject(Map<String, JsonValue> map) implements JsonValue { }  
}
```

```
var json = new JsonObject(Map.of(  
    "name", new JsonString("John"),  
    "age", new JsonNumber(30))  
);
```

```
if (json instanceof JsonObject(var map) &&  
    map.get("age") instanceof JsonNumber(int a)) {  
    int age = a;  
}
```

507: Primitive Types in Patterns, instanceof, and switch



01

Сразу стабильные
новые JEРы

02

Из Preview в
стабильные JEРы

03

Следующее
Preview

04

Experimental
Incubator

04

JEP 508: Vector API (10th Incubator)

04

JEP 509: JFR CPU-Time Profiling

Experimental

И ждем вас на
следующих горячих JEP'ах!

Выгода от обновлений есть!
ОБНОВЛЯЙТЕСЬ!

```
[ERROR] Failed to execute goal  
org.apache.maven.plugins:maven-compiler-plugin:3.14.0:compile  
(default-compile) on project demo:  
Fatal error compiling: error:  
release version 25 not supported →  
[Help 1]
```

FAILURE: Build failed with an exception.

* What went wrong: 25

* Try:

> Run with --stacktrace option to get the stack trace.

> Run with --info or --debug option to get more log output.

> Run with --scan to get full insights.

> Get more help at <https://help.gradle.org>.

BUILD FAILED in 2s

Спасибо!



Кулешов



Кошелев



Цесько

Как писали деды



```
void scalarComputation(float[] a, float[] b, float[] c) {  
    for (int i = 0; i < a.length; i++) {  
        c[i] = (a[i] * a[i] + b[i] * b[i]) * -1.0f;  
    }  
}
```

508: Vector API (10th Incubator)

```
static final VectorSpecies<Float> SPECIES = FloatVector.SPECIES_PREFERRED;

void vectorComputation(float[] a, float[] b, float[] c) {
    for (int i = 0; i < a.length; i += SPECIES.length()) {
        // VectorMask<Float> m;
        var m = SPECIES.indexInRange(i, a.length);
        // FloatVector va, vb, vc;
        var va = FloatVector.fromArray(SPECIES, a, i, m);
        var vb = FloatVector.fromArray(SPECIES, b, i, m);
        var vc = va.mul(va)
                  .add(vb.mul(vb))
                  .neg();
        vc.intoArray(c, i, m);
    }
}
```

508: Vector API (10th Incubator)

- Не путаем с auto-vectorization!
- Vector computations that **reliably** compile at runtime to **optimal vector instructions** on supported CPUs
- "All Java **object allocations** are elided"
- Изменения
 - Поддержали MemorySegment
 - Перешли на Foreign Function & Memory API
 - Поддержали векторизацию Float16

508: Vector API (10th Incubator)

- Не путаем с auto-vectorization!
- Vector computations that **reliably** compile at runtime to **optimal vector instructions** on supported CPUs
- "All Java **object allocations** are elided"
- Изменения
 - Поддержали MemorySegment
 - Перешли на Foreign Function & Memory API
 - Поддержали векторизацию Float16
- Выйдет из инкубатора **не раньше финализации Valhalla**
 - value-based classes → value classes

509: JFR CPU-Time Profiling (Experimental)

Проблемы текущего execution sampler

- Только потоки, исполняющие Java код (даже если ушли в натив)
- Теряет сэмплы и не репортит потери
- Только подмножество потоков в каждый интервал сэмплирования

В результате неточные профили (особенно короткие)

509: JFR CPU-Time Profiling (Experimental)

- Linux kernel CPU timer
- Нативные функции не показывает
 - Засчитывает в Java код
- jdk.CPUTimeSample
 - Поле biased
- jdk.CPUTimeSamplesLost
 - Например, буфера переполнились
- Параметр throttle
 - (Time) 10ms -- потоки, которые провели на ядре больше 10 мс
 - (Rate) 500/s -- 500 ивентов/секунду равномерно по нативным потокам

509: JFR CPU-Time Profiling (Experimental)

	matchesKey	run()		HttpClient.<init>(URL, Proxy, int)	parse(boolean)	implFlush()
	get(Object)	run()	isASCIISuperset	New(URL, Proxy, int, boolean, HttpURLConnection)	<init>(String)	flush()
	computeIfAbsent(...)	executePrivileged	<clinit>()	New(URL, Proxy, int, HttpURLConnection)	createURI	print
	computeIfAbsent(...)	doPrivileged		HttpURLConnection.getNewHttpClient(URL, Proxy, int)	toURI(URL)	print
<init>						
getLazyLogger		getAuthCache(...)		HttpURLConnection.plainConnect0()		writeRequests
getLogger		getDefault()		HttpURLConnection.plainConnect()		writeRequests
HttpURLConnection.<clinit>()		<init>		HttpURLConnection.connect()		writeRequests()
Handler.openConnection(URL, Proxy)				HttpURLConnection.getInputStream0()		
Handler.openConnection(URL)				HttpURLConnection.getInputStream()		
URL.openConnection()				HttpURLConnection.getResponseCode()		
HttpRequestsExample.makeRequest(String)						
HttpRequestsExample.tenFastRequests()						
HttpRequestsExample\$\$Lambda\$+0x000079e5cf081a08.1168019749.run()						
Thread.runWith(Object, Runnable)						
Thread.run()						