

Вебинар

# Отладка в IntelliJ IDEA



Спикер: Шибков Константин

30 сентября 2021

образовательная платформа

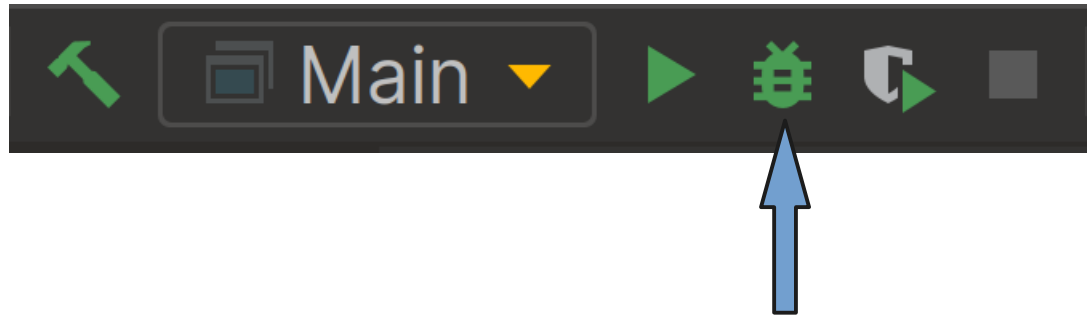
Skillbox

# Логирование не заменяет отладку

**Логирование** – получение информации постфактум. Для получения новых данных после изменения, требуется пройти весь цикл работы программы.

**Отладка** – изучение работы программы непосредственно во время исполнения. “Замораживаем” время и осматриваемся, изучаем состояние. При желании меняем состояние на лету и оцениваем результат.

# Запуск режима отладки (Debug)

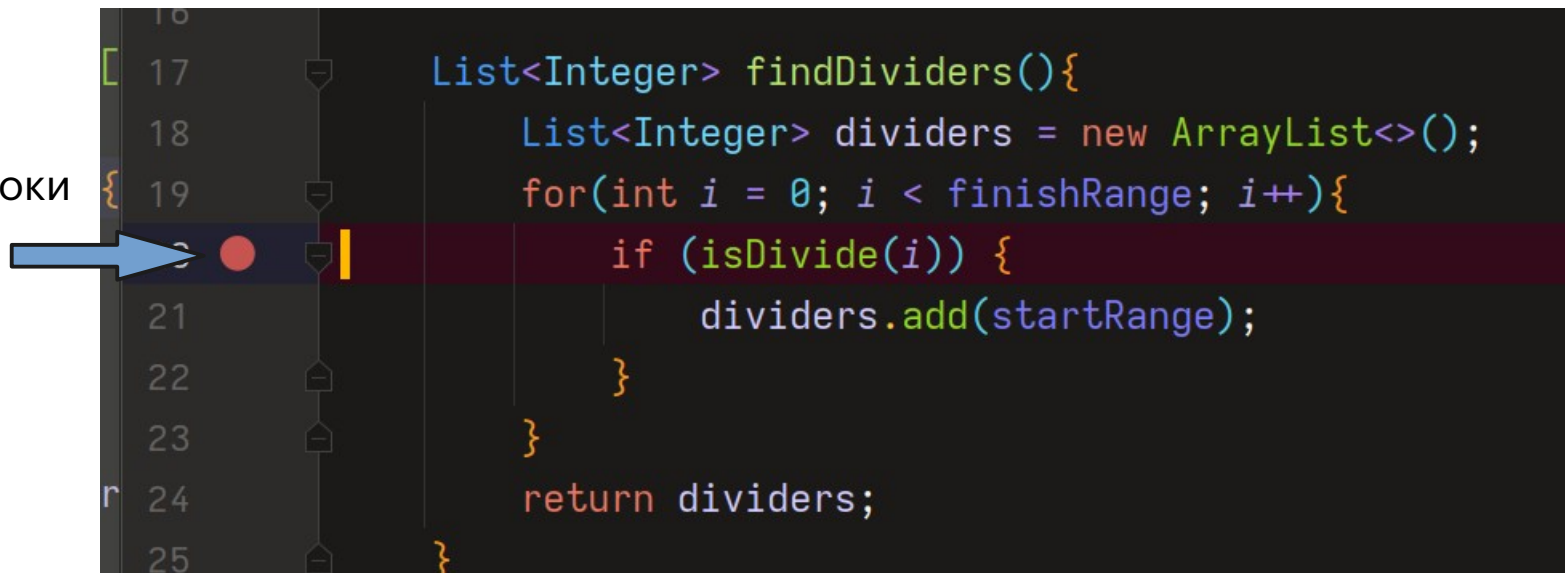


или горячие клавиши:

Windows	Shift+F9
macOS	⌘D

# Установка breakpoint

кликнуть на панели  
правее номера строки



или нажать горячие клавиши, находясь на строке:

