Infrastructure

Дебаггинг в Java

Особенности работы со сложными сценариями

Александр Токарев
YTsaurus SPYT Tech Lead
(Apache Spark в YTsaurus)
в Яндексе

Александр Токарев

YTsaurus SPYT Tech Lead
(Apache Spark в YTsaurus)
в Яндексе

Пишет на Java с 2009 года

Ex: Qiwi, CleverData, Leroy Merlin

План доклада

Joker<?>
2025

Дебаггинг в Java: полное руководство

Александр Токарев

Яндекс

План доклада

1 Краткое содержание предыдущей серии

План доклада

1. Краткое содержание предыдущей серии
2. JDB — консольный отладчик в составе JDK

План доклада

1. Краткое содержание предыдущей серии
2. JDB — консольный отладчик в составе JDK
3. Отладка автогенерённого кода и байт-кода

План доклада

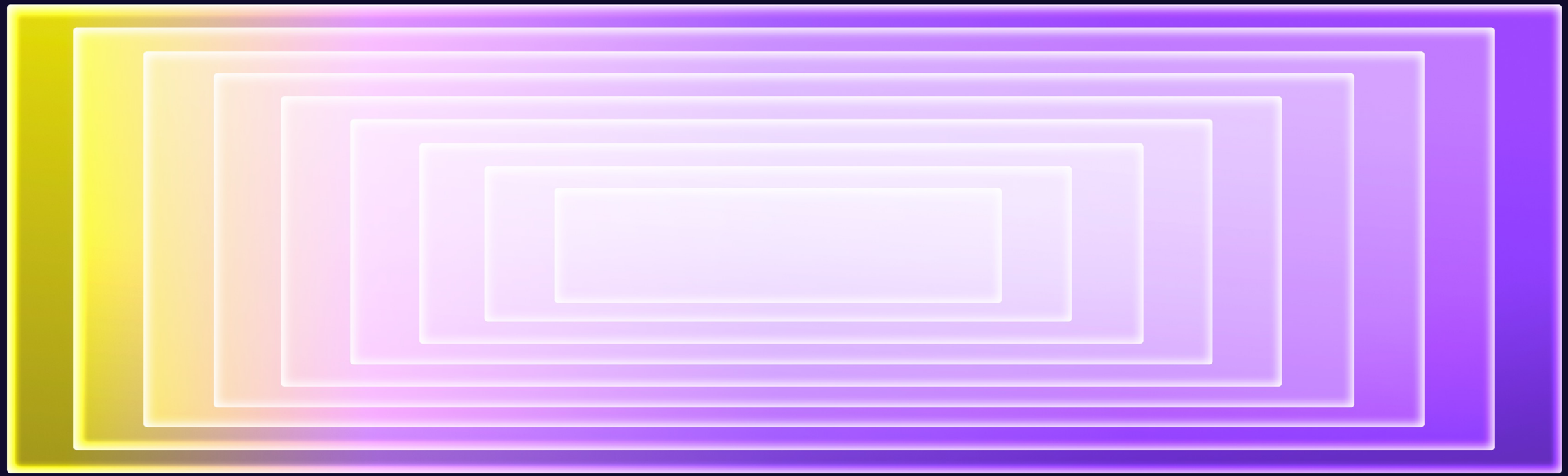
1. Краткое содержание предыдущей серии
2. JDB — консольный отладчик в составе JDK
3. Отладка автогенерённого кода и байт-кода
4. Отладка многопоточных приложений

План доклада

1. Краткое содержание предыдущей серии
2. JDB — консольный отладчик в составе JDK
3. Отладка автогенерённого кода и байт-кода
4. Отладка многопоточных приложений
5. Отладка интеграции через FFM API

План доклада

1. Краткое содержание предыдущей серии
2. JDB — консольный отладчик в составе JDK
3. Отладка автогенерённого кода и байт-кода
4. Отладка многопоточных приложений
5. Отладка интеграции через FFM API
6. Отладка за пределами JPDA



Краткое содержание предыдущей серии

Отладка и дебаггинг

Отладка и дебаггинг

Debugging переводится
как «отладка»

Отладка и дебаггинг

Debugging переводится как «отладка»

В русском языке отладка — более широкое понятие, которое, помимо дебаггинга, также включает следующие методики:

1. Логирование
2. Трейсинг
3. Профилирование
4. Тестирование

Отладка и дебаггинг

Debugging переводится как «отладка»

В русском языке отладка — более широкое понятие, которое, помимо дебаггинга, также включает следующие методики:

1. Логирование
2. Трейсинг
3. Профилирование
4. Тестирование

Более точный термин, соответствующий отладке, — Troubleshooting

Отладка и дебаггинг

Debugging переводится как «отладка»

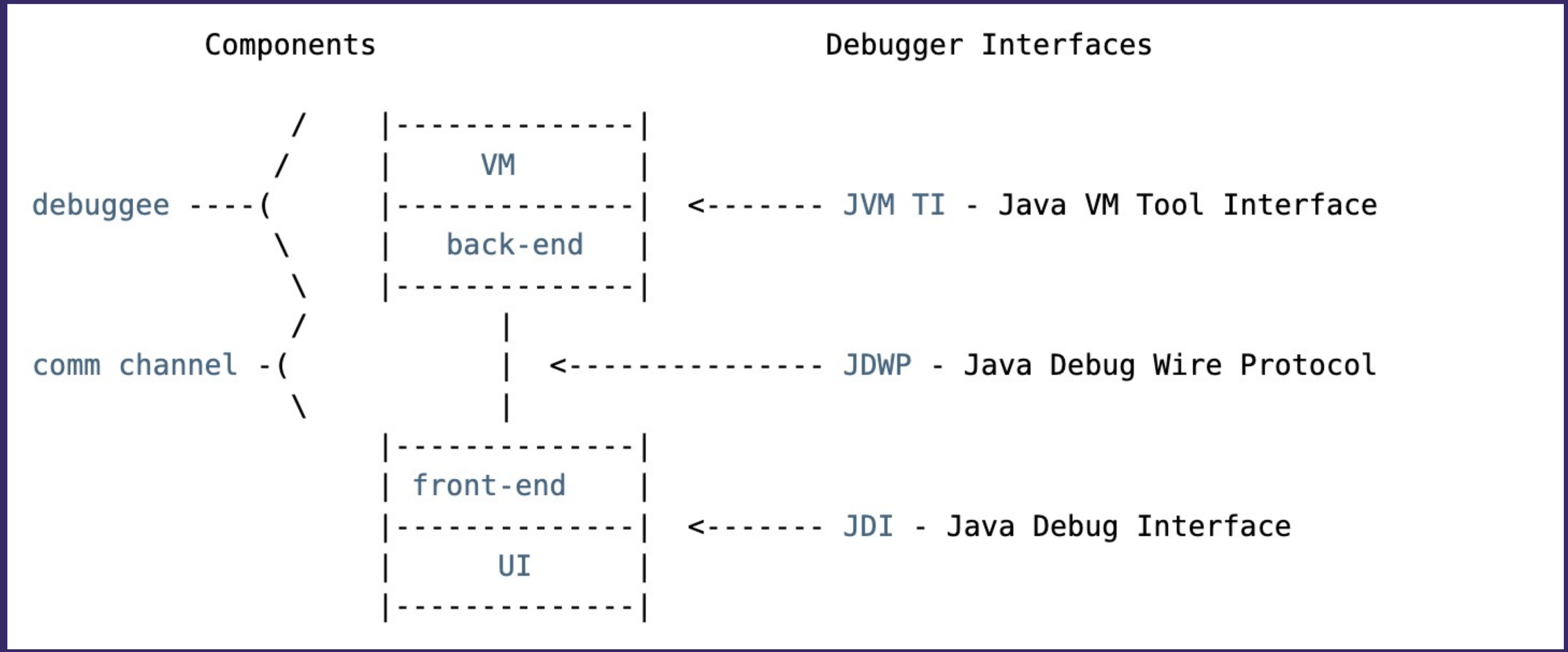
В русском языке отладка — более широкое понятие, которое, помимо дебаггинга, также включает следующие методики:

1. Логирование
2. Трейсинг
3. Профилирование
4. Тестирование

Более точный термин, соответствующий отладке, — Troubleshooting

Дебаггинг — отладка во время выполнения программы с использованием отладчика

Java Platform Debug Architecture (JPDA)



JVM TI

Нативный интерфейс внутри JVM, который взаимодействует с агентами по событийной модели

Агенты, в свою очередь, могут обращаться к JVM TI через API

API

JVM TI

Нативный интерфейс внутри JVM, который взаимодействует с агентами по событийной модели

Агенты, в свою очередь, могут обращаться к JVM TI через API

API

JVM TI-агент может быть реализован на любом языке, поддерживающем соглашения о вызове методов языка C

</>

JVM TI

Нативный интерфейс внутри JVM, который взаимодействует с агентами по событийной модели

Агенты, в свою очередь, могут обращаться к JVM TI через API

API

JVM TI-агент может быть реализован на любом языке, поддерживающем соглашения о вызове методов языка C

</>

Пример агента — JDWP-агент

JDWP

Описывает структуру сообщений между отладчиком и JVM

JDWP

Описывает структуру сообщений между отладчиком и JVM

Доступ по JDWP к отлаживаемому процессу реализован как JVM TI-агент

Пример запуска:

```
> -agentlib:jdwp=transport=
  dt_socket,server=y,suspend=y,
  address=localhost:5005
```

JDWP

Описывает структуру сообщений между отладчиком и JVM

Доступ по JDWP к отлаживаемому процессу реализован как JVM TI-агент

Пример запуска:

```
> -agentlib:jdwp=transport=
  dt_socket,server=y,suspend=y,
  address=localhost:5005
```

Основные параметры:

- Transport — используемый протокол передачи данных

JDWP

Описывает структуру сообщений между отладчиком и JVM

Доступ по JDWP к отлаживаемому процессу реализован как JVM TI-агент

Пример запуска:

```
> -agentlib:jdwp=transport=
  dt_socket,server=y,suspend=y,
  address=localhost:5005
```

Основные параметры:

- Transport — используемый протокол передачи данных
- Server:
  n — сам подключается к отладчику (характерно для запуска через IDE)
  y — ждёт подключения отладчика

JDWP

Описывает структуру сообщений между отладчиком и JVM

Доступ по JDWP к отлаживаемому процессу реализован как JVM TI-агент

Пример запуска:

```
> -agentlib:jdwp=transport=
  dt_socket,server=y,suspend=y,
  address=localhost:5005
```

Основные параметры:

- Transport — используемый протокол передачи данных
- Server:
  n — сам подключается к отладчику (характерно для запуска через IDE)
  y — ждёт подключения отладчика
- Suspend — ожидает подключения отладчика для начала выполнения

JDWP

Описывает структуру сообщений между отладчиком и JVM

Доступ по JDWP к отлаживаемому процессу реализован как JVM TI-агент

Пример запуска:

```
> -agentlib:jdwp=transport=
  dt_socket,server=y,suspend=y,
  address=localhost:5005
```

Основные параметры:

- Transport — используемый протокол передачи данных
- Server:
  n — сам подключается к отладчику (характерно для запуска через IDE)
  y — ждёт подключения отладчика
- Suspend — ожидает подключения отладчика для начала выполнения
- Address — хост и порт, к которому должен обращаться отладчик

JDI

Pure Java реализация клиентской части для взаимодействия по JDWP

JDI

Pure Java реализация клиентской части для взаимодействия по JDWP

Позволяет разрабатывать собственные дебаггеры

JDI

Pure Java реализация клиентской части для взаимодействия по JDWP

Дебаггеры в IDE прямым или косвенным образом используют JDI

Позволяет разрабатывать собственные дебаггеры

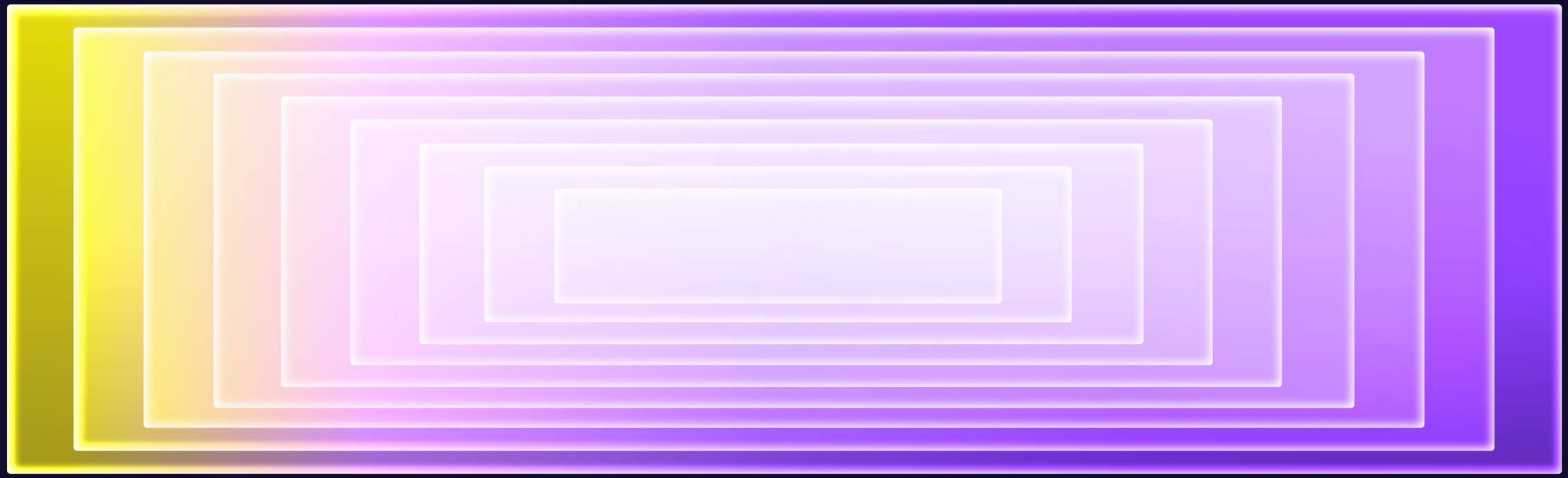
JDI

Pure Java реализация клиентской части для взаимодействия по JDWP

Дебаггеры в IDE прямым или косвенным образом используют JDI

Позволяет разрабатывать собственные дебаггеры

Наиболее простой пример JDI-приложения — консольный отладчик JDB



JDB — консольный отладчик в составе JDK

JDB

Работает в средах без GUI через CLI

JDB

Работает в средах без GUI
через CLI

Нетребователен
к ресурсам

JDB

Работает в средах без GUI через CLI

Возможность автоматизации отладки через скрипты

Нетребователен к ресурсам

Основные группы команд JDB

Управление выполнением

`> run, cont, step, step up, stepi, next`

Работа с точками останова

(breakpoints)

`> stop, clear`

Просмотр состояния

`> print, eval, dump, locals, list, where, wherei`

Изменение и мониторинг переменных

`> set, watch, unwatch`

Работа с потоками

`> threads, thread, suspend, resume, lock, threadlocks`

Информация о загруженных классах

`> classes, class, methods, fields`

`</>` Прочие команды: `trace, monitor, kill…`

Пример работы с JDB

```
 3> public class DebugDemo {
 4>     static void main(String[] args) {
 5>         int x = 5;
 6>         int y = 10;
 7>         Calculator calc = new Calculator(2);
 8>         int result = calc.add(x, y);
 9>         calc.setMultiplier(4);
10>         IO.println("Result: " + result);
11>     }
12> }
```

Пример работы с JDB

```
 3> public class Calculator {
 4>     private int multiplier;
 5>
 6>     Calculator(int multiplier) {
 7>         this.multiplier = multiplier;
 8>     }
 9>
10>     public int add(int a, int b) {
11>         int sum = a + b;
12>         sum = multiply(sum);
13>         return sum;
14>     }
15>
16>     private int multiply(int value) {
```

Пример работы с JDB

```
10>     public int add(int a, int b) {
11>         int sum = a + b;
12>         sum = multiply(sum);
13>         return sum;
14>     }
15>
16>     private int multiply(int value) {
17>         return value * multiplier;
18>     }
19>
20>     public void setMultiplier(int multiplier) {
21>         this.multiplier = multiplier;
22>     }
23> }
```

Пример работы с JDB: сборка и запуск

Пример работы с JDB: сборка и запуск

https://github.com/alextokarew/java-debugging

Пример работы с JDB: сборка и запуск

https://github.com/alextokarew/java-debugging

```
./gradlew build
```

Пример работы с JDB: сборка и запуск

https://github.com/alextokarew/java-debugging

```
./gradlew build
```

```
jdb -sourcepath app/src/main/java \
    -classpath app/build/classes/java/main \
    com.github.alextokarew.javadebugging.jdb.DebugDemo
```