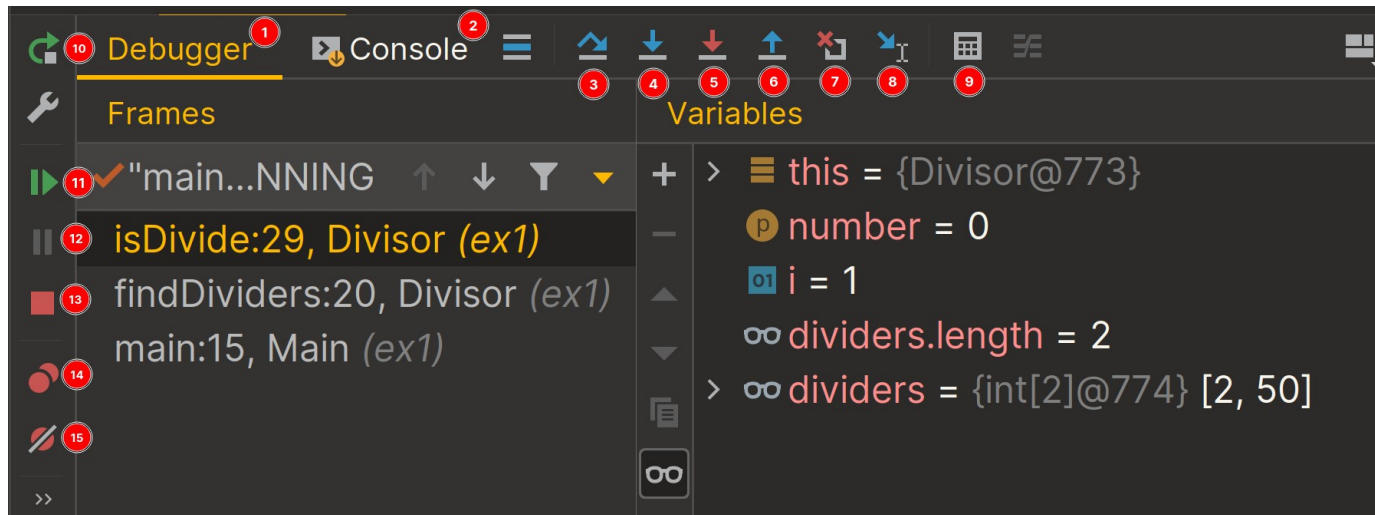
Windows	Ctrl+F8
macOS	⌘F8

# Интерфейс вкладки Debugger

1 – Вкладка с инструментами дебага  
2 – Консоль запуска с дебагом (для  
обычного запуска используется другая  
консоль)

## Навигация по коду

3 – Step Over (F8) - перейти к следующему  
выражению  
4 – Step Into (F7) – пройти внутрь метода в  
выражении (методы сторонних библиотек  
игнорируются)  
5 – Force Step Into (Alt+Shift+F8 /  $\text{Ctrl} \uparrow$  F8) –  
принудительно перейти в метод, даже  
пропущенный командой Step Into  
6 – Step Out (Shift+F8 /  $\text{Ctrl} \uparrow$  F8) – выйти из  
текущего метода, к выражению вызвавшее  
его.  
7 – Drop Frame – удалить из стека методов,  
выбранный метод и все выше него.  
8 – Run to Cursor (Alt+Shift+F9 /  $\text{Ctrl} \uparrow$  F9) –  
перейти и остановиться на выражении под  
курсором.



9 – Evaluate Expression - Открыть окно для  
вычислений и написания выражений (можно  
менять состояние объектов)  
10 – Rerun (Ctrl + F5 /  $\text{Ctrl} \uparrow$  F5) – перезапуск  
программы в режиме Debug  
11 – Resume (F9) – продолжить выполнение  
программы.  
12 – Pause – остановить программу, будет  
показана строка на которой программа  
находилась.

13 – Stop (Ctrl + F2 /  $\text{Ctrl} \uparrow$  F2) – остановка  
приложения  
14 – View Breakpoints (Ctrl + Shift + F8 /  $\text{Ctrl} \uparrow$  F8) – посмотреть список точек останова  
15 – Mute Breakpoints – включить режим -  
игнорировать все точки останова .

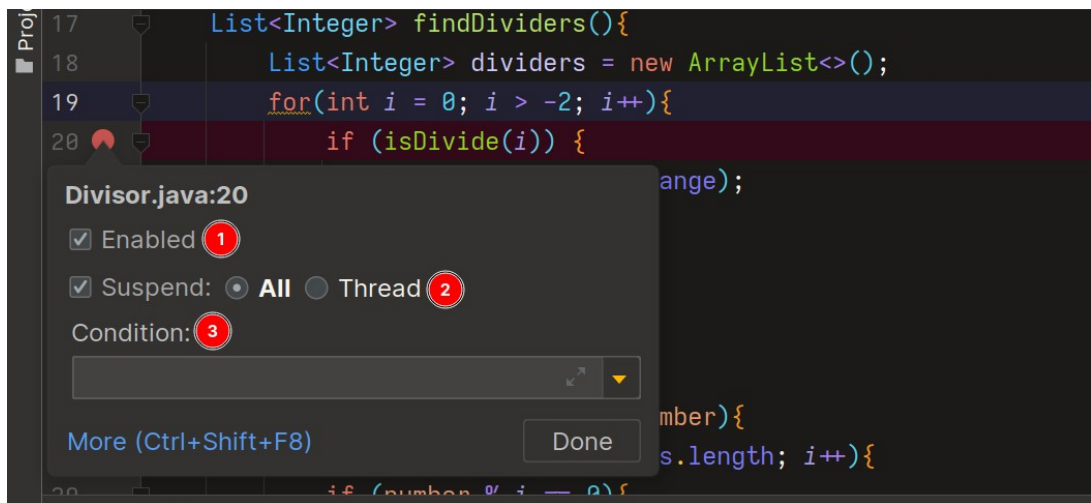
# Интерфейс настройки breakpoints

При открытии меню на точке останова:

1 – Enabled - Активна ли точка останова

3 – Suspend - все потоки будут остановлены или только который дошел до точки останова.

4 – Условие при котором будет активна точка останова



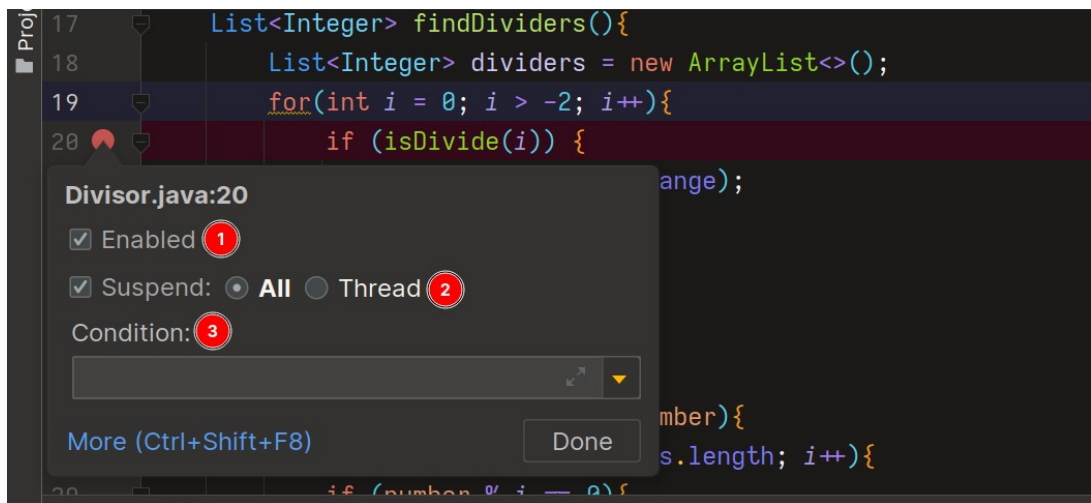
# Интерфейс настройки breakpoints

При открытии меню на точке останова:

1 – Enabled - Активна ли точка останова

3 – Suspend - все потоки будут остановлены или только который дошел до точки останова.

4 – Условие при котором будет активна точка останова



# Конструктор по умолчанию

Для остановки при выполнении конструктора по умолчанию, поставьте точку на имя класса.

```
4 import java.util.List;
5
6 public class Divisor {
7     int[] dividers;    dividers: null
8     int startRange;    startRange: 0
9     int finishRange;   finishRange: 0
10
```