Пример работы с JDB: сборка и запуск

https://github.com/alextokarew/java-debugging

```
./gradlew build
```

```
jdb -sourcepath app/src/main/java \
    -classpath app/build/classes/java/main \
    com.github.alextokarew.javadebugging.jdb.DebugDemo
```

Пример работы с JDB: сборка и запуск

https://github.com/alextokarew/java-debugging

```
./gradlew build
```

```
jdb -sourcepath app/src/main/java \
    -classpath app/build/classes/java/main \
    com.github.alextokarew.javadebugging.jdb.DebugDemo
```

Пример работы с JDB

Пример работы с JDB

```
> stop at com.github.alextokarew.javadebugging.jdb.DebugDemo.main
```

Пример работы с JDB

```
> stop at com.github.alextokarew.javadebugging.jdb.DebugDemo.main
  Deferring breakpoint ....DebugDemo.main.
  It will be set after the class is loaded.
```

Пример работы с JDB

```
> stop at com.github.alextokarew.javadebugging.jdb.DebugDemo.main
  Deferring breakpoint ....DebugDemo.main.
  It will be set after the class is loaded.
> run
```

Пример работы с JDB

```
> stop at com.github.alextokarew.javadebugging.jdb.DebugDemo.main
  Deferring breakpoint ....DebugDemo.main.
  It will be set after the class is loaded.
> run
  ...
  Breakpoint hit: "thread=main", DebugDemo.main(), line=5 bci=0
  5             int x = 5;
```

Пример работы с JDB

```
> stop at com.github.alextokarew.javadebugging.jdb.DebugDemo.main
  Deferring breakpoint ....DebugDemo.main.
  It will be set after the class is loaded.
> run
  ...
  Breakpoint hit: "thread=main", DebugDemo.main(), line=5 bci=0
  5               int x = 5;
```

Пример работы с JDB

```
 3> public class DebugDemo {
 4>     static void main(String[] args) {
 5>         int x = 5;
 6>         int y = 10;
 7>         Calculator calc = new Calculator(2);
 8>         int result = calc.add(x, y);
 9>         calc.setMultiplier(4);
10>         IO.println("Result: " + result);
11>     }
12> }
```

Пример работы с JDB

```
> stop at com.github.alextokarew.javadebugging.jdb.DebugDemo.main
  Deferring breakpoint ....DebugDemo.main.
  It will be set after the class is loaded.
> run
  ...
  Breakpoint hit: "thread=main", DebugDemo.main(), line=5 bci=0
  5               int x = 5;
> next
```

Пример работы с JDB

```
> stop at com.github.alextokarew.javadebugging.jdb.DebugDemo.main
  Deferring breakpoint ....DebugDemo.main.
  It will be set after the class is loaded.
> run
  ...
  Breakpoint hit: "thread=main", DebugDemo.main(), line=5 bci=0
  5               int x = 5;
> next
  Step completed: "thread=main", DebugDemo.main(), line=6 bci=2
  6               int y = 10;
```

Пример работы с JDB

```
> stop at com.github.alextokarew.javadebugging.jdb.DebugDemo.main
  Deferring breakpoint ....DebugDemo.main.
  It will be set after the class is loaded.
> run
  ...
  Breakpoint hit: "thread=main", DebugDemo.main(), line=5 bci=0
  5               int x = 5;
> next
  Step completed: "thread=main", DebugDemo.main(), line=6 bci=2
  6               int y = 10;
> !!
```

Пример работы с JDB

```
> stop at com.github.alextokarew.javadebugging.jdb.DebugDemo.main
  Deferring breakpoint ....DebugDemo.main.
  It will be set after the class is loaded.
> run
  ...
  Breakpoint hit: "thread=main", DebugDemo.main(), line=5 bci=0
  5               int x = 5;
> next
  Step completed: "thread=main", DebugDemo.main(), line=6 bci=2
  6               int y = 10;
> !!
  Step completed: "thread=main", DebugDemo.main(), line=7 bci=5
  7               Calculator calc = new Calculator(2);
```

Пример работы с JDB

```
> stop at com.github.alextokarew.javadebugging.jdb.DebugDemo.main
  Deferring breakpoint ....DebugDemo.main.
  It will be set after the class is loaded.
> run
  ...
  Breakpoint hit: "thread=main", DebugDemo.main(), line=5 bci=0
  5               int x = 5;
> next
  Step completed: "thread=main", DebugDemo.main(), line=6 bci=2
  6               int y = 10;
> !!
  Step completed: "thread=main", DebugDemo.main(), line=7 bci=5
  7               Calculator calc = new Calculator(2);
> !!
```

Пример работы с JDB

```
   It will be set after the class is loaded.
> run
   ...
   Breakpoint hit: "thread=main", DebugDemo.main(), line=5 bci=0
   5               int x = 5;
> next
   Step completed: "thread=main", DebugDemo.main(), line=6 bci=2
   6               int y = 10;
> !!
   Step completed: "thread=main", DebugDemo.main(), line=7 bci=5
   7               Calculator calc = new Calculator(2);
> !!
   Step completed: "thread=main", DebugDemo.main(), line=8 bci=14
   8               int result = calc.add(x, y);
```

Пример работы с JDB

```
> run
  ...
  Breakpoint hit: "thread=main", DebugDemo.main(), line=5 bci=0
  5               int x = 5;
> next
  Step completed: "thread=main", DebugDemo.main(), line=6 bci=2
  6               int y = 10;
> !!
  Step completed: "thread=main", DebugDemo.main(), line=7 bci=5
  7               Calculator calc = new Calculator(2);
> !!
  Step completed: "thread=main", DebugDemo.main(), line=8 bci=14
  8               int result = calc.add(x, y);
> locals
```

Пример работы с JDB

```
> !!
  Step completed: "thread=main", DebugDemo.main(), line=7 bci=5
  7             Calculator calc = new Calculator(2);
> !!
  Step completed: "thread=main", DebugDemo.main(), line=8 bci=14
  8             int result = calc.add(x, y);
> locals
  Method arguments:
  args = instance of java.lang.String[0] (id=436)
  Local variables:
  x = 5
  y = 10
  calc = instance of
  com.github.alextokarew.javadebugging.jdb.Calculator(id=437)
```

Пример работы с JDB

```
  Step completed: "thread=main", DebugDemo.main(), line=7 bci=5
  7             Calculator calc = new Calculator(2);
> !!
  Step completed: "thread=main", DebugDemo.main(), line=8 bci=14
  8             int result = calc.add(x, y);
> locals
  Method arguments:
  args = instance of java.lang.String[0] (id=436)
  Local variables:
  x = 5
  y = 10
  calc = instance of
  com.github.alextokarew.javadebugging.jdb.Calculator(id=437)
> dump calc
```

Пример работы с JDB

```
  Step completed: "thread=main", DebugDemo.main(), line=8 bci=14
  8             int result = calc.add(x, y);
> locals
  Method arguments:
  args = instance of java.lang.String[0] (id=436)
  Local variables:
  x = 5
  y = 10
  calc = instance of
  com.github.alextokarew.javadebugging.jdb.Calculator(id=437)
> dump calc
   calc = {
      multiplier: 2
  }
```

Пример работы с JDB

```
 3> public class DebugDemo {
 4>     static void main(String[] args) {
 5>         int x = 5;
 6>         int y = 10;
 7>         Calculator calc = new Calculator(2);
 8>         int result = calc.add(x, y);
 9>         calc.setMultiplier(4);
10>         IO.println("Result: " + result);
11>     }
12> }
```

Пример работы с JDB

```
   8            int result = calc.add(x, y);
> locals
   Method arguments:
   args = instance of java.lang.String[0] (id=436)
   Local variables:
   x = 5
   y = 10
   calc = instance of
   com.github.alextokarew.javadebugging.jdb.Calculator(id=437)
> dump calc
    calc = {
       multiplier: 2
   }
> step
```

Пример работы с JDB

```
  Method arguments:
  args = instance of java.lang.String[0] (id=436)
  Local variables:
  x = 5
  y = 10
  calc = instance of
  com.github.alextokarew.javadebugging.jdb.Calculator(id=437)
> dump calc
   calc = {
      multiplier: 2
  }
> step
  Step completed: "thread=main", Calculator.add(), line=11 bci=0
  11              int sum = a + b;
```

Пример работы с JDB

```
  args = instance of java.lang.String[0] (id=436)
  Local variables:
  x = 5
  y = 10
  calc = instance of
  com.github.alextokarew.javadebugging.jdb.Calculator(id=437)
> dump calc
   calc = {
      multiplier: 2
  }
> step
  Step completed: "thread=main", Calculator.add(), line=11 bci=0
  11              int sum = a + b;
> next
```

Пример работы с JDB

```
  x = 5
  y = 10
  calc = instance of
  com.github.alextokarew.javadebugging.jdb.Calculator(id=437)
> dump calc
   calc = {
      multiplier: 2
  }
> step
  Step completed: "thread=main", Calculator.add(), line=11 bci=0
  11              int sum = a + b;
> next
  Step completed: "thread=main", Calculator.add(), line=12 bci=4
  12              sum = multiply(sum);
```

Пример работы с JDB

```
   y = 10
   calc = instance of
   com.github.alextokarew.javadebugging.jdb.Calculator(id=437)
> dump calc
    calc = {
       multiplier: 2
   }
> step
   Step completed: "thread=main", Calculator.add(), line=11 bci=0
   11              int sum = a + b;
> next
   Step completed: "thread=main", Calculator.add(), line=12 bci=4
   12              sum = multiply(sum);
> step
```

Пример работы с JDB

```
  com.github.alextokarew.javadebugging.jdb.Calculator(id=437)
> dump calc
   calc = {
      multiplier: 2
  }
> step
  Step completed: "thread=main", Calculator.add(), line=11 bci=0
  11              int sum = a + b;
> next
  Step completed: "thread=main", Calculator.add(), line=12 bci=4
  12              sum = multiply(sum);
> step
  Step completed: "thread=...", Calculator.multiply(), line=17 bci=0
  17              return value * multiplier;
```

Пример работы с JDB

```
> dump calc
 calc = {
    multiplier: 2
}
> step
Step completed: "thread=main", Calculator.add(), line=11 bci=0
11              int sum = a + b;
> next
Step completed: "thread=main", Calculator.add(), line=12 bci=4
12              sum = multiply(sum);
> step
Step completed: "thread=...", Calculator.multiply(), line=17 bci=0
17              return value * multiplier;
> where
```

Пример работы с JDB

```
   }
> step
   Step completed: "thread=main", Calculator.add(), line=11 bci=0
   11              int sum = a + b;
> next
   Step completed: "thread=main", Calculator.add(), line=12 bci=4
   12              sum = multiply(sum);
> step
   Step completed: "thread=...", Calculator.multiply(), line=17 bci=0
   17              return value * multiplier;
> where
     [1] Calculator.multiply (Calculator.java:17)
     [2] Calculator.add (Calculator.java:12)
     [3] DebugDemo.main (DebugDemo.java:8)
```

Пример работы с JDB

```
> step
  Step completed: "thread=main", Calculator.add(), line=11 bci=0
  11              int sum = a + b;
> next
  Step completed: "thread=main", Calculator.add(), line=12 bci=4
  12              sum = multiply(sum);
> step
  Step completed: "thread=...", Calculator.multiply(), line=17 bci=0
  17              return value * multiplier;
> where
    [1] Calculator.multiply (Calculator.java:17)
    [2] Calculator.add (Calculator.java:12)
    [3] DebugDemo.main (DebugDemo.java:8)
> eval value * 3
```

Пример работы с JDB

```
  Step completed: "thread=main", Calculator.add(), line=11 bci=0
  11              int sum = a + b;
> next
  Step completed: "thread=main", Calculator.add(), line=12 bci=4
  12              sum = multiply(sum);
> step
  Step completed: "thread=...", Calculator.multiply(), line=17 bci=0
  17              return value * multiplier;
> where
    [1] Calculator.multiply (Calculator.java:17)
    [2] Calculator.add (Calculator.java:12)
    [3] DebugDemo.main (DebugDemo.java:8)
> eval value * 3
   value * 3 = 45
```

Пример работы с JDB

```
   11              int sum = a + b;
> next
   Step completed: "thread=main", Calculator.add(), line=12 bci=4
   12              sum = multiply(sum);
> step
   Step completed: "thread=...", Calculator.multiply(), line=17 bci=0
   17              return value * multiplier;
> where
     [1] Calculator.multiply (Calculator.java:17)
     [2] Calculator.add (Calculator.java:12)
     [3] DebugDemo.main (DebugDemo.java:8)
> eval value * 3
    value * 3 = 45
> set value = 20
```

Пример работы с JDB

```
> next
  Step completed: "thread=main", Calculator.add(), line=12 bci=4
  12              sum = multiply(sum);
> step
  Step completed: "thread=...", Calculator.multiply(), line=17 bci=0
  17              return value * multiplier;
> where
    [1] Calculator.multiply (Calculator.java:17)
    [2] Calculator.add (Calculator.java:12)
    [3] DebugDemo.main (DebugDemo.java:8)
> eval value * 3
   value * 3 = 45
> set value = 20
   value = 20 = 20
```

Пример работы с JDB

```
 3> public class Calculator {
 4>     private int multiplier;
 5>
 6>     Calculator(int multiplier) {
 7>         this.multiplier = multiplier;
 8>     }
 9>
10>     public int add(int a, int b) {
11>         int sum = a + b;
12>         sum = multiply(sum);
13>         return sum;
14>     }
15>
16>     private int multiply(int value) {
```

Пример работы с JDB

```
  Step completed: "thread=main", Calculator.add(), line=12 bci=4
  12              sum = multiply(sum);
> step
  Step completed: "thread=...", Calculator.multiply(), line=17 bci=0
  17              return value * multiplier;
> where
    [1] Calculator.multiply (Calculator.java:17)
    [2] Calculator.add (Calculator.java:12)
    [3] DebugDemo.main (DebugDemo.java:8)
> eval value * 3
   value * 3 = 45
> set value = 20
   value = 20 = 20
> watch Calculator.multiplier
```

Пример работы с JDB

```
   12              sum = multiply(sum);
> step
   Step completed: "thread=...", Calculator.multiply(), line=17 bci=0
   17              return value * multiplier;
> where
     [1] Calculator.multiply (Calculator.java:17)
     [2] Calculator.add (Calculator.java:12)
     [3] DebugDemo.main (DebugDemo.java:8)
> eval value * 3
    value * 3 = 45
> set value = 20
    value = 20 = 20
> watch Calculator.multiplier
   Set watch modification of Calculator.multiplier
```

Пример работы с JDB

```
> step
  Step completed: "thread=...", Calculator.multiply(), line=17 bci=0
  17              return value * multiplier;
> where
    [1] Calculator.multiply (Calculator.java:17)
    [2] Calculator.add (Calculator.java:12)
    [3] DebugDemo.main (DebugDemo.java:8)
> eval value * 3
   value * 3 = 45
> set value = 20
   value = 20 = 20
> watch Calculator.multiplier
  Set watch modification of Calculator.multiplier
> cont
```

Пример работы с JDB

```
> where
  [1] Calculator.multiply (Calculator.java:17)
  [2] Calculator.add (Calculator.java:12)
  [3] DebugDemo.main (DebugDemo.java:8)
> eval value * 3
 value * 3 = 45
> set value = 20
 value = 20 = 20
> watch Calculator.multiplier
Set watch modification of Calculator.multiplier
> cont
Field (Calculator.multiplier) is 2, will be 4: "thread=main",
Calculator.setMultiplier(), line=21 bci=2
21              this.multiplier = multiplier;
```

Пример работы с JDB

```
    [1] Calculator.multiply (Calculator.java:17)
    [2] Calculator.add (Calculator.java:12)
    [3] DebugDemo.main (DebugDemo.java:8)
> eval value * 3
  value * 3 = 45
> set value = 20
  value = 20 = 20
> watch Calculator.multiplier
  Set watch modification of Calculator.multiplier
> cont
  Field (Calculator.multiplier) is 2, will be 4: "thread=main",
  Calculator.setMultiplier(), line=21 bci=2
  21              this.multiplier = multiplier;
> where
```

Пример работы с JDB

```
    [3] DebugDemo.main (DebugDemo.java:8)
> eval value * 3
  value * 3 = 45
> set value = 20
  value = 20 = 20
> watch Calculator.multiplier
  Set watch modification of Calculator.multiplier
> cont
  Field (Calculator.multiplier) is 2, will be 4: "thread=main",
  Calculator.setMultiplier(), line=21 bci=2
  21              this.multiplier = multiplier;
> where
    [1] Calculator.setMultiplier (Calculator.java:21)
    [2] DebugDemo.main (DebugDemo.java:9)
```

Пример работы с JDB

```
 3> public class DebugDemo {
 4>     static void main(String[] args) {
 5>         int x = 5;
 6>         int y = 10;
 7>         Calculator calc = new Calculator(2);
 8>         int result = calc.add(x, y);
 9>         calc.setMultiplier(4);
10>         IO.println("Result: " + result);
11>     }
12> }
```