# Контроль значения переменной

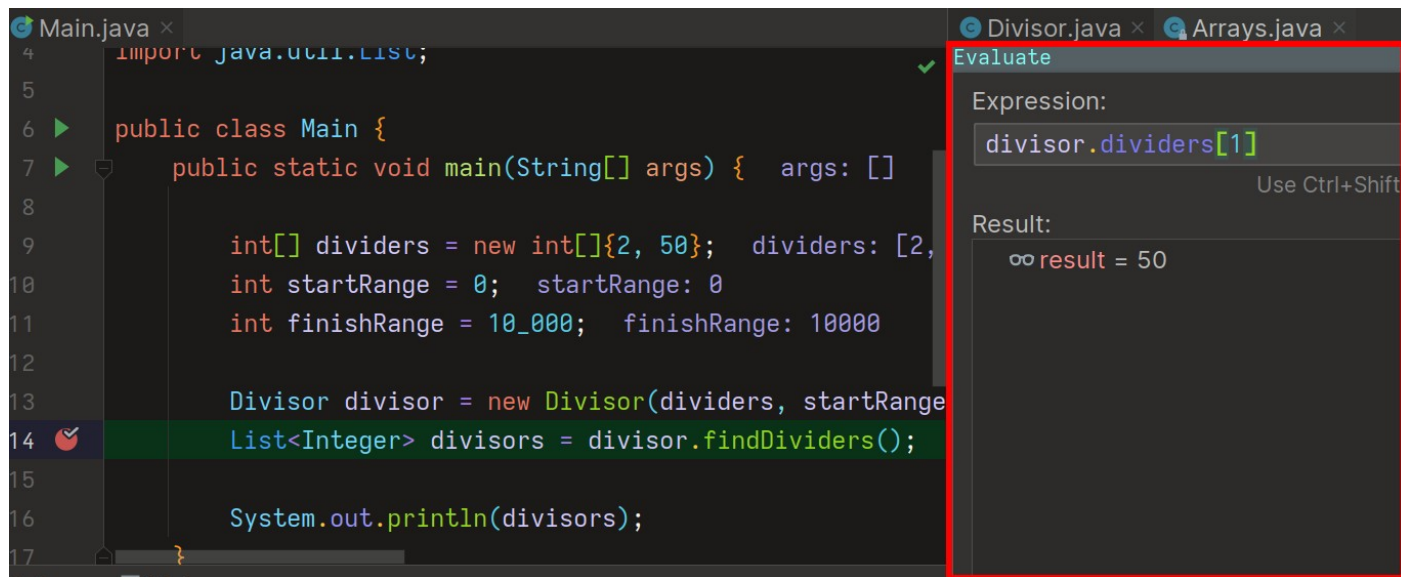
Для контроля в точке установки значения переменной → поставьте точку на переменной, в каждом выражении меняющем значение, программа будет остановлена

1 – Точка останова, поставлена на строку 8.

2 – Программа остановится на строке 13, так как там изменяется значение переменной. Точку останова не ставили на этой строке.

```
5
6 public class Divisor {
7     int[] dividers; dividers: [2, 50]
8     int startRange; startRange: 0
9     int finishRange; finishRange: 0
10
11     public Divisor(int[] dividers, int startRange,
12         this.dividers = dividers; dividers: [2, 50]
13         this.startRange = startRange; startRange:
14         this.finishRange = finishRange;
15     }
16
```

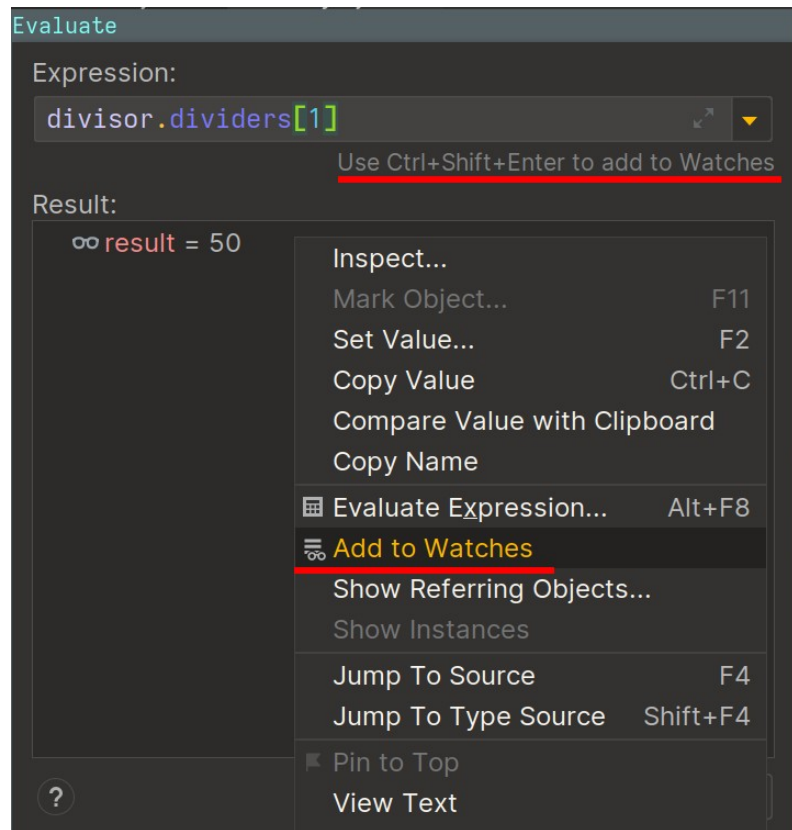
# Evaluate Expressions



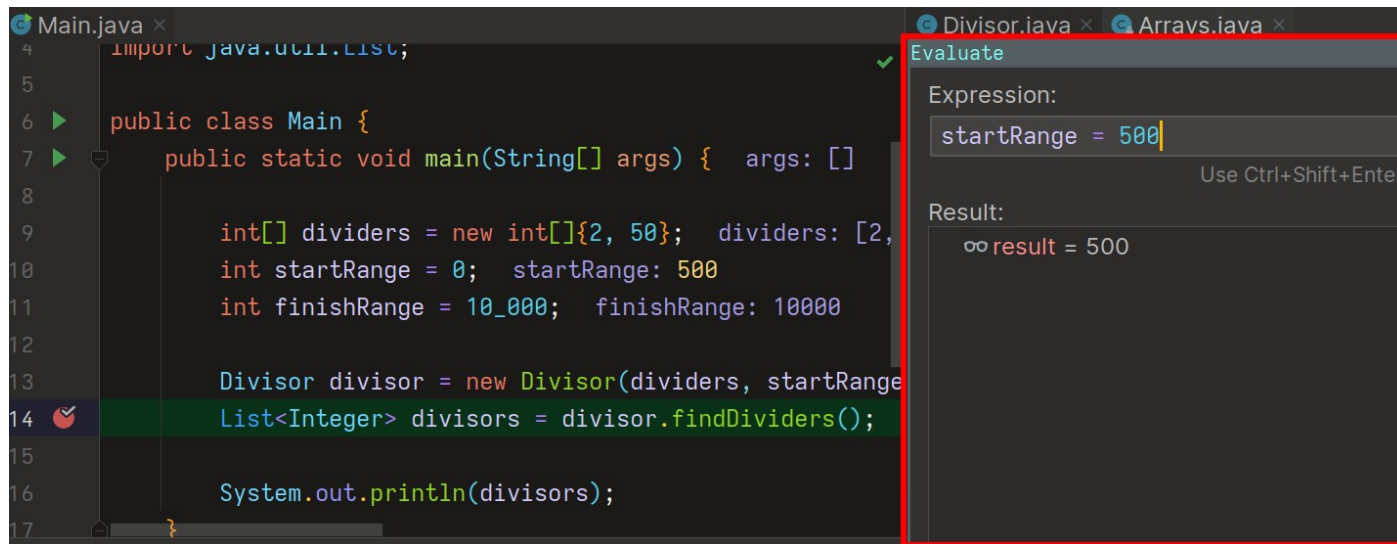
Откройте Evaluate Expressions и можете вводить любое выражение для исследования состояния. Возможно обращаться к приватным полям классов, использовать стримы.

# Evaluate Expressions

Если вам требуется следить за состоянием объекта или переменной, можете добавить в список отслеживания (Watches)