Пример работы с JDB

```
> eval value * 3
   value * 3 = 45
> set value = 20
   value = 20 = 20
> watch Calculator.multiplier
  Set watch modification of Calculator.multiplier
> cont
  Field (Calculator.multiplier) is 2, will be 4: "thread=main",
  Calculator.setMultiplier(), line=21 bci=2
  21              this.multiplier = multiplier;
> where
    [1] Calculator.setMultiplier (Calculator.java:21)
    [2] DebugDemo.main (DebugDemo.java:9)
> cont
```

Пример работы с JDB

```
    value = 20 = 20
> watch Calculator.multiplier
  Set watch modification of Calculator.multiplier
> cont
  Field (Calculator.multiplier) is 2, will be 4: "thread=main",
  Calculator.setMultiplier(), line=21 bci=2
  21              this.multiplier = multiplier;
> where
    [1] Calculator.setMultiplier (Calculator.java:21)
    [2] DebugDemo.main (DebugDemo.java:9)
> cont
  Result: 40

  The application exited
```

JDB в CI/CD

JDB в CI/CD

Воспроизведение нештатных ситуаций, для которых не предусмотрено хуков в коде

JDB в CI/CD

Воспроизведение нештатных ситуаций, для которых не предусмотрено хуков в коде

Отладка флапающих тестов, не воспроизводящихся локально

JDB в CI/CD

Воспроизведение нештатных ситуаций, для которых не предусмотрено хуков в коде

Отладка флапающих тестов, не воспроизводящихся локально

Сбор расширенной отладочной информации

Пример использования JDB в CI/CD

Пример использования JDB в CI/CD

```
java -agentlib:jdwp=..suspend=y,.. -cp .. MyTestsSuite
```

Пример использования JDB в CI/CD

```
java -agentlib:jdwp=..suspend=y,.. -cp .. MyTestsSuite

jdb -attach localhost:5005 < debug_script.jdb \
  > debug_output.txt
```

Пример использования JDB в CI/CD

```
java -agentlib:jdwp=..suspend=y,.. -cp .. MyTestsSuite

jdb -attach localhost:5005 < debug_script.jdb \
  > debug_output.txt

# debug_script.jdb
stop at MyTest.testSomething
cont
```

Пример использования JDB в CI/CD

```
java -agentlib:jdwp=..suspend=y,.. -cp .. MyTestsSuite

jdb -attach localhost:5005 < debug_script.jdb \
  > debug_output.txt

# debug_script.jdb
stop at MyTest.testSomething
cont
print someVariable
dump someObject
```

Пример использования JDB в CI/CD

```
java -agentlib:jdwp=..suspend=y,.. -cp .. MyTestsSuite

jdb -attach localhost:5005 < debug_script.jdb \
  > debug_output.txt

# debug_script.jdb
stop at MyTest.testSomething
cont
print someVariable
dump someObject
kill 1 new java.lang.Exception("BOOOM")
```

Достоинства JDB

Достоинства JDB

Встроен
в JDK

Достоинства JDB

Встроен
в JDK

Минимальные
требования
к ресурсам

Достоинства JDB

Встроен
в JDK

Минимальные
требования
к ресурсам

Возможность
встраивать
в CI/CD-
процессы

Достоинства JDB

Встроен
в JDK

Минимальные
требования
к ресурсам

Возможность
встраивать
в CI/CD-
процессы

Отладка
на уровне
байт-кода

Инструкции stepi
и wherei

Недостатки JDB

Недостатки JDB

Нет условных точек останова

Недостатки JDB

Нет условных точек останова

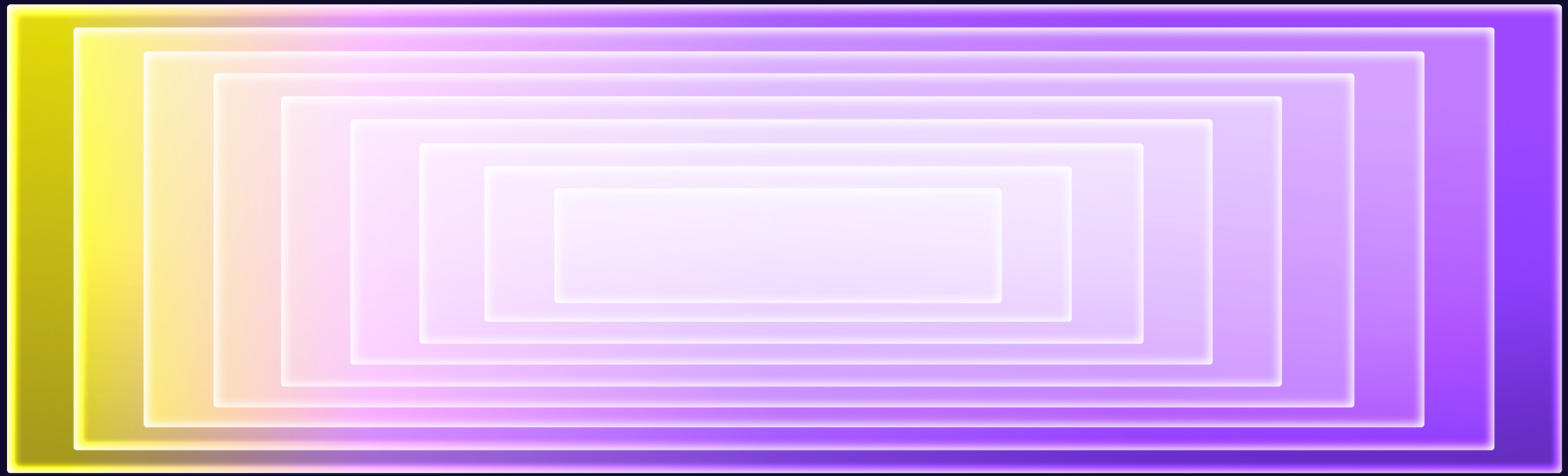
Интерфейс CLI с большим количеством команд

Недостатки JDB

Нет условных точек останова

Интерфейс CLI с большим количеством команд

Нет истории выполнения



Отладка
на уровне байт-кода

Когда нужно отлаживать байт-код?

Когда нужно отлаживать байт-код?

Работа со сторонними
библиотеками
без отладочной
информации в байт-коде

Когда нужно отлаживать байт-код?

Работа со сторонними библиотеками без отладочной информации в байт-коде

Отладка инструментирования и трансформации байт-кода

Например, через javaagent

Когда нужно отлаживать байт-код?

Работа со сторонними библиотеками без отладочной информации в байт-коде

Отладка инструментирования и трансформации байт-кода

Например, через javaagent

Автогенерация байт-кода в приложении

Отладочные символы в байт-коде

Отладочные символы в байт-коде

При компиляции через javac добавляются флагом –g

Отладочные символы в байт-коде

При компиляции через javac добавляются флагом –g

Другие компиляторы (kotlinc, scalac и т. д.) также могут добавлять отладочные символы

Отладочные символы в байт-коде

При компиляции через javac добавляются флагом –g

Другие компиляторы (kotlinc, scalac и т. д.) также могут добавлять отладочные символы

</>

Атрибут класса SourceFile

Информация об исходном файле

```
>-g:source
```

Отладочные символы в байт-коде

При компиляции через javac добавляются флагом –g

Другие компиляторы (kotlinc, scalac и т. д.) также могут добавлять отладочные символы

</>

Атрибут класса SourceFile

Информация об исходном файле

`>-g:source`

Атрибут метода LocalVariableTable

информация о локальных переменных и их области видимости

`>-g:vars`

Отладочные символы в байт-коде

При компиляции через javac добавляются флагом –g

Другие компиляторы (kotlinc, scalac и т. д.) также могут добавлять отладочные символы

</>

Атрибут класса SourceFile

Информация об исходном файле

`>-g:source`

Атрибут метода LocalVariableTable

информация о локальных переменных и их области видимости

`>-g:vars`

Атрибут метода LineNumberTable

Соответствие строк исходного файла инструкциям байт-кода

`>-g:lines`

Пример на Kotlin

```
1> fun main() {
2>     val a = 1
3>     val b = 2
4>     val c = a*2 + b*3
5>     println("c = " + c)
6> }
```

javap -v SampleClassKt

javap -v SampleClassKt

```
{
public static final void main();
  Code:
  ...

  ...
}
```

javap -v SampleClassKt

```
...
public static final void main();
    Code:
       0: iconst_1
       1: istore_0
       2: iconst_2
       3: istore_1
       4: iload_0
       5: iconst_2
       6: imul
       7: iload_1
       8: iconst_3
       9: imul
      10: iadd
      11: istore_2
      12: new           #11
      ...
      42: return
```

```
    {
  public static final void main();
    Code:
    ...

    ...
  }
```

javap -v SampleClassKt

```
...
public static final void main();
    Code:
       0: iconst_1
       1: istore_0
       2: iconst_2
       3: istore_1
       4: iload_0
       5: iconst_2
       6: imul
       7: iload_1
       8: iconst_3
       9: imul
      10: iadd
      11: istore_2
      12: new           #11
      ...
      42: return
```

```
    {
public static final void main();
  Code:
  ...
  LineNumberTable:
    line 2: 0
    line 3: 2
    line 4: 4
    line 5: 12
    line 6: 42

    ...
}
```

javap -v SampleClassKt

```
...
public static final void main();
    Code:
       0: iconst_1
       1: istore_0
       2: iconst_2
       3: istore_1
       4: iload_0
       5: iconst_2
       6: imul
       7: iload_1
       8: iconst_3
       9: imul
      10: iadd
      11: istore_2
      12: new           #11
      ...
      42: return
```

```
    {
public static final void main();
  Code:
  ...
  LineNumberTable:
    line 2: 0
    line 3: 2
    line 4: 4
    line 5: 12
    line 6: 42

    ...
}
```

Пример на Kotlin

```
1> fun main() {
2>     val a = 1
3>     val b = 2
4>     val c = a*2 + b*3
5>     println("c = " + c)
6> }
```

javap -v SampleClassKt

```
...
public static final void main();
    Code:
       0: iconst_1
       1: istore_0
       2: iconst_2
       3: istore_1
       4: iload_0
       5: iconst_2
       6: imul
       7: iload_1
       8: iconst_3
       9: imul
      10: iadd
      11: istore_2
      12: new           #11
      ...
      42: return
```

```
    {
public static final void main();
  Code:
  ...
  LineNumberTable:
    line 2: 0
    line 3: 2
    line 4: 4
    line 5: 12
    line 6: 42

    ...
}
```

javap -v SampleClassKt

```
...
public static final void main();
    Code:
       0: iconst_1
       1: istore_0
       2: iconst_2
       3: istore_1
       4: iload_0
       5: iconst_2
       6: imul
       7: iload_1
       8: iconst_3
       9: imul
      10: iadd
      11: istore_2
      12: new           #11
      ...
      42: return
```

```
    {
public static final void main();
  Code:
  ...
  LineNumberTable:
    line 2: 0
    line 3: 2
    line 4: 4
    line 5: 12
    line 6: 42
  LocalVariableTable:
    Start  Length  Slot  Name   Signature
       12      31     2     c   I
        4      39     1     b   I
        2      41     0     a   I

  ...
}
```

javap -v SampleClassKt

```
...
public static final void main();
    Code:
       0: iconst_1
       1: istore_0
       2: iconst_2
       3: istore_1
       4: iload_0
       5: iconst_2
       6: imul
       7: iload_1
       8: iconst_3
       9: imul
      10: iadd
      11: istore_2
      12: new           #11
      ...
      42: return
```

```
    {
public static final void main();
  Code:
  ...
  LineNumberTable:
    line 2: 0
    line 3: 2
    line 4: 4
    line 5: 12
    line 6: 42
  LocalVariableTable:
    Start  Length  Slot  Name   Signature
       12      31     2     c   I
        4      39     1     b   I
        2      41     0     a   I

  ...
}
```

Пример на Kotlin

```
1> fun main() {
2>     val a = 1
3>     val b = 2
4>     val c = a*2 + b*3
5>     println("c = " + c)
6> }
```

javap -v SampleClassKt

```
...
public static final void main();
    Code:
       0: iconst_1
       1: istore_0
       2: iconst_2
       3: istore_1
       4: iload_0
       5: iconst_2
       6: imul
       7: iload_1
       8: iconst_3
       9: imul
      10: iadd
      11: istore_2
      12: new           #11
      ...
      42: return
```

```
    {
public static final void main();
  Code:
  ...
  LineNumberTable:
    line 2: 0
    line 3: 2
    line 4: 4
    line 5: 12
    line 6: 42
  LocalVariableTable:
    Start  Length  Slot  Name   Signature
       12      31     2     c   I
        4      39     1     b   I
        2      41     0     a   I

  ...
}
```

javap -v SampleClassKt

```
...
public static final void main();
    Code:
       0: iconst_1
       1: istore_0
       2: iconst_2
       3: istore_1
       4: iload_0
       5: iconst_2
       6: imul
       7: iload_1
       8: iconst_3
       9: imul
      10: iadd
      11: istore_2
      12: new           #11
      ...
      42: return
```

```
    {
public static final void main();
  Code:
  ...
  LineNumberTable:
    line 2: 0
    line 3: 2
    line 4: 4
    line 5: 12
    line 6: 42
  LocalVariableTable:
    Start  Length  Slot  Name   Signature
       12      31     2     c   I
        4      39     1     b   I
        2      41     0     a   I

  ...
}
```

javap -v SampleClassKt

```
...
public static final void main();
    Code:
       0: iconst_1
       1: istore_0
       2: iconst_2
       3: istore_1
       4: iload_0
       5: iconst_2
       6: imul
       7: iload_1
       8: iconst_3
       9: imul
      10: iadd
      11: istore_2
      12: new           #11
      ...
      42: return
```

```
    {
public static final void main();
  Code:
  ...
  LineNumberTable:
    line 2: 0
    line 3: 2
    line 4: 4
    line 5: 12
    line 6: 42
  LocalVariableTable:
    Start  Length  Slot  Name   Signature
       12      31     2     c   I
        4      39     1     b   I
        2      41     0     a   I

  ...
}
```

javap -v SampleClassKt

```
...
public static final void main();
    Code:
       0: iconst_1
       1: istore_0
       2: iconst_2
       3: istore_1
       4: iload_0
       5: iconst_2
       6: imul
       7: iload_1
       8: iconst_3
       9: imul
      10: iadd
      11: istore_2
      12: new           #11
      ...
      42: return
```

```
    {
public static final void main();
  Code:
  ...
  LineNumberTable:
    line 2: 0
    line 3: 2
    line 4: 4
    line 5: 12
    line 6: 42
  LocalVariableTable:
    Start  Length  Slot  Name   Signature
       12      31     2     c   I
        4      39     1     b   I
        2      41     0     a   I

    ...
}
SourceFile: "SampleClass.kt"
```