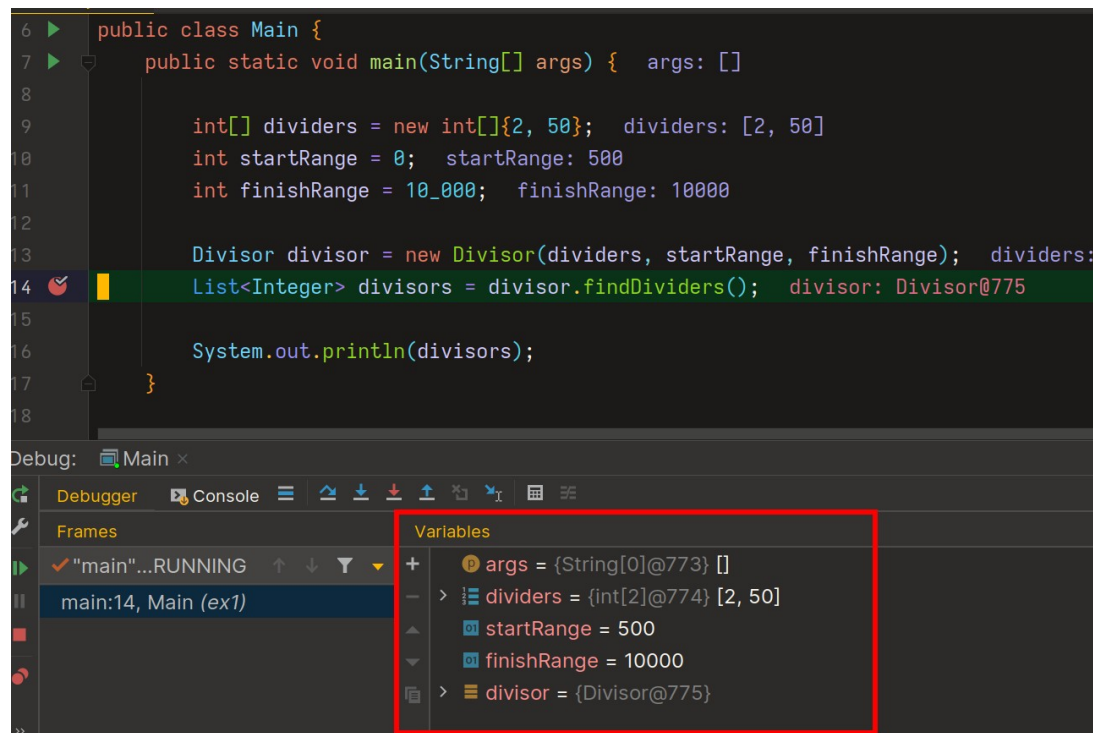


# Evaluate Expressions



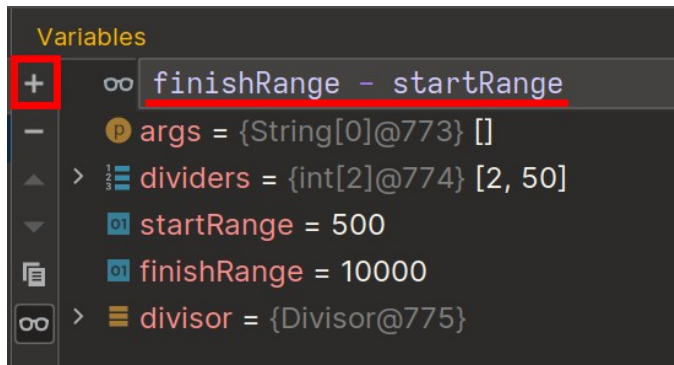
Возможно изменение значений переменных  
и объектов.

# Variables

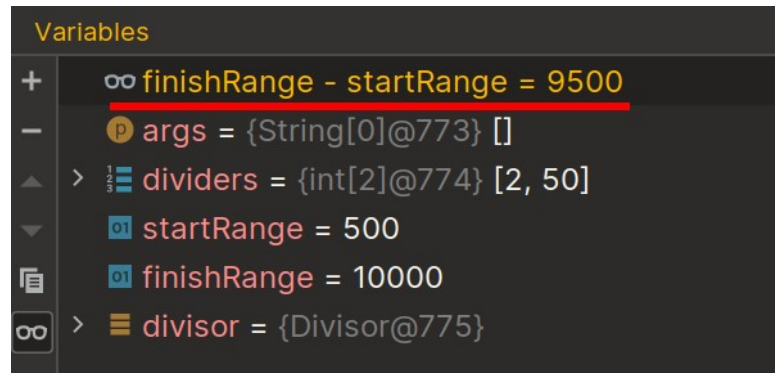


При остановке, во вкладке Variables доступны все переменные в области видимости метода.

# Variables



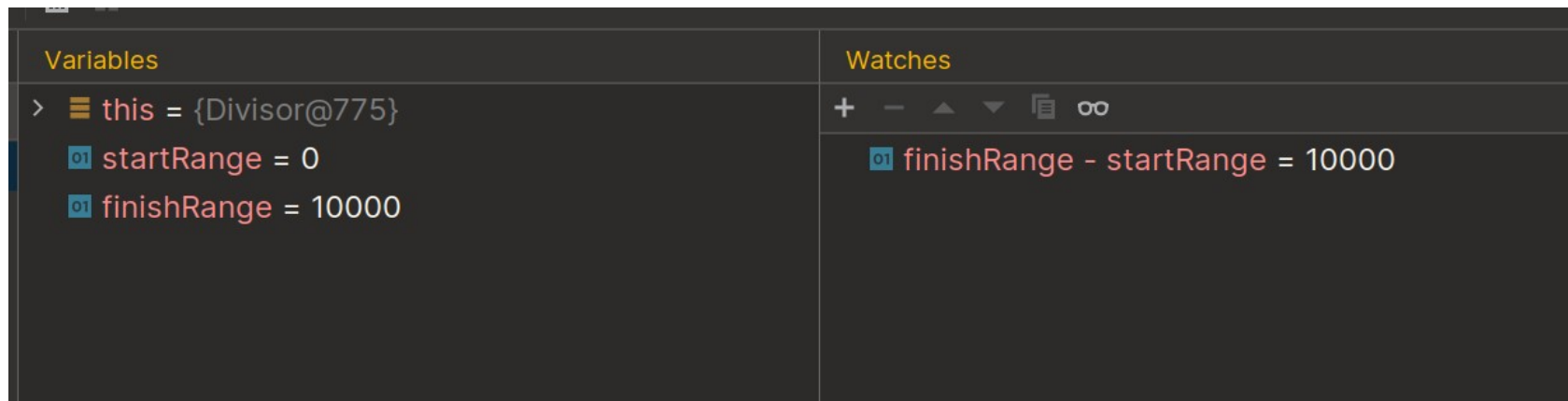
```
Variables
+ finishRange - startRange
- args = {String[0]@773} []
> 1 2 3 dividers = {int[2]@774} [2, 50]
01 startRange = 500
01 finishRange = 10000
> 01 divisor = {Divisor@775}
```




```
Variables
+ finishRange - startRange = 9500
- args = {String[0]@773} []
> 1 2 3 dividers = {int[2]@774} [2, 50]
01 startRange = 500
01 finishRange = 10000
> 01 divisor = {Divisor@775}
```