Инструменты для отладки на уровне байт-кода

Инструменты для отладки на уровне байт-кода

Нет универсального инструмента для отладки jvm байт-кода

Инструменты для отладки на уровне байт-кода

Нет универсального инструмента для отладки jvm байт-кода

javap

Просмотр инструкций байт-кода

Инструменты для отладки на уровне байт-кода

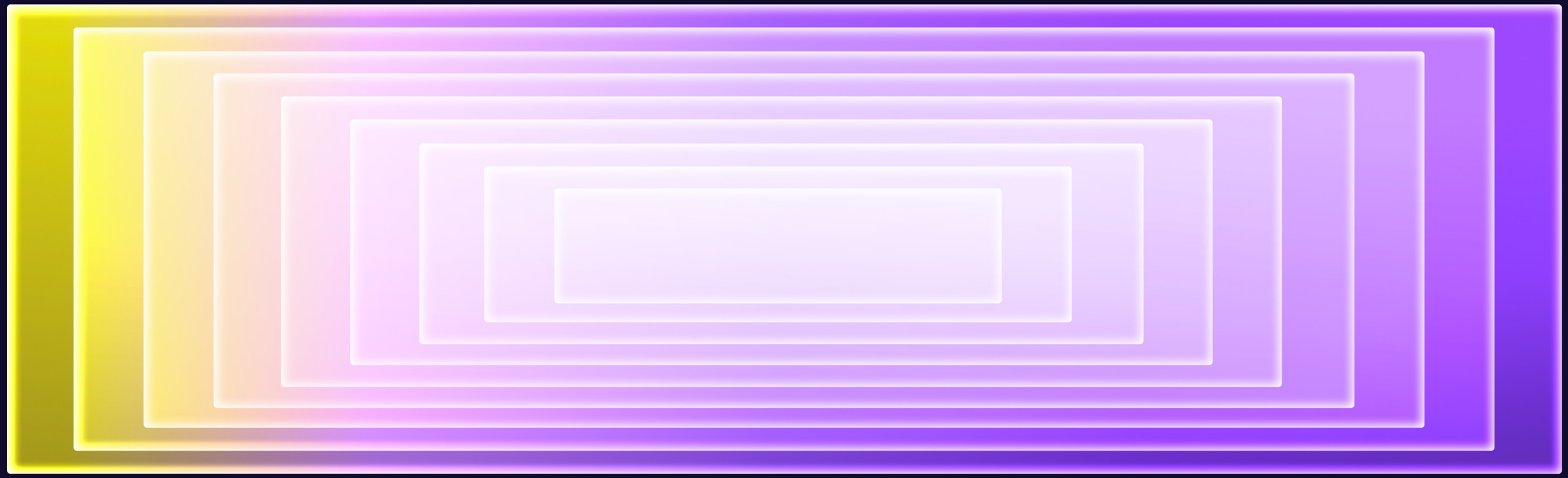
Нет универсального инструмента для отладки jvm байт-кода

javap

Просмотр инструкций байт-кода

JDB

Использовать специальные инструкции вроде stepi и wherei



Отладка автогенерённого кода

Автогенерация в Runtime: основные кейсы

Автогенерация в Runtime: основные кейсы

ORM
и прокси-объектов

AOP

Перехват вызовов,
логирование, транзакции

RPC
и сериализации

Автогенерация в Runtime: основные кейсы

ORM
и прокси-объектов

AOP

Перехват вызовов,
логирование, транзакции

RPC
и сериализации

Мокирование в
тестировании

Автогенерация в Runtime: основные кейсы

ORM
и прокси-объектов

AOP

Перехват вызовов,
логирование, транзакции

RPC
и сериализации

Мокирование в
тестировании

Динамические
языки на JVM

Groovy, Kotlin scripting

Автогенерация в Runtime: основные кейсы

ORM
и прокси-объектов

AOP

Перехват вызовов,
логирование, транзакции

RPC
и сериализации

Мокирование в
тестировании

Динамические
языки на JVM

Groovy, Kotlin scripting

Оптимизации

Спецификация методов
под конкретные данные,
например, в Apache Spark

Основные этапы отладки автогенерённого байт-кода

Сгенерировать класс

Основные этапы отладки автогенерённого байт-кода

Сгенерировать класс

Сериализовать его в массив байт и сохранить class file

Основные этапы отладки автогенерённого байт-кода

Сгенерировать класс

Сериализовать его в массив байт и сохранить class file

Получить листинг байт-код-инструкций через javap -v

Основные этапы отладки автогенерённого байт-кода

- Сгенерировать класс
- Сериализовать его в массив байт и сохранить class file
- Получить листинг байт-код-инструкций через javap -v
- Поставить breakpoint на точку входа

Основные этапы отладки автогенерённого байт-кода

Сгенерировать класс

Сериализовать его в массив байт и сохранить class file

Получить листинг байт-код-инструкций через javap -v

Поставить breakpoint на точку входа

Идти пошагово по инструкциям stepi

Позиция определяется через wherei

Основные этапы отладки автогенерённого байт-кода

Сгенерировать класс

Сериализовать его в массив байт и сохранить class file

Получить листинг байт-код-инструкций через javap -v

Поставить breakpoint на точку входа

Идти пошагово по инструкциям stepi

Позиция определяется через wherei

Если доступна отладочная информация:

можно также смотреть значения локальных переменных и пошагово идти в соответствии с LineNumbersTable

```
1> public interface Entrypoint {
2>     int calculate(int a, int b);
3> }
4>
```

Автогенерация байт-кода на ClassFile API

```
 1> void buildClass(ClassBuilder cb) {
 2>
 3>
 4>
 5>
 6>
 7>
 8>
 9>
10>
11> }
```

Автогенерация байт-кода на ClassFile API

```
 1> void buildClass(ClassBuilder cb) {
 2>   cb.withFlags(ClassFile.ACC_PUBLIC)
 3>     .withInterfaceSymbols(ClassDesc.of(“Entrypoint”))
 4>
 5>
 6>
 7>
 8>
 9>
10>
11> }
```

Автогенерация байт-кода на ClassFile API

```
 1> void buildClass(ClassBuilder cb) {
 2>   cb.withFlags(ClassFile.ACC_PUBLIC)
 3>     .withInterfaceSymbols(ClassDesc.of(“Entrypoint”))
 4>     ...
 5>     .withMethodBody(
 6>       "calculate",
 7>       MethodTypeDesc.of(CD_int, List.of(CD_int,CD_int)),
 8>       ClassFile.ACC_PUBLIC,
 9>       BytecodeDebugging::buildMethodBody)
10>
11> }
```

Автогенерация байт-кода на ClassFile API

```
1> void buildMethodBody(CodeBuilder cb) {
2>     cb.iload(1)
3>       .iload(2)
4>       .imul()
5>       .return_(TypeKind.INT);
6> }
```

Отладка байт-кода при помощи JDB

```
35> Entrypoint instance = generateInstance();
36> int result = instance.calculate(4, 6);
37> IO.println("Result of calculate(4, 6) is " + result);
```

```
> stop at BytecodeDebugging:36
```

```
> stop at BytecodeDebugging:36
  Deferring breakpoint BytecodeDebugging:36.
  It will be set after the class is loaded.
```

```
> stop at BytecodeDebugging:36
  Deferring breakpoint BytecodeDebugging:36.
  It will be set after the class is loaded.
> run
  Breakpoint hit: "thread=main", ...main(), line=36 bci=121
  36              int result = instance.calculate(4, 6);
```

```
> stop at BytecodeDebugging:36
  Deferring breakpoint BytecodeDebugging:36.
  It will be set after the class is loaded.
> run
  Breakpoint hit: "thread=main", ...main(), line=36 bci=121
  36              int result = instance.calculate(4, 6);
> stepi (x4)
```

```
> stop at BytecodeDebugging:36
  Deferring breakpoint BytecodeDebugging:36.
  It will be set after the class is loaded.
> run
  Breakpoint hit: "thread=main", ...main(), line=36 bci=121
  36              int result = instance.calculate(4, 6);
> stepi (x4)
  Step completed: "thread=main", ...calculate(), line=-1 bci=0
```

```
> stop at BytecodeDebugging:36
  Deferring breakpoint BytecodeDebugging:36.
  It will be set after the class is loaded.
> run
  Breakpoint hit: "thread=main", ...main(), line=36 bci=121
  36              int result = instance.calculate(4, 6);
> stepi (x4)
  Step completed: "thread=main", ...calculate(), line=-1 bci=0
> wherei
    [1] SampleCalculator.calculate (null), pc = 0
    [2] BytecodeDebugging.main (BytecodeDebugging.java:36), pc = 126
```

```
> stop at BytecodeDebugging:36
  Deferring breakpoint BytecodeDebugging:36.
  It will be set after the class is loaded.
> run
  Breakpoint hit: "thread=main", ...main(), line=36 bci=121
  36              int result = instance.calculate(4, 6);
> stepi (x4)
  Step completed: "thread=main", ...calculate(), line=-1 bci=0
> wherei
    [1] SampleCalculator.calculate (null), pc = 0
    [2] BytecodeDebugging.main (BytecodeDebugging.java:36), pc = 126
> stepi
  Step completed: "thread=main", ...calculate(), line=-1 bci=1
```

```
> stop at BytecodeDebugging:36
  Deferring breakpoint BytecodeDebugging:36.
  It will be set after the class is loaded.
> run
  Breakpoint hit: "thread=main", ...main(), line=36 bci=121
  36              int result = instance.calculate(4, 6);
> stepi (x4)
  Step completed: "thread=main", ...calculate(), line=-1 bci=0
> wherei
    [1] SampleCalculator.calculate (null), pc = 0
    [2] BytecodeDebugging.main (BytecodeDebugging.java:36), pc = 126
> stepi
  Step completed: "thread=main", ...calculate(), line=-1 bci=1
> locals
  Local variable information not available.  Compile with -g to
  generate variable information
```

Автогенерация байт-кода на ClassFile API с отладочными символами

```
1> void buildMethodBody(CodeBuilder cb) {
2>      cb.iload(1)
3>        .iload(2)
4>        .imul()
5>        .return_(TypeKind.INT);
6> }
```

Автогенерация байт-кода на ClassFile API с отладочными символами

```
 1> void buildMethodBody(CodeBuilder cb) {
 2>      cb
 3>
 4>
 5>
 6>
 7>
 8>
 9>        .iload(1)
10>        .iload(2)
11>
12>        .imul()
13>        .return_(TypeKind.INT);
14> }
```

Автогенерация байт-кода на ClassFile API с отладочными символами

```
 1> void buildMethodBody(CodeBuilder cb) {
 2>     cb
 3>
 4>       .localVariable(1, "a", CD_int, cb.startLabel(),
 5>                      cb.endLabel())
 6>       .localVariable(2, "b", CD_int, cb.startLabel(),
 7>                      cb.endLabel())
 8>
 9>       .iload(1)
10>       .iload(2)
11>
12>       .imul()
13>       .return_(TypeKind.INT);
14> }
```

Автогенерация байт-кода на ClassFile API с отладочными символами

```
 1> void buildMethodBody(CodeBuilder cb) {
 2>     cb.localVariable(0, "this", ClassDesc.of(...),
 3>                      cb.startLabel(), cb.endLabel())
 4>       .localVariable(1, "a", CD_int, cb.startLabel(),
 5>                      cb.endLabel())
 6>       .localVariable(2, "b", CD_int, cb.startLabel(),
 7>                      cb.endLabel())
 8>
 9>       .iload(1)
10>       .iload(2)
11>
12>       .imul()
13>       .return_(TypeKind.INT);
14> }
```

Автогенерация байт-кода на ClassFile API с отладочными символами

```
 1> void buildMethodBody(CodeBuilder cb) {
 2>     cb.localVariable(0, "this", ClassDesc.of(...),
 3>                      cb.startLabel(), cb.endLabel())
 4>       .localVariable(1, "a", CD_int, cb.startLabel(),
 5>                      cb.endLabel())
 6>       .localVariable(2, "b", CD_int, cb.startLabel(),
 7>                      cb.endLabel())
 8>       .lineNumber(1)
 9>       .iload(1)
10>       .iload(2)
11>       .lineNumber(2)
12>       .imul()
13>       .return_(TypeKind.INT);
14> }
```

Автогенерация байт-кода на ClassFile API

```
 1> void buildClass(ClassBuilder cb) {
 2>   cb.withFlags(ClassFile.ACC_PUBLIC)
 3>     .withInterfaceSymbols(ClassDesc.of(“Entrypoint”))
 4>     ...
 5>     .withMethodBody(
 6>       "calculate",
 7>       MethodTypeDesc.of(CD_int, List.of(CD_int,CD_int)),
 8>       ClassFile.ACC_PUBLIC,
 9>       BytecodeDebugging::buildMethodBody)
10>
11> }
```

Автогенерация байт-кода на ClassFile API

```
 1> void buildClass(ClassBuilder cb) {
 2>   cb.withFlags(ClassFile.ACC_PUBLIC)
 3>     .withInterfaceSymbols(ClassDesc.of(“Entrypoint”))
 4>     ...
 5>     .withMethodBody(
 6>       "calculate",
 7>       MethodTypeDesc.of(CD_int, List.of(CD_int,CD_int)),
 8>       ClassFile.ACC_PUBLIC,
 9>       BytecodeDebugging::buildMethodBody)
10>     .with(SourceFileAttribute.of("expression.expr"));
11> }
```

JDB: запуск с указанием исходников

https://github.com/alextokarew/java-debugging

```
./gradlew build
```

```
jdb -sourcepath app/src/main/java:app/src/main/exprs \
    -classpath app/build/classes/java/main \
    com.github.alextokarew.javadebugging.codegen.BytecodeDebugging
```

JDB: запуск с указанием исходников

https://github.com/alextokarew/java-debugging

```
./gradlew build
```

```
jdb -sourcepath app/src/main/java:app/src/main/exprs \
    -classpath app/build/classes/java/main \
    com.github.alextokarew.javadebugging.codegen.BytecodeDebugging
```

Псевдоисходник:
.../exprs/com/github/alextokarew/javadebugging/codegen/expression.expr

```
> stop at BytecodeDebugging:36
  Deferring breakpoint BytecodeDebugging:36.
  It will be set after the class is loaded.
> run
  Breakpoint hit: "thread=main", ...main(), line=36 bci=121
  36              int result = instance.calculate(4, 6);
```

```
> stop at BytecodeDebugging:36
  Deferring breakpoint BytecodeDebugging:36.
  It will be set after the class is loaded.
> run
  Breakpoint hit: "thread=main", ...main(), line=36 bci=121
  36              int result = instance.calculate(4, 6);
> step
  Step completed: "thread=main", ...calculate(), line=1 bci=0
  1    6 4
```

```
> stop at BytecodeDebugging:36
  Deferring breakpoint BytecodeDebugging:36.
  It will be set after the class is loaded.
> run
  Breakpoint hit: "thread=main", ...main(), line=36 bci=121
  36              int result = instance.calculate(4, 6);
> step
  Step completed: "thread=main", ...calculate(), line=1 bci=0
  1    6 4
> list
  1 => 6 4
  2    *
```

```
> stop at BytecodeDebugging:36
  Deferring breakpoint BytecodeDebugging:36.
  It will be set after the class is loaded.
> run
  Breakpoint hit: "thread=main", ...main(), line=36 bci=121
  36              int result = instance.calculate(4, 6);
> step
  Step completed: "thread=main", ...calculate(), line=1 bci=0
  1    6 4
> list
  1 => 6 4
  2    *
> where
    [1] SampleCalculator.calculate (expression.expr:1)
    [2] BytecodeDebugging.main (BytecodeDebugging.java:36)
```

```
> stop at BytecodeDebugging:36
  Deferring breakpoint BytecodeDebugging:36.
  It will be set after the class is loaded.
> run
  Breakpoint hit: "thread=main", ...main(), line=36 bci=121
  36              int result = instance.calculate(4, 6);
> step
  Step completed: "thread=main", ...calculate(), line=1 bci=0
  1    6 4
> list
  1 => 6 4
  2    *
> where
    [1] SampleCalculator.calculate (expression.expr:1)
    [2] BytecodeDebugging.main (BytecodeDebugging.java:36)
> step
  Step completed: "thread=main", ...calculate(), line=2 bci=2
  2    *
```

```
> run
  Breakpoint hit: "thread=main", ...main(), line=36 bci=121
  36            int result = instance.calculate(4, 6);
> step
  Step completed: "thread=main", ...calculate(), line=1 bci=0
  1    6 4
> list
  1 => 6 4
  2    *
> where
    [1] SampleCalculator.calculate (expression.expr:1)
    [2] BytecodeDebugging.main (BytecodeDebugging.java:36)
> step
  Step completed: "thread=main", ...calculate(), line=2 bci=2
  2    *
> locals
  Method arguments:
  a = 4
  b = 6
```

Отладка автогенерённого байт-кода: выводы

Отладка автогенерённого байт-кода: выводы

</>

Для удобства отладки
добавляйте отладочную
информацию
под feature flag

Отладка автогенерённого байт-кода: выводы

</>

Для удобства отладки добавляйте отладочную информацию под feature flag

{ƒ}

this — тоже локальная переменная, которая всегда находится в нулевом слоте

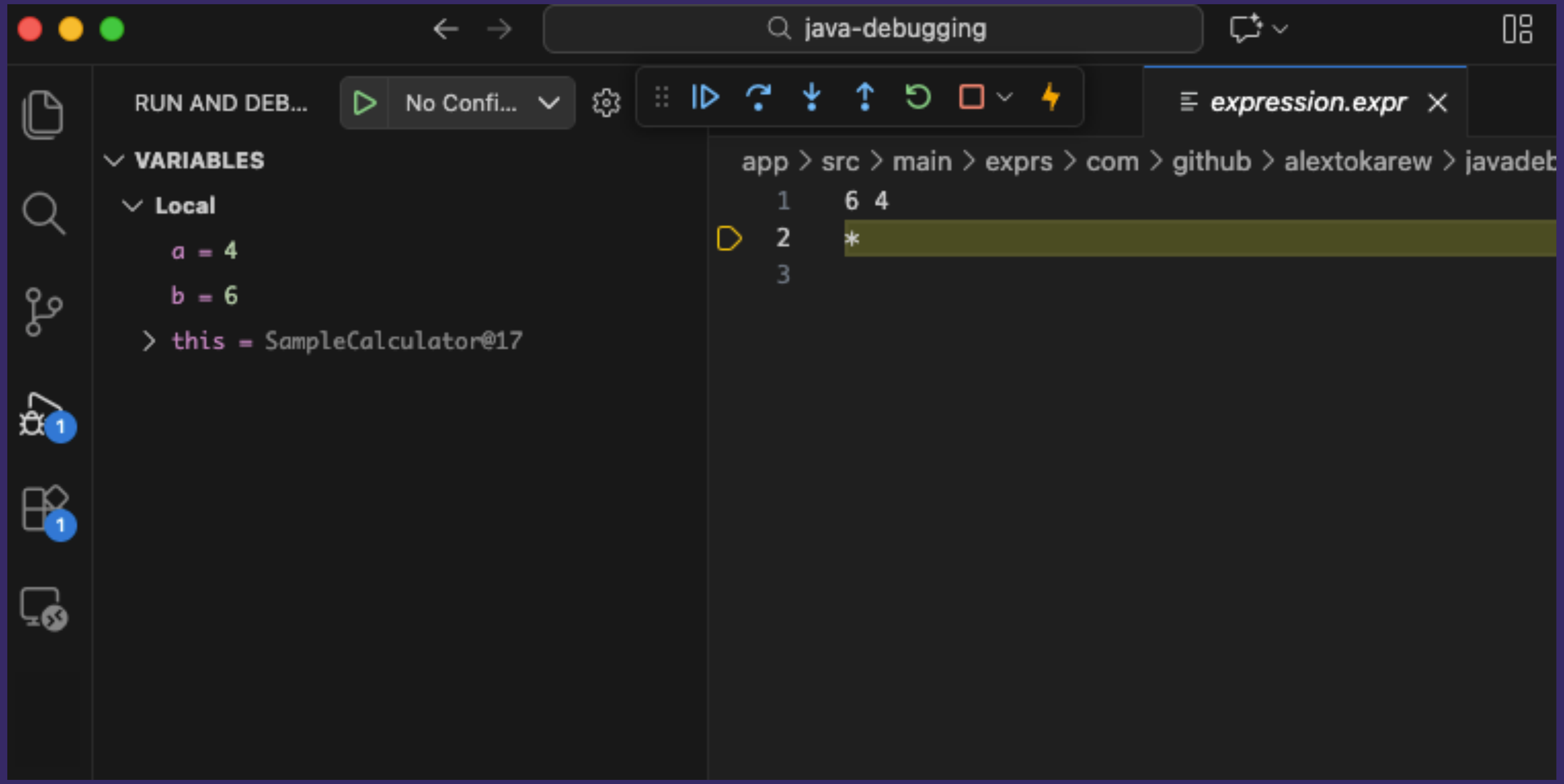
Отладка автогенерённого байт-кода: выводы

Для удобства отладки добавляйте отладочную информацию под feature flag

this — тоже локальная переменная, которая всегда находится в нулевом слоте

Package name — matters!

- JDB увидел псевдоисходник и показал нужные строки
- В VSCode и Eclipse получилось зайти дебаггером внутрь автогенерённого исходника, в Intellij IDEA — нет


java-debugging
RUN AND DEB...
No Confi...
expression.expr
VARIABLES
Local
a = 4
b = 6
this = SampleCalculator@17
app > src > main > exprs > com > github > alextokarew > javadeb
1 6 4
2 *
3

Основные проблемы при отладке на уровне байт-кода

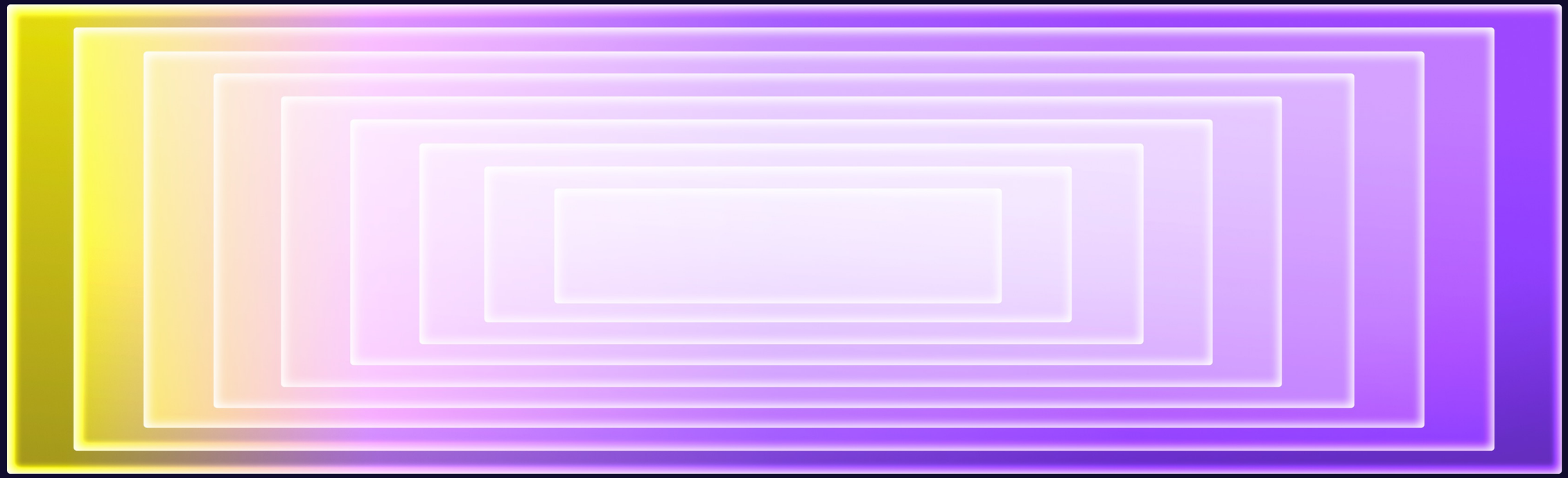
Основные проблемы при отладке на уровне байт-кода

Нет возможности смотреть значения локальных переменных в случае отсутствия раздела LocalVariableTable

Основные проблемы при отладке на уровне байт-кода

Нет возможности смотреть значения локальных переменных в случае отсутствия раздела LocalVariableTable

JDI не поддерживает инспекцию операндного стека даже при наличии полной отладочной информации



Отладка многопоточных приложений

Основные проблемы при отладке многопоточных приложений

Основные проблемы при отладке многопоточных приложений

Трудновоспроизводимые ошибки

Основные проблемы при отладке многопоточных приложений

Трудновоспроизводимые ошибки

Дебаггинг может помочь понять причину возникновения гонок и дедлоков

Основные проблемы при отладке многопоточных приложений

Трудновоспроизводимые ошибки

Дебаггинг может помочь понять причину возникновения гонок и дедлоков

Для их детектирования лучше воспользоваться другими инструментами

- jcmd Thread.print (aka jstack)
- jcstress
- Java PathFinder

```
 6> private int counter = 0;
 7>
 8> private void run() {
 9>   try(var pool = Executors.newFixedThreadPool(2)) {
10>     for (int i = 0; i < 2; i++) {
11>       pool.submit(() -> {
12>         counter++;
13>       });
14>     }
15>   }
16> }
17>
18> private void main() {
19>   var races = new Races();
20>   races.run();
21>   IO.println("Counter: Expected 2, actual " +  races.counter);
22> }
```

```
 6> private int counter = 0;
 7>
 8> private void run() {
 9>   try(var pool = Executors.newFixedThreadPool(2)) {
10>     for (int i = 0; i < 2; i++) {
11>       pool.submit(() -> {
12>         counter++;
13>       });
14>     }
15>   }
16> }
17>
18> private void main() {
19>   var races = new Races();
20>   races.run();
21>   IO.println("Counter: Expected 2, actual " +  races.counter);
22> }
```

JDB and Races

```
> stop thread at Races:12
> run
Breakpoint hit: "thread=pool-1-thread-1”
Breakpoint hit: "thread=pool-1-thread-2"
```

JDB and Races

```
> stop thread at Races:12
> run
Breakpoint hit: "thread=pool-1-thread-1”
Breakpoint hit: "thread=pool-1-thread-2"
> threads
...
Group main:
  (java.lang.Thread)1     main                cond. waiting
  (java.lang.Thread)643   pool-1-thread-1     running (at breakpoint)
  (java.lang.Thread)645   pool-1-thread-2     running (at breakpoint)
```

JDB and Races

```
> stop thread at Races:12
> run
Breakpoint hit: "thread=pool-1-thread-1”
Breakpoint hit: "thread=pool-1-thread-2"
> threads
...
Group main:
  (java.lang.Thread)1     main                cond. waiting
  (java.lang.Thread)643   pool-1-thread-1     running (at breakpoint)
  (java.lang.Thread)645   pool-1-thread-2     running (at breakpoint)
> suspend 645
```

JDB and Races

```
> stop thread at Races:12
> run
Breakpoint hit: "thread=pool-1-thread-1”
Breakpoint hit: "thread=pool-1-thread-2"
> threads
...
Group main:
  (java.lang.Thread)1     main                cond. waiting
  (java.lang.Thread)643   pool-1-thread-1     running (at breakpoint)
  (java.lang.Thread)645   pool-1-thread-2     running (at breakpoint)
> suspend 645
> thread 643
```

JDB and Races

```
private void lambda$run$0();
  descriptor: ()V
  flags: (0x1002) ACC_PRIVATE, ACC_SYNTHETIC
  Code:
    stack=3, locals=1, args_size=1
       0: aload_0
       1: dup
       2: getfield      #7                  // Field counter:I
       5: iconst_1
       6: iadd
       7: putfield      #7                  // Field counter:I
      10: return
    LineNumberTable:
      line 12: 0
      line 13: 10
    LocalVariableTable:
      Start  Length  Slot  Name   Signature
          0      11     0  this
Lcom/github/alextokarew/javadebugging/multithreaded/Races;
```

JDB and Races

```
private void lambda$run$0();
  descriptor: ()V
  flags: (0x1002) ACC_PRIVATE, ACC_SYNTHETIC
  Code:
    stack=3, locals=1, args_size=1
       0: aload_0
       1: dup
       2: getfield      #7                  // Field counter:I
       5: iconst_1
       6: iadd
       7: putfield      #7                  // Field counter:I
      10: return
    LineNumberTable:
      line 12: 0
      line 13: 10
    LocalVariableTable:
      Start  Length  Slot  Name   Signature
          0      11     0  this
Lcom/github/alextokarew/javadebugging/multithreaded/Races;
```

JDB and Races

```
private void lambda$run$0();
  descriptor: ()V
  flags: (0x1002) ACC_PRIVATE, ACC_SYNTHETIC
  Code:
    stack=3, locals=1, args_size=1
       0: aload_0
       1: dup
       2: getfield      #7                  // Field counter:I
       5: iconst_1
       6: iadd
       7: putfield      #7                  // Field counter:I
      10: return
    LineNumberTable:
      line 12: 0
      line 13: 10
    LocalVariableTable:
      Start  Length  Slot  Name   Signature
          0      11     0  this
Lcom/github/alextokarew/javadebugging/multithreaded/Races;
```

JDB and Races

```
> stop thread at Races:12
> run
Breakpoint hit: "thread=pool-1-thread-1”
Breakpoint hit: "thread=pool-1-thread-2"
> threads
...
Group main:
  (java.lang.Thread)1     main                 cond. waiting
  (java.lang.Thread)643   pool-1-thread-1      running (at breakpoint)
  (java.lang.Thread)645   pool-1-thread-2      running (at breakpoint)
> suspend 645
> thread 643
pool-1-thread-1[1] stepi (x3)
Step completed: "thread=pool-1-thread-1", ...$run$0(), line=12 bci=5
```

JDB and Races

```
Breakpoint hit: "thread=pool-1-thread-1”
Breakpoint hit: "thread=pool-1-thread-2"
> threads
...
Group main:
  (java.lang.Thread)1     main                 cond. waiting
  (java.lang.Thread)643   pool-1-thread-1      running (at breakpoint)
  (java.lang.Thread)645   pool-1-thread-2      running (at breakpoint)
> suspend 645
> thread 643
pool-1-thread-1[1] stepi (x3)
Step completed: "thread=pool-1-thread-1", ...$run$0(), line=12 bci=5
pool-1-thread-1[1] print counter
conuter = 0
```

JDB and Races

```
...
Group main:
  (java.lang.Thread)1     main                cond. waiting
  (java.lang.Thread)643   pool-1-thread-1     running (at breakpoint)
  (java.lang.Thread)645   pool-1-thread-2     running (at breakpoint)
> suspend 645
> thread 643
pool-1-thread-1[1] stepi (x3)
Step completed: "thread=pool-1-thread-1", ...$run$0(), line=12 bci=5
pool-1-thread-1[1] print counter
conuter = 0
pool-1-thread-1[1] suspend 643
pool-1-thread-1[1] thread 645
pool-1-thread-2[1] resume 645
```

JDB and Races

```
  (java.lang.Thread)1     main                cond. waiting
  (java.lang.Thread)643   pool-1-thread-1     running (at breakpoint)
  (java.lang.Thread)645   pool-1-thread-2     running (at breakpoint)
> suspend 645
> thread 643
pool-1-thread-1[1] stepi (x3)
Step completed: "thread=pool-1-thread-1", ...$run$0(), line=12 bci=5
pool-1-thread-1[1] print counter
conuter = 0
pool-1-thread-1[1] suspend 643
pool-1-thread-1[1] thread 645
pool-1-thread-2[1] resume 645
pool-1-thread-2[1] stepi (x3)
Step completed: "thread=pool-1-thread-1", ...$run$0(), line=12 bci=5
```

JDB and Races

```
  (java.lang.Thread)645   pool-1-thread-2     running (at breakpoint)
> suspend 645
> thread 643
pool-1-thread-1[1] stepi (x3)
Step completed: "thread=pool-1-thread-1", ...$run$0(), line=12 bci=5
pool-1-thread-1[1] print counter
conuter = 0
pool-1-thread-1[1] suspend 643
pool-1-thread-1[1] thread 645
pool-1-thread-2[1] resume 645
pool-1-thread-2[1] stepi (x3)
Step completed: "thread=pool-1-thread-1", ...$run$0(), line=12 bci=5
pool-1-thread-2[1] print counter
conuter = 0
```

JDB and Races

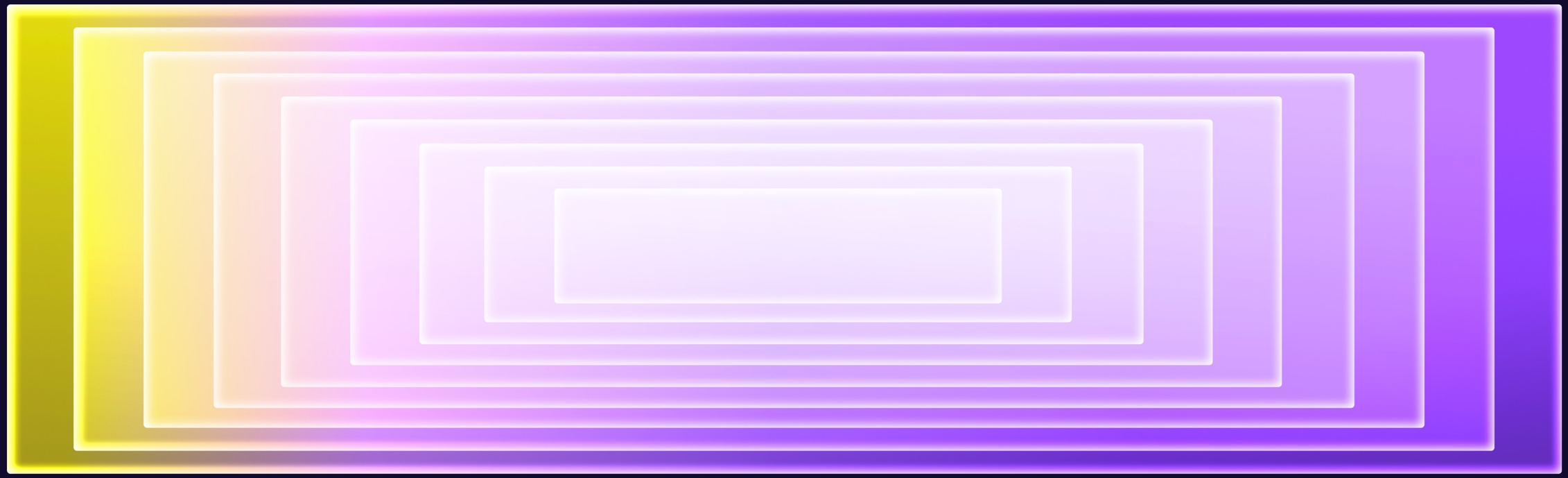
```
pool-1-thread-1[1] stepi (x3)
Step completed: "thread=pool-1-thread-1", ...$run$0(), line=12 bci=5
pool-1-thread-1[1] print counter
conuter = 0
pool-1-thread-1[1] suspend 643
pool-1-thread-1[1] thread 645
pool-1-thread-2[1] resume 645
pool-1-thread-2[1] stepi (x3)
Step completed: "thread=pool-1-thread-1", ...$run$0(), line=12 bci=5
pool-1-thread-2[1] print counter
conuter = 0
pool-1-thread-2[1] resume 643
pool-1-thread-2[1] cont
```