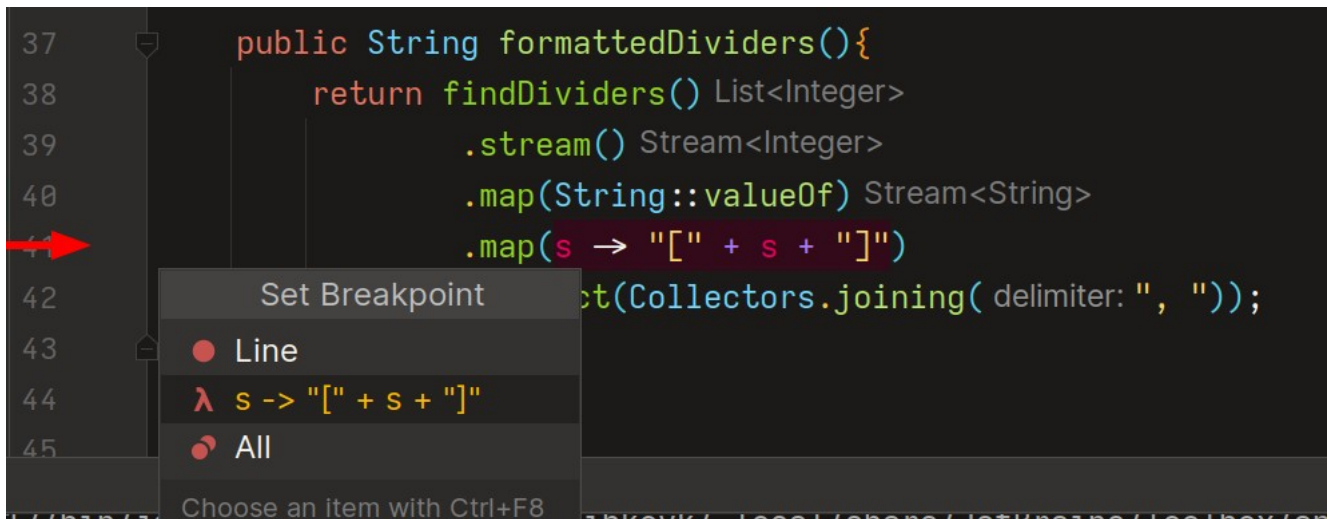
Вы можете добавлять в список вычисления и отслеживать их, для создания нажмите на кнопку + в меню слева

# Watches



Если отслеживаемых выражений достаточно много, можно выделить в отдельную вкладку, нажав на 