JDB and Races

```
private void lambda$run$0();
  descriptor: ()V
  flags: (0x1002) ACC_PRIVATE, ACC_SYNTHETIC
  Code:
    stack=3, locals=1, args_size=1
       0: aload_0
       1: dup
       2: getfield      #7                  // Field counter:I
       5: iconst_1
       6: iadd
       7: putfield      #7                  // Field counter:I
      10: return
    LineNumberTable:
      line 12: 0
      line 13: 10
    LocalVariableTable:
      Start  Length  Slot  Name   Signature
          0      11     0  this
Lcom/github/alextokarew/javadebugging/multithreaded/Races;
```

JDB and Races

```
pool-1-thread-1[1] stepi (x3)
Step completed: "thread=pool-1-thread-1", ...$run$0(), line=12 bci=5
pool-1-thread-1[1] print counter
conuter = 0
pool-1-thread-1[1] suspend 643
pool-1-thread-1[1] thread 645
pool-1-thread-2[1] resume 645
pool-1-thread-2[1] stepi (x3)
Step completed: "thread=pool-1-thread-1", ...$run$0(), line=12 bci=5
pool-1-thread-2[1] print counter
conuter = 0
pool-1-thread-2[1] resume 643
pool-1-thread-2[1] cont
Counter: Expected 2, actual 1
```



Async Stack Traces

Async Stack Traces в Intellij IDEA

Позволяет установить
связь между
отлаживаемым потоком
и потоком, запустившим
асинхронное вычисление

Async Stack Traces в Intellij IDEA

Позволяет установить связь между отлаживаемым потоком и потоком, запустившим асинхронное вычисление

В случае каскадного запуска задач позволяет отследить всю цепочку асинхронных вызовов

```
12>  private static CompletableFuture<User> getUserByName(String name) {
13>    return CompletableFuture.supplyAsync(() -> {
15>      try { Thread.sleep(800); } catch (InterruptedException ignored) {}
16>      return new User(1, name);
17>    });
18>  }
```

```
12>  private static CompletableFuture<User> getUserByName(String name) {
13>    return CompletableFuture.supplyAsync(() -> {
15>      try { Thread.sleep(800); } catch (InterruptedException ignored) {}
16>      return new User(1, name);
17>    });
18>  }
19>
20>  private static CompletableFuture<List<Order>> getOrders(int userId) {
21>    return CompletableFuture.supplyAsync(() -> {
23>      try { Thread.sleep(1000); } catch (InterruptedException ignored) {}
24>      if (userId == 1) throw new RuntimeException("BOOM");
25>      return Arrays.asList(new Order(101, userId, "Book"),
27>                           new Order(102, userId, "Laptop"));
29>    });
30>  }
```

```
12>  private static CompletableFuture<User> getUserByName(String name) {
13>    return CompletableFuture.supplyAsync(() -> {
15>      try { Thread.sleep(800); } catch (InterruptedException ignored) {}
16>      return new User(1, name);
17>    });
18>  }
19>
20>  private static CompletableFuture<List<Order>> getOrders(int userId) {
21>    return CompletableFuture.supplyAsync(() -> {
23>      try { Thread.sleep(1000); } catch (InterruptedException ignored) {}
24>      if (userId == 1) throw new RuntimeException("BOOM");
25>      return Arrays.asList(new Order(101, userId, "Book"),
27>                           new Order(102, userId, "Laptop"));
29>    });
30>  }
31>
32>  static void main(String[] args) throws ExecutionException, InterruptedException {
33>    CompletableFuture<User> userFuture = getUserByName(args[0]);
34>    CompletableFuture<List<Order>> ordersFuture = userFuture.thenCompose(
        user -> getOrders(user.id));
35>    IO.println("Orders for user: " + ordersFuture.get());
36>  }
```

```
Exception in thread "main" java.util.concurrent.ExecutionException:
java.lang.RuntimeException: BOOM
    at CompletableFuture.wrapInExecutionException(CompletableFuture.java:345)
    at CompletableFuture.reportGet(CompletableFuture.java:440)
    at CompletableFuture.get(CompletableFuture.java:2094)
    at AsyncExample.main(AsyncExample.java:35)
Caused by: java.lang.RuntimeException: BOOM
    at AsyncExample.lambda$getOrders$0(AsyncExample.java:24)
    at CompletableFuture$AsyncSupply.run(CompletableFuture.java:1789)
    at CompletableFuture$AsyncSupply.exec(CompletableFuture.java:1781)
    at ForkJoinTask.doExec(ForkJoinTask.java:511)
    at ForkJoinPool$WorkQueue.topLevelExec(ForkJoinPool.java:1450)
    at ForkJoinPool.runWorker(ForkJoinPool.java:2019)
    at ForkJoinWorkerThread.run(ForkJoinWorkerThread.java:187)
```

```
Exception in thread "main" java.util.concurrent.ExecutionException:
java.lang.RuntimeException: BOOM
    at CompletableFuture.wrapInExecutionException(CompletableFuture.java:345)
    at CompletableFuture.reportGet(CompletableFuture.java:440)
    at CompletableFuture.get(CompletableFuture.java:2094)
    at AsyncExample.main(AsyncExample.java:35)
Caused by: java.lang.RuntimeException: BOOM
    at AsyncExample.lambda$getOrders$0(AsyncExample.java:24)
    at CompletableFuture$AsyncSupply.run(CompletableFuture.java:1789)
    at CompletableFuture$AsyncSupply.exec(CompletableFuture.java:1781)
    at ForkJoinTask.doExec(ForkJoinTask.java:511)
    at ForkJoinPool$WorkQueue.topLevelExec(ForkJoinPool.java:1450)
    at ForkJoinPool.runWorker(ForkJoinPool.java:2019)
    at ForkJoinWorkerThread.run(ForkJoinWorkerThread.java:187)
```

```
12>  private static CompletableFuture<User> getUserByName(String name) {
13>    return CompletableFuture.supplyAsync(() -> {
15>      try { Thread.sleep(800); } catch (InterruptedException ignored) {}
16>      return new User(1, name);
17>    });
18>  }
19>
20>  private static CompletableFuture<List<Order>> getOrders(int userId) {
21>    return CompletableFuture.supplyAsync(() -> {
23>      try { Thread.sleep(1000); } catch (InterruptedException ignored) {}
24>      if (userId == 1) throw new RuntimeException("BOOM");
25>      return Arrays.asList(new Order(101, userId, "Book"),
27>                           new Order(102, userId, "Laptop"));
29>    });
30>  }
31>
32>  static void main(String[] args) throws ExecutionException, InterruptedException {
33>    CompletableFuture<User> userFuture = getUserByName(args[0]);
34>    CompletableFuture<List<Order>> ordersFuture = userFuture.thenCompose(
        user -> getOrders(user.id));
35>    IO.println("Orders for user: " + ordersFuture.get());
36>  }
```

```
Exception in thread "main" java.util.concurrent.ExecutionException:
java.lang.RuntimeException: BOOM
    at ...
    at AsyncExample.main(AsyncExample.java:35)
Caused by: java.lang.RuntimeException: BOOM
    at AsyncExample.lambda$getOrders$0(AsyncExample.java:24)
    at CompletableFuture$AsyncSupply.run$$$capture(CompletableFuture.java:1789)
    at CompletableFuture$AsyncSupply.run(CompletableFuture.java)
    at --- Async.Stack.Trace --- (captured by IntelliJ IDEA debugger)
    at CompletableFuture$AsyncSupply.<init>(CompletableFuture.java:1775)
    at ...
    at AsyncExample.getOrders(AsyncExample.java:21)
    at AsyncExample.lambda$main$0(AsyncExample.java:34)
    at CompletableFuture$UniCompose.tryFire(CompletableFuture.java:1171)
    at ...
    at CompletableFuture$AsyncSupply.run(CompletableFuture.java)
    at --- Async.Stack.Trace --- (captured by IntelliJ IDEA debugger)
    at AsyncSupply.<init>(CompletableFuture.java:1775)
    at ...
    at AsyncExample.getUserByName(AsyncExample.java:13)
    at AsyncExample.main(AsyncExample.java:33)
```

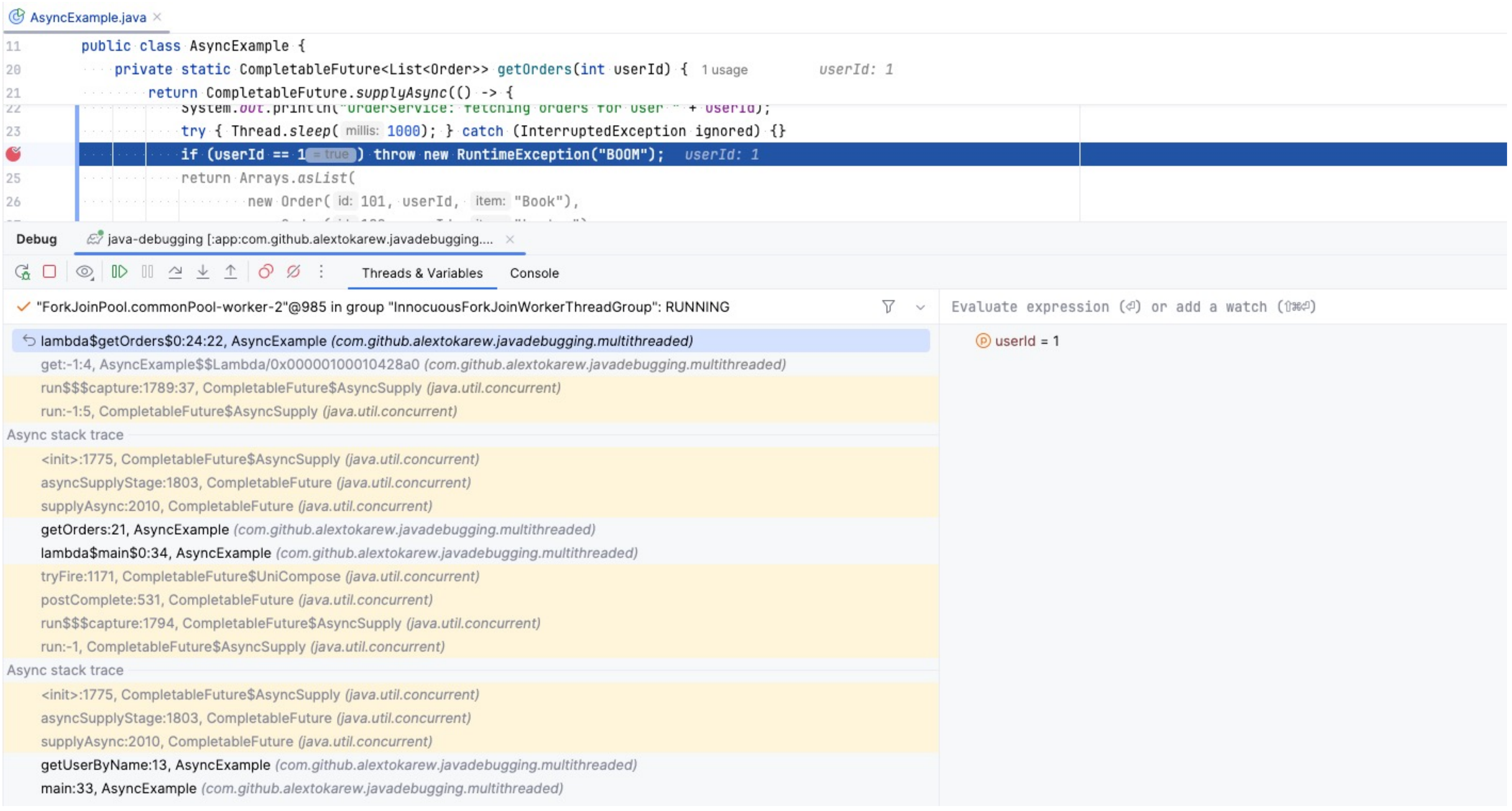
```
Exception in thread "main" java.util.concurrent.ExecutionException:
java.lang.RuntimeException: BOOM
    at ...
    at AsyncExample.main(AsyncExample.java:35)
Caused by: java.lang.RuntimeException: BOOM
    at AsyncExample.lambda$getOrders$0(AsyncExample.java:24)
    at CompletableFuture$AsyncSupply.run$$$capture(CompletableFuture.java:1789)
    at CompletableFuture$AsyncSupply.run(CompletableFuture.java)
    at --- Async.Stack.Trace --- (captured by IntelliJ IDEA debugger)
    at CompletableFuture$AsyncSupply.<init>(CompletableFuture.java:1775)
    at ...
    at AsyncExample.getOrders(AsyncExample.java:21)
    at AsyncExample.lambda$main$0(AsyncExample.java:34)
    at CompletableFuture$UniCompose.tryFire(CompletableFuture.java:1171)
    at ...
    at CompletableFuture$AsyncSupply.run(CompletableFuture.java)
    at --- Async.Stack.Trace --- (captured by IntelliJ IDEA debugger)
    at AsyncSupply.<init>(CompletableFuture.java:1775)
    at ...
    at AsyncExample.getUserByName(AsyncExample.java:13)
    at AsyncExample.main(AsyncExample.java:33)
```

```
Exception in thread "main" java.util.concurrent.ExecutionException:
java.lang.RuntimeException: BOOM
    at ...
    at AsyncExample.main(AsyncExample.java:35)
Caused by: java.lang.RuntimeException: BOOM
    at AsyncExample.lambda$getOrders$0(AsyncExample.java:24)
    at CompletableFuture$AsyncSupply.run$$$capture(CompletableFuture.java:1789)
    at CompletableFuture$AsyncSupply.run(CompletableFuture.java)
    at --- Async.Stack.Trace --- (captured by IntelliJ IDEA debugger)
    at CompletableFuture$AsyncSupply.<init>(CompletableFuture.java:1775)
    at ...
    at AsyncExample.getOrders(AsyncExample.java:21)
    at AsyncExample.lambda$main$0(AsyncExample.java:34)
    at CompletableFuture$UniCompose.tryFire(CompletableFuture.java:1171)
    at ...
    at CompletableFuture$AsyncSupply.run(CompletableFuture.java)
    at --- Async.Stack.Trace --- (captured by IntelliJ IDEA debugger)
    at AsyncSupply.<init>(CompletableFuture.java:1775)
    at ...
    at AsyncExample.getUserByName(AsyncExample.java:13)
    at AsyncExample.main(AsyncExample.java:33)
```