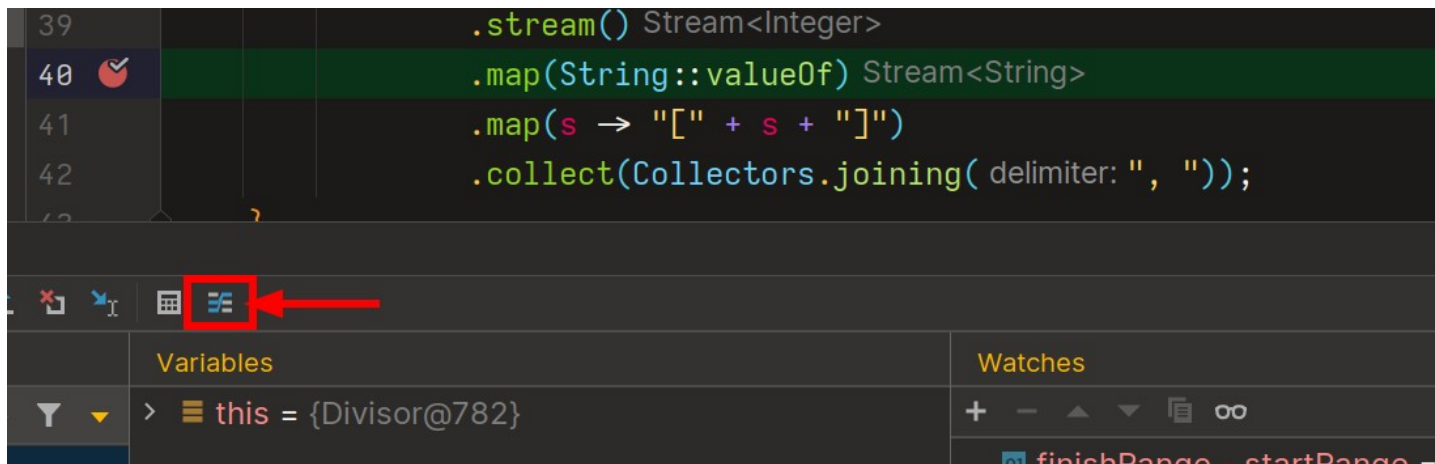
# Stream Debug



В современных версиях IDEA, есть встроенный визуальный дебаг Stream

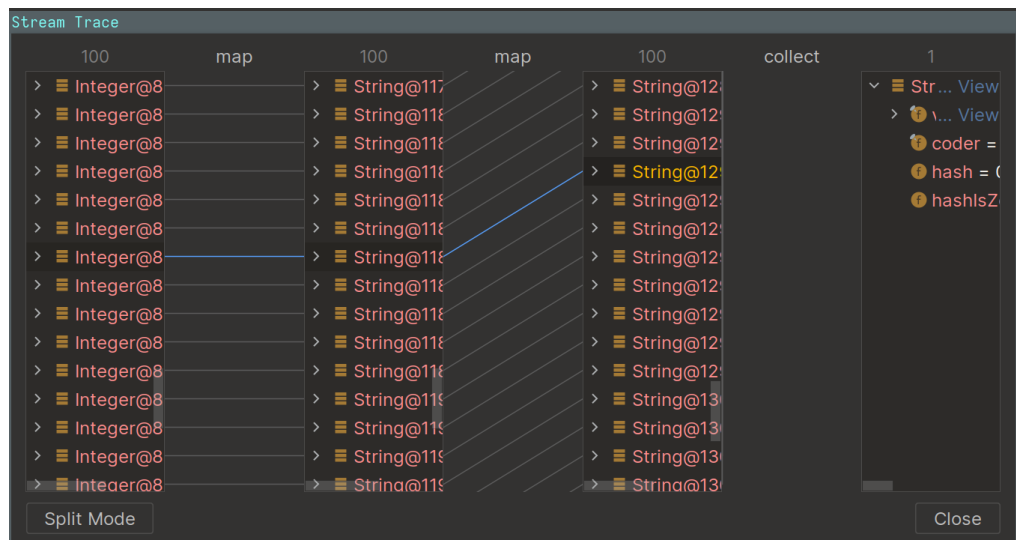


# Stream Debug



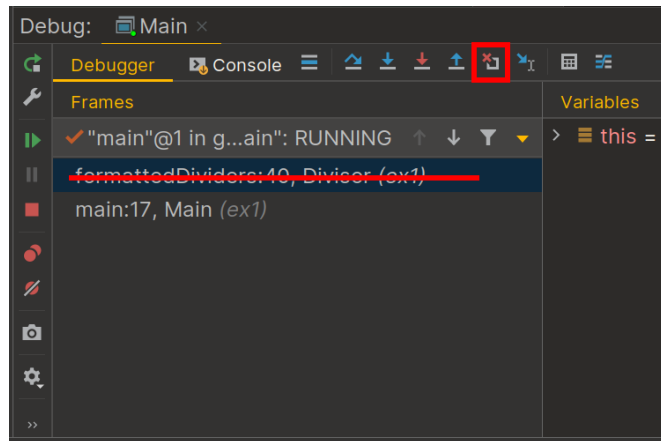
После остановки, появится активная кнопка Trace Current Stream Chain

# Stream Debug



После остановки, появится активная кнопка Trace Current Stream Chain, при нажатии откроет окно где будет показан весь путь каждой ветки.

# Frame Drop



Возможно удалить из стека методов указанный метод, и все методы выше и “откатиться” к выражению вызвавшее этот метод. Позволяет еще раз запустить выполнение метода при необходимости, без перезапуска всего приложения.

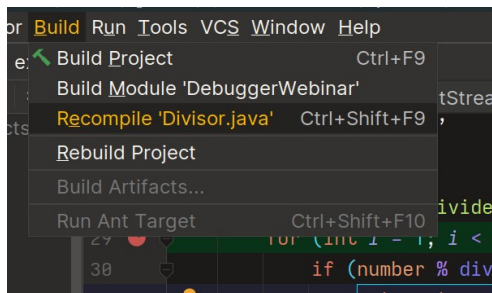
# «Горячая замена» кода

Нашли и поправили баг в одном из методов, возможно  
перекомпилировать файл, удалить фрейм и запустить уже исправленный  
код.

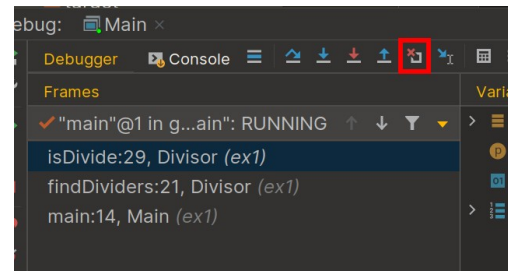
Меняем  
код



Перекомпилируем класс



Удаляем фрейм с измененным кодом



Такой вариант не всегда возможно использовать, так как методы могут  
изменять состояние приложения, что приведет к падению приложения.

Вебинар

# Спасибо за внимание!

Предложения и вопросы:

tg: @sendel

sendel@sendel.ru



Спикер: Шибков Константин

образовательная платформа

Skillbox