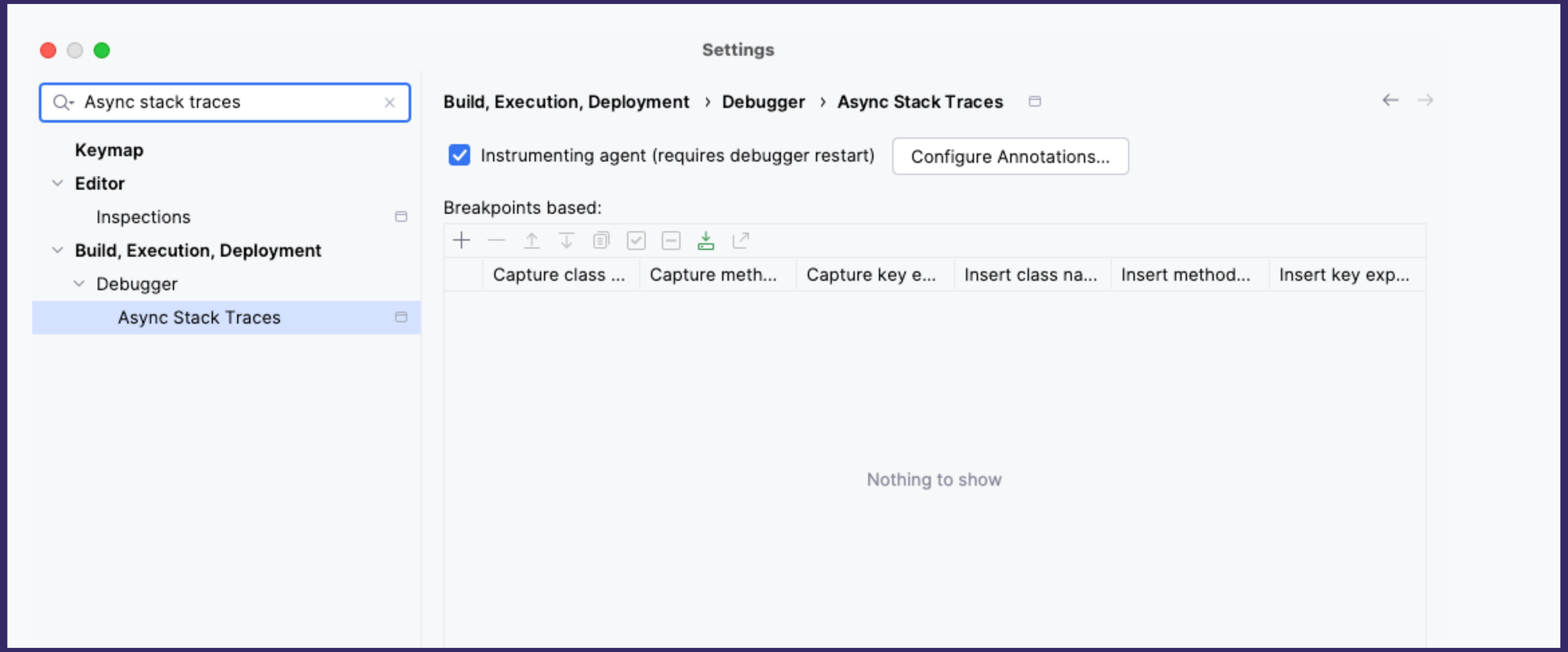
```
12>  private static CompletableFuture<User> getUserByName(String name) {
13>    return CompletableFuture.supplyAsync(() -> {
15>      try { Thread.sleep(800); } catch (InterruptedException ignored) {}
16>      return new User(1, name);
17>    });
18>  }
19>
20>  private static CompletableFuture<List<Order>> getOrders(int userId) {
21>    return CompletableFuture.supplyAsync(() -> {
23>      try { Thread.sleep(1000); } catch (InterruptedException ignored) {}
24>      if (userId == 1) throw new RuntimeException("BOOM");
25>      return Arrays.asList(new Order(101, userId, "Book"),
27>                           new Order(102, userId, "Laptop"));
29>    });
30>  }
31>
32>  static void main(String[] args) throws ExecutionException, InterruptedException {
33>    CompletableFuture<User> userFuture = getUserByName(args[0]);
34>    CompletableFuture<List<Order>> ordersFuture = userFuture.thenCompose(
        user -> getOrders(user.id));
35>    IO.println("Orders for user: " + ordersFuture.get());
36>  }
```

```
Exception in thread "main" java.util.concurrent.ExecutionException:
java.lang.RuntimeException: BOOM
    at ...
    at AsyncExample.main(AsyncExample.java:35)
Caused by: java.lang.RuntimeException: BOOM
    at AsyncExample.lambda$getOrders$0(AsyncExample.java:24)
    at CompletableFuture$AsyncSupply.run$$$capture(CompletableFuture.java:1789)
    at CompletableFuture$AsyncSupply.run(CompletableFuture.java)
    at --- Async.Stack.Trace --- (captured by IntelliJ IDEA debugger)
    at CompletableFuture$AsyncSupply.<init>(CompletableFuture.java:1775)
    at ...
    at AsyncExample.getOrders(AsyncExample.java:21)
    at AsyncExample.lambda$main$0(AsyncExample.java:34)
    at CompletableFuture$UniCompose.tryFire(CompletableFuture.java:1171)
    at ...
    at CompletableFuture$AsyncSupply.run(CompletableFuture.java)
    at --- Async.Stack.Trace --- (captured by IntelliJ IDEA debugger)
    at AsyncSupply.<init>(CompletableFuture.java:1775)
    at ...
    at AsyncExample.getUserByName(AsyncExample.java:13)
    at AsyncExample.main(AsyncExample.java:33)
```

```
12>  private static CompletableFuture<User> getUserByName(String name) {
13>    return CompletableFuture.supplyAsync(() -> {
15>      try { Thread.sleep(800); } catch (InterruptedException ignored) {}
16>      return new User(1, name);
17>    });
18>  }
19>
20>  private static CompletableFuture<List<Order>> getOrders(int userId) {
21>    return CompletableFuture.supplyAsync(() -> {
23>      try { Thread.sleep(1000); } catch (InterruptedException ignored) {}
24>      if (userId == 1) throw new RuntimeException("BOOM");
25>      return Arrays.asList(new Order(101, userId, "Book"),
27>                           new Order(102, userId, "Laptop"));
29>    });
30>  }
31>
32>  static void main(String[] args) throws ExecutionException, InterruptedException {
33>    CompletableFuture<User> userFuture = getUserByName(args[0]);
34>    CompletableFuture<List<Order>> ordersFuture = userFuture.thenCompose(
        user -> getOrders(user.id));
35>    IO.println("Orders for user: " + ordersFuture.get());
36>  }
```


AsyncExample.java
public class AsyncExample {
private static CompletableFuture<List<Order>> getOrders(int userId) { 1 usage userId: 1
return CompletableFuture.supplyAsync(() -> {
try { Thread.sleep(millis: 1000); } catch (InterruptedException ignored) {}
if (userId == 1 = true) throw new RuntimeException("BOOM"); userId: 1
return Arrays.asList(
new Order(id: 101, userId, item: "Book"),
Debug
java-debugging [:app:com.github.alextokarew.javadebugging....
Threads & Variables
Console
"ForkJoinPool.commonPool-worker-2"@985 in group "InnocuousForkJoinWorkerThreadGroup": RUNNING
Evaluate expression (⏎) or add a watch (⇧⌘⏎)
userId = 1
lambda$getOrders$0:24:22, AsyncExample (com.github.alextokarew.javadebugging.multithreaded)
get:-1:4, AsyncExample$$Lambda/0x00000100010428a0 (com.github.alextokarew.javadebugging.multithreaded)
run$$$capture:1789:37, CompletableFuture$AsyncSupply (java.util.concurrent)
run:-1:5, CompletableFuture$AsyncSupply (java.util.concurrent)
Async stack trace
<init>:1775, CompletableFuture$AsyncSupply (java.util.concurrent)
asyncSupplyStage:1803, CompletableFuture (java.util.concurrent)
supplyAsync:2010, CompletableFuture (java.util.concurrent)
getOrders:21, AsyncExample (com.github.alextokarew.javadebugging.multithreaded)
lambda$main$0:34, AsyncExample (com.github.alextokarew.javadebugging.multithreaded)
tryFire:1171, CompletableFuture$UniCompose (java.util.concurrent)
postComplete:531, CompletableFuture (java.util.concurrent)
run$$$capture:1794, CompletableFuture$AsyncSupply (java.util.concurrent)
run:-1, CompletableFuture$AsyncSupply (java.util.concurrent)
Async stack trace
<init>:1775, CompletableFuture$AsyncSupply (java.util.concurrent)
asyncSupplyStage:1803, CompletableFuture (java.util.concurrent)
supplyAsync:2010, CompletableFuture (java.util.concurrent)
getUserByName:13, AsyncExample (com.github.alextokarew.javadebugging.multithreaded)
main:33, AsyncExample (com.github.alextokarew.javadebugging.multithreaded)

Async Stack Traces в Intellij IDEA



Async Stack Traces

По умолчанию умеет работать
с Java Concurrency API и Swing

Async Stack Traces

По умолчанию умеет работать с Java Concurrency API и Swing

При запуске отлаживаемого процесса IDEA добавляет javaagent

Который добавляет отладочную информацию на этапе загрузки классов

```
> java ... -javaagent:debugger-agent.jar=...
```

Async Stack Traces

По умолчанию умеет работать с Java Concurrency API и Swing

При запуске отлаживаемого процесса IDEA добавляет javaagent

Который добавляет отладочную информацию на этапе загрузки классов

```
> java ... -javaagent:debugger-agent.jar=...
```

Также позволяет указать произвольные точки запуска и выполнения асинхронных задач через аннотации

```
> org.jetbrains.annotations.@Async.Schedule
```

```
и @Async.Execute
```

В IDEA есть возможность переопределить эти аннотации

Дебаггинг многопоточки: выводы

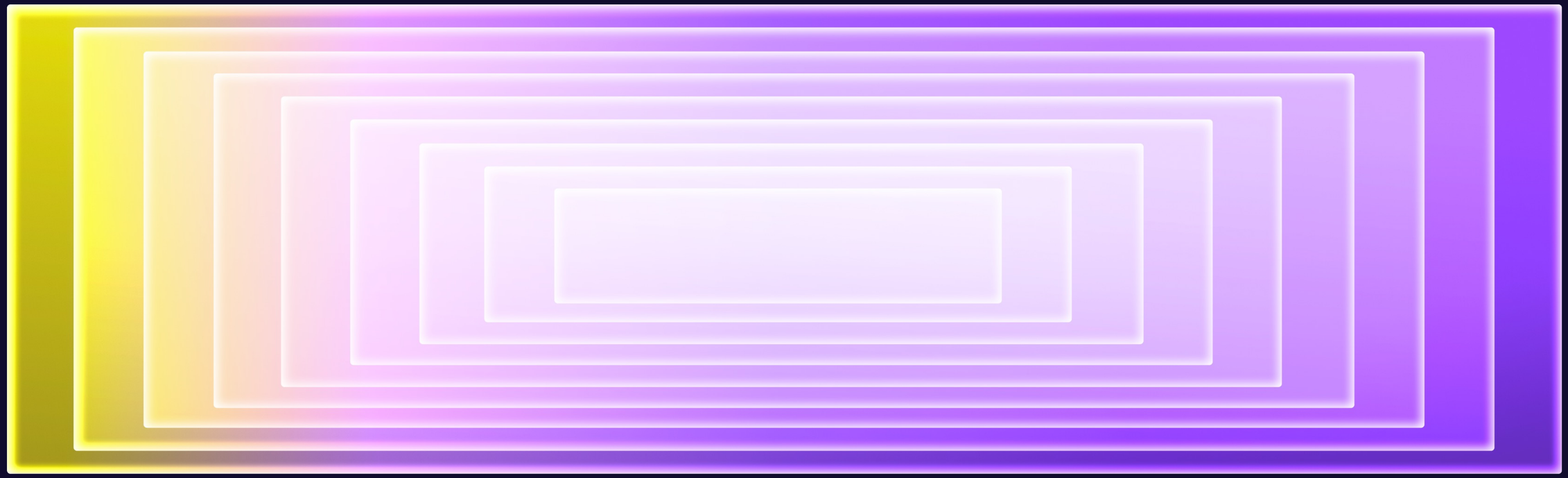
Дебаггинг многопоточки: выводы

Дебаггер — не самый подходящий инструмент для поиска проблем в многопоточном коде

Дебаггинг многопоточки: выводы

Дебаггер — не самый подходящий инструмент для поиска проблем в многопоточном коде

Инструменты дебаггинга из JDI и IDEA в ряде случаев могут быть полезны для воспроизведения проблем, вызванных многопоточностью.



Отладка интеграции через FFM API

Joker<?>
2024

FFM API — ничего общего с радио

Настя Лисицкая

Яндекс

Основные особенности

Одновременная работа с отладчиками JVM и нативного кода

Основные особенности

Одновременная работа с отладчиками JVM и нативного кода

Нативный код должен быть скомпилирован с отладочной информацией

Для gcc обязательно использовать флаг -g для включения отладочной информации и -O0, чтобы отключить оптимизации

Пример отладки с использованием FFM API

```
 1> #include <iostream>
 2> #include "fibffm.h"
 3>
 4> int fibonacci(int n) {
 5>     int prev = 0, result = 1;
 6>     for (int i = 1; i < n; i++) {
 7>         int old_result = result;
 8>         result = result + prev;
 9>         prev = old_result;
10>     }
11>     std::cout << "Finish ffm fibonacci" << std::endl;
12>     return result;
13> }
```

Пример отладки с использованием FFM API

```
 8> try (Arena arena = Arena.ofConfined()) {
 9>   var sl = SymbolLookup.libraryLookup(".../libcpp-fib.so", arena);
10>   MemorySegment memSeg = sl.find("fibonacci").orElseThrow();
11>   Linker linker = Linker.nativeLinker();
12>
13>   var desc = FunctionDescriptor.of(ValueLayout.JAVA_INT,
14>                                    ValueLayout.JAVA_INT);
15>
16>   MethodHandle fibonacci = linker.downcallHandle(memSeg, desc);
17>   var result = fibonacci.invoke(12);
18>   IO.println("Result: " + result);
19> }
```

Пример отладки с использованием FFM API

Пример отладки с использованием FFM API

</> Запуск JVM-процесса с JDWP-агентом:

```
> java ... -agentlib:jdwp=...
server=y,suspend=y,address=5005 com...FibFFM
```

Пример отладки с использованием FFM API

</> Запуск JVM-процесса с JDWP-агентом:

```
> java ... -agentlib:jdwp=...
server=y,suspend=y,address=5005 com...FibFFM
```

</> Подключение нативного отладчика к процессу JVM:

```
> gdb -p <JVM_Process_PID>
```

Пример отладки с использованием FFM API

</> Запуск JVM-процесса с JDWP-агентом:

```
> java ... -agentlib:jdwp=...
server=y,suspend=y,address=5005 com...FibFFM
```

</> Подключение нативного отладчика к процессу JVM:

```
> gdb -p <JVM_Process_PID>
```

</> Подключение JPDA-отладчика к JVM:

```
> jdb -attach 5005
```

Пример отладки с использованием FFM API

JDB

GDB

Пример отладки с использованием FFM API

JDB

GDB

```
1> break fibonacci
Function "fibonacci" not defined. Make
breakpoint pending ... (y or [n])?
```

Пример отладки с использованием FFM API

JDB

GDB

```
1> break fibonacci
Function "fibonacci" not defined. Make
breakpoint pending ... (y or [n])?
2> continue
```

Пример отладки с использованием FFM API

JDB

```
3> stop in FibFFM:18
```

GDB

```
1> break fibonacci
Function "fibonacci" not defined. Make
breakpoint pending ... (y or [n])?
2> continue
```

Пример отладки с использованием FFM API

```
 8> try (Arena arena = Arena.ofConfined()) {
 9>   var sl = SymbolLookup.libraryLookup(".../libcpp-fib.so", arena);
10>   MemorySegment memSeg = sl.find("fibonacci").orElseThrow();
11>   Linker linker = Linker.nativeLinker();
12>
13>   var desc = FunctionDescriptor.of(ValueLayout.JAVA_INT,
14>                                    ValueLayout.JAVA_INT);
15>
16>   MethodHandle fibonacci = linker.downcallHandle(memSeg, desc);
17>   var result = fibonacci.invoke(12);
18>   IO.println("Result: " + result);
19> }
```

Пример отладки с использованием FFM API

JDB

```
3> stop in FibFFM:18
4> run
```

GDB

```
1> break fibonacci
Function "fibonacci" not defined. Make
breakpoint pending ... (y or [n])?
2> continue
Thread 2 "java" hit Breakpoint 1
```

Пример отладки с использованием FFM API

JDB

```
3> stop in FibFFM:18
4> run
```

GDB

```
1> break fibonacci
Function "fibonacci" not defined. Make
breakpoint pending ... (y or [n])?
2> continue
Thread 2 "java" hit Breakpoint 1
5> break +3
```

Пример отладки с использованием FFM API

```
 1> #include <iostream>
 2> #include "fibffm.h"
 3>
 4> int fibonacci(int n) {
 5>     int prev = 0, result = 1;
 6>     for (int i = 1; i < n; i++) {
 7>         int old_result = result;
 8>         result = result + prev;
 9>         prev = old_result;
10>     }
11>     std::cout << "Finish ffm fibonacci" << std::endl;
12>     return result;
13> }
```

Пример отладки с использованием FFM API

JDB

```
3> stop in FibFFM:18
4> run
```

GDB

```
1> break fibonacci
Function "fibonacci" not defined. Make
breakpoint pending ... (y or [n])?
2> continue
Thread 2 "java" hit Breakpoint 1
5> break +3
6> continue (x3)
Thread 2 "java" hit Breakpoint 2
```

Пример отладки с использованием FFM API

JDB

```
3> stop in FibFFM:18
4> run
```

GDB

```
1> break fibonacci
Function "fibonacci" not defined. Make
breakpoint pending ... (y or [n])?
2> continue
Thread 2 "java" hit Breakpoint 1
5> break +3
6> continue (x3)
Thread 2 "java" hit Breakpoint 2
7> info locals
old_result = 2
```

Пример отладки с использованием FFM API

JDB

```
3> stop in FibFFM:18
4> run
```

GDB

```
1> break fibonacci
Function "fibonacci" not defined. Make
breakpoint pending ... (y or [n])?
2> continue
Thread 2 "java" hit Breakpoint 1
5> break +3
6> continue (x3)
Thread 2 "java" hit Breakpoint 2
7> info locals
old_result = 2
8> set old_result=123
```

Пример отладки с использованием FFM API

JDB

```
3> stop in FibFFM:18
4> run
```

GDB

```
1> break fibonacci
Function "fibonacci" not defined. Make
breakpoint pending ... (y or [n])?
2> continue
Thread 2 "java" hit Breakpoint 1
5> break +3
6> continue (x3)
Thread 2 "java" hit Breakpoint 2
7> info locals
old_result = 2
8> set old_result=123
9> del
```

Пример отладки с использованием FFM API

JDB

```
3> stop in FibFFM:18
4> run
Breakpoint hit: "thread=main" ...
```

GDB

```
1> break fibonacci
Function "fibonacci" not defined. Make
breakpoint pending ... (y or [n])?
2> continue
Thread 2 "java" hit Breakpoint 1
5> break +3
6> continue (x3)
Thread 2 "java" hit Breakpoint 2
7> info locals
old_result = 2
8> set old_result=123
9> del
10> continue
```

Пример отладки с использованием FFM API

JDB

```
3> stop in FibFFM:18
4> run
Breakpoint hit: "thread=main" ...
11> print result
result = "2685"
```

GDB

```
1> break fibonacci
Function "fibonacci" not defined. Make
breakpoint pending ... (y or [n])?
2> continue
Thread 2 "java" hit Breakpoint 1
5> break +3
6> continue (x3)
Thread 2 "java" hit Breakpoint 2
7> info locals
old_result = 2
8> set old_result=123
9> del
10> continue
```

Пример отладки с использованием FFM API

JDB

```
3> stop in FibFFM:18
4> run
Breakpoint hit: "thread=main" ...
11> print result
result = "2685"
12> cont
```

GDB

```
1> break fibonacci
Function "fibonacci" not defined. Make
breakpoint pending ... (y or [n])?
2> continue
Thread 2 "java" hit Breakpoint 1
5> break +3
6> continue (x3)
Thread 2 "java" hit Breakpoint 2
7> info locals
old_result = 2
8> set old_result=123
9> del
10> continue
```

Отладка при использовании FFM API

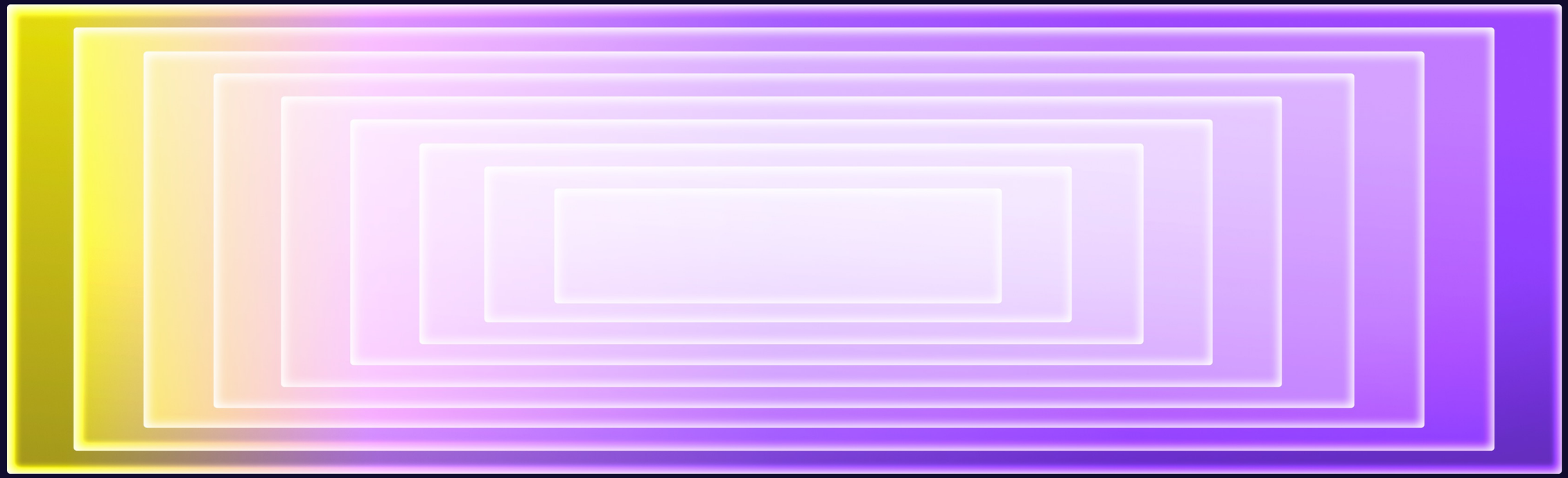
Отладка при использовании FFM API

При использовании FFM API нативные библиотеки динамически линкуются к Java-процессу, поэтому gdb нужно подключать к нему

Отладка при использовании FFM API

При использовании FFM API нативные библиотеки динамически линкуются к Java-процессу, поэтому gdb нужно подключать к нему

Подход также применим для отладки JNI



Отладка
за пределами JPDA

B Intellij IDEA

Async stacktraces

В Intellij IDEA

Async stacktraces

Уже рассмотрели

В Intellij IDEA

Async stacktraces

Уже рассмотрели

Java Stream Debugging

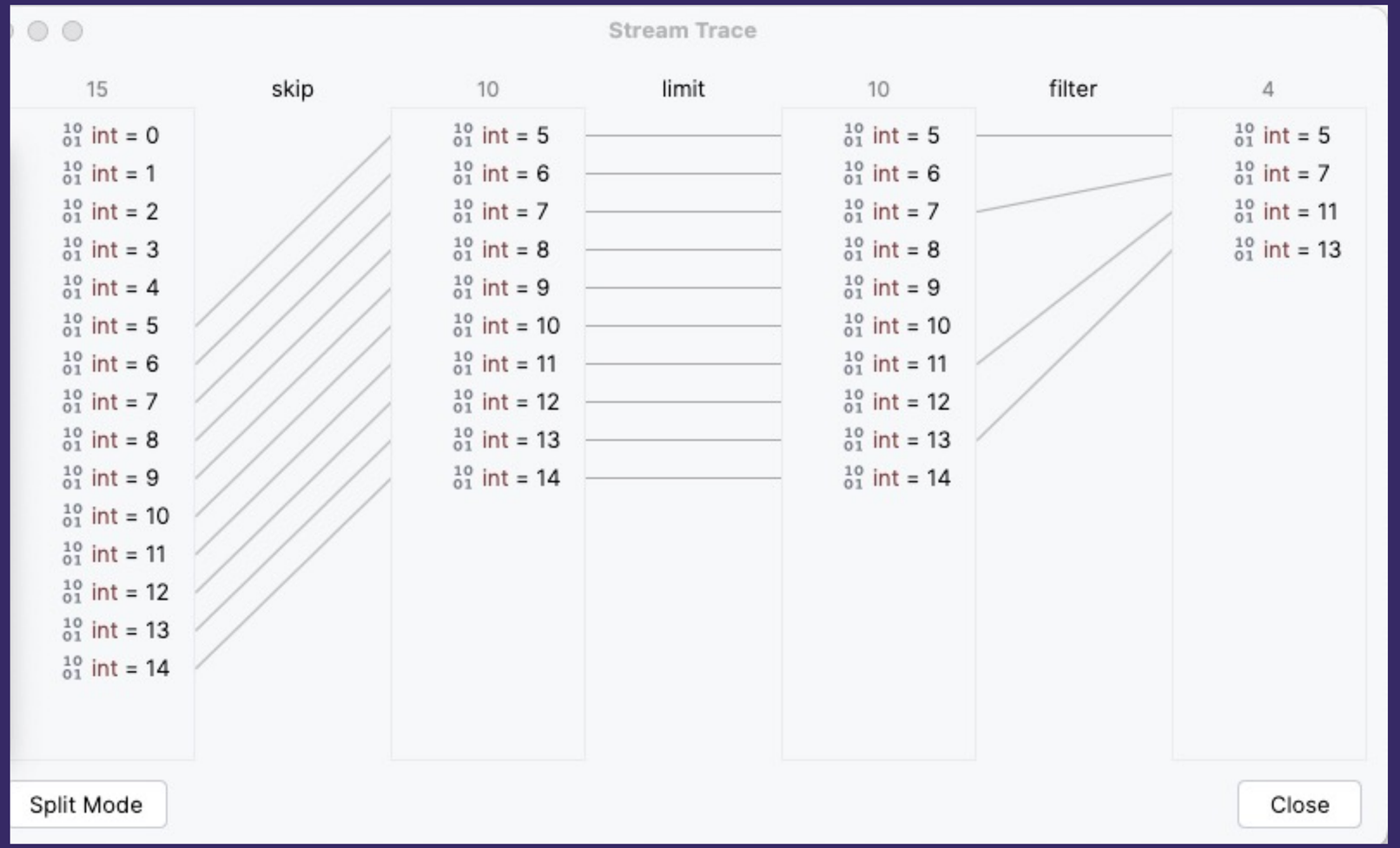
Java Stream debugging

```
// Prime numbers from 5 to 14

int skip = 5;
int limit = 10;

IntStream.iterate(0, n -> n + 1)
         .skip(skip)
         .limit(limit)
         .filter(StreamChain::isPrime)
         .forEach(System.out::println);
```

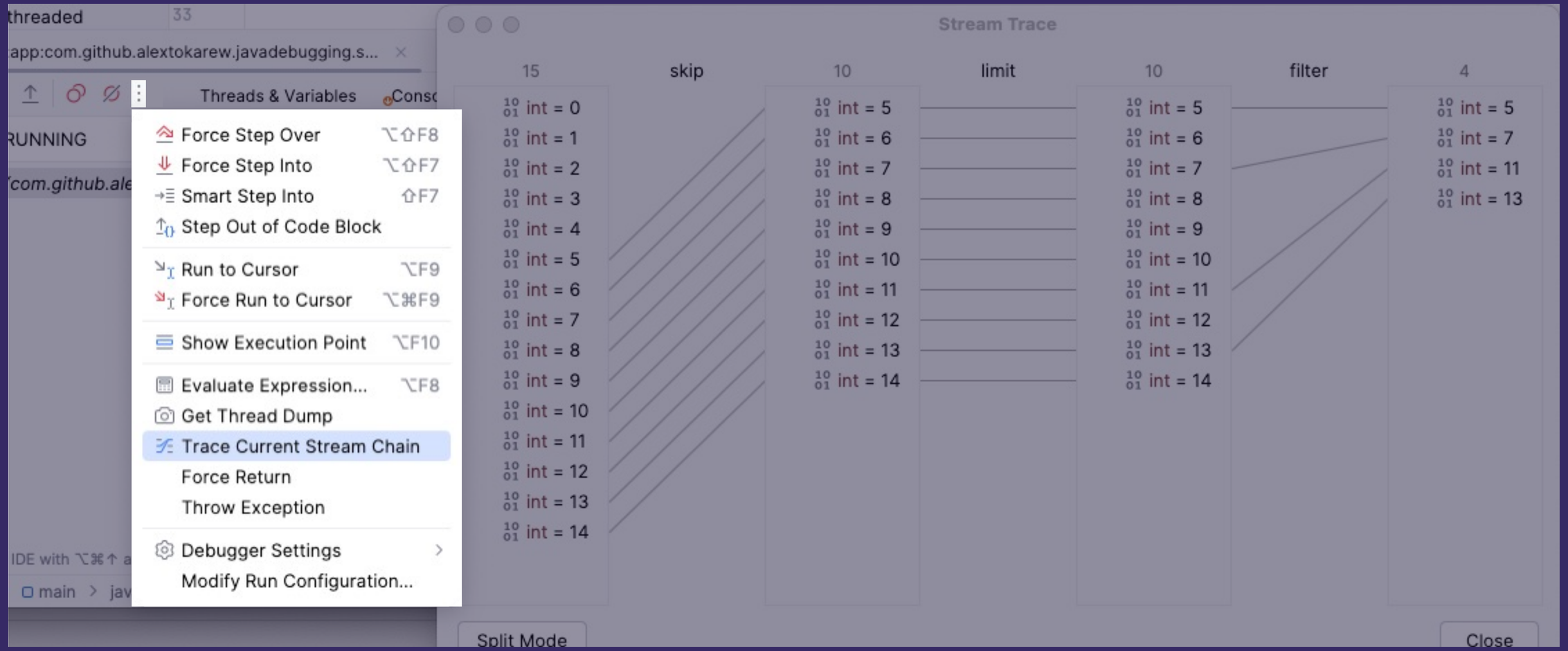
Java Stream debugging

```
// Prime numbers from 5 to 14

int skip = 5;
int limit = 10;

IntStream.iterate(0, n -> n + 1)
         .skip(skip)
         .limit(limit)
         .filter(StreamChain::isPrime)
         .forEach(System.out::println);
```

Java Stream debugging



Java Stream debugging



Java Stream debugging: ограничения

Java Stream debugging: ограничения

Работает только
для файлов
из активного проекта

Java Stream debugging: ограничения

Работает только для файлов из активного проекта

С библиотеками и декомпилированным кодом — не работает

Java Stream debugging: ограничения

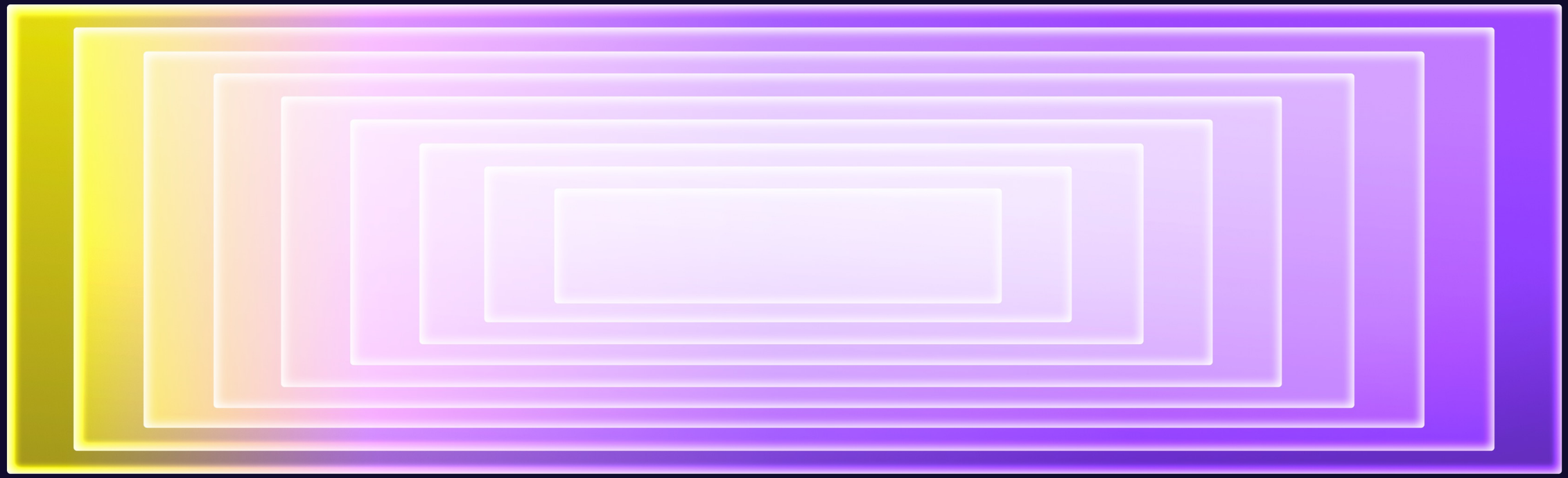
Работает только для файлов из активного проекта

С библиотеками и декомпилированным кодом — не работает

Не работает с объединением предварительно обработанных стримов

Вот так — не работает

```
List<String> fruits = List.of("apple", "Banana", "cherry", "date");
List<String> berries = List.of(" blueberry", "Raspberry  ", "  strawberry",
"gooseberry", "apple");

Stream<String> processedFruits = fruits.stream()
                                       .map(String::toLowerCase)
                                       .filter(s -> s.length() >= 5)
                                       .map(s -> "fruit_" + s);
Stream<String> processedBerries = berries.stream()
                                         .map(s -> s.strip())
                                         .filter(s -> s.contains("berry"))
                                         .map(s -> s + "_berry");

Map<Integer, List<String>> result = Stream.concat(processedFruits, processedBerries)
                                          .distinct()
                                          .sorted()
                                          .collect(groupingBy(String::length,
                                           toList()));
```



Заключение

Выводы

Отладка и дебаггинг — это целое направление Software Engineering, для которого даже двух докладов мало

Выводы

Отладка и дебаггинг — это целое направление Software Engineering, для которого даже двух докладов мало

JDB — с виду простой, но достаточно мощный инструмент отладки, предоставляющий широкий спектр возможностей для дебаггинга

Выводы

Отладка и дебаггинг — это целое направление Software Engineering, для которого даже двух докладов мало

JDB — с виду простой, но достаточно мощный инструмент отладки, предоставляющий широкий спектр возможностей для дебаггинга

Комбинируйте дебаггинг с другими методами отладки для получения более полной картины

Ссылки

https://github.com/alextokarew/java-debugging

Java Platform Debug Architecture (JPDA)

Java Troubleshooting Guide

Infrastructure

Вопросы

Александр Токарев

YTsaurus SPYT Tech Lead
(Apache Spark в YTsaurus)
в Яндексе