DebuggingSamples

Sample repository with code snippets to practice debugging with IntelliJ IDEA

Examples

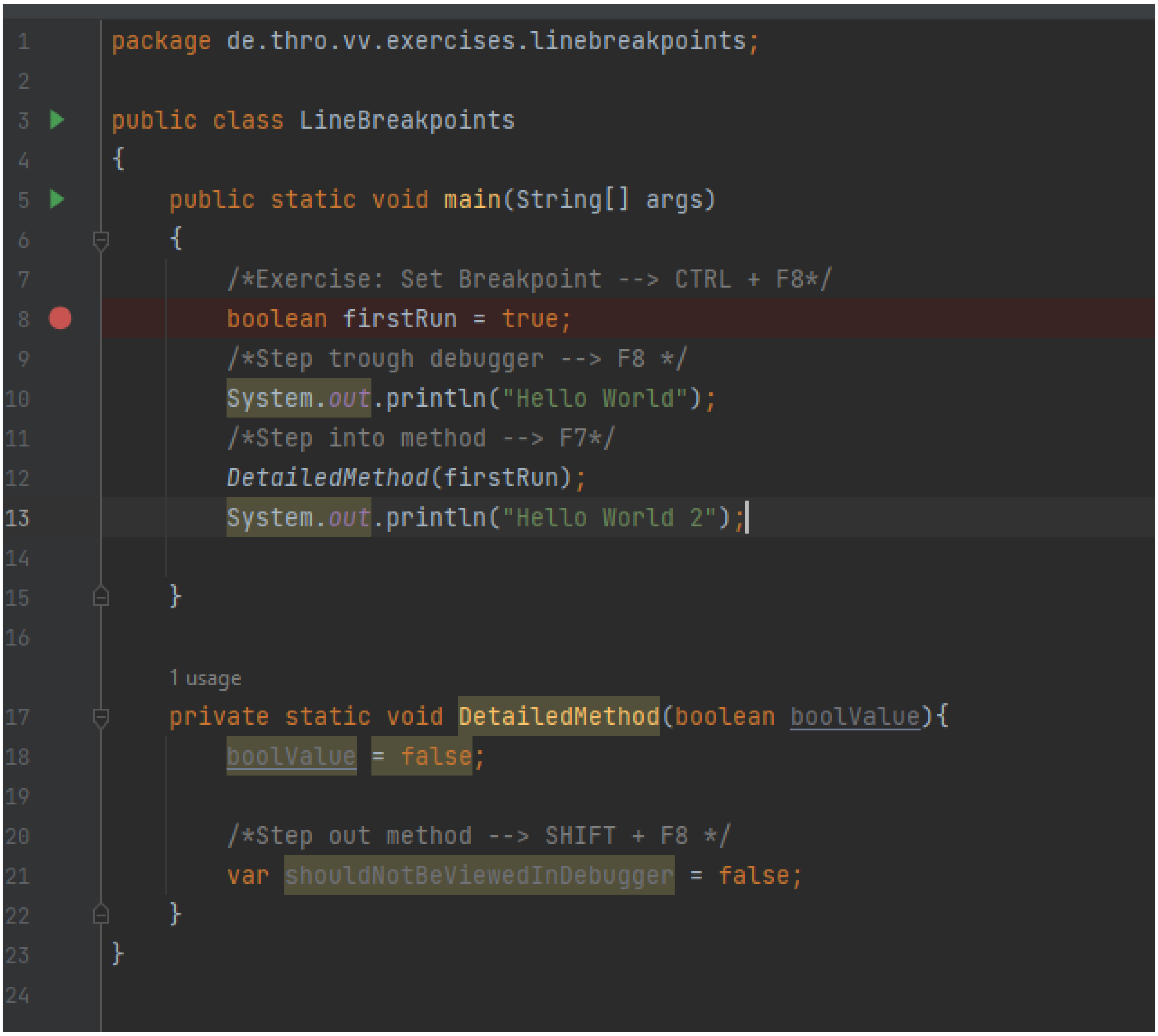
Breakpoints

Breakpoints are special markers in the development environment that stop regular program execution at a desired point. This allows the developer to examine the program state and behavior to detect errors and incorrect behavior.

Types of breakpoints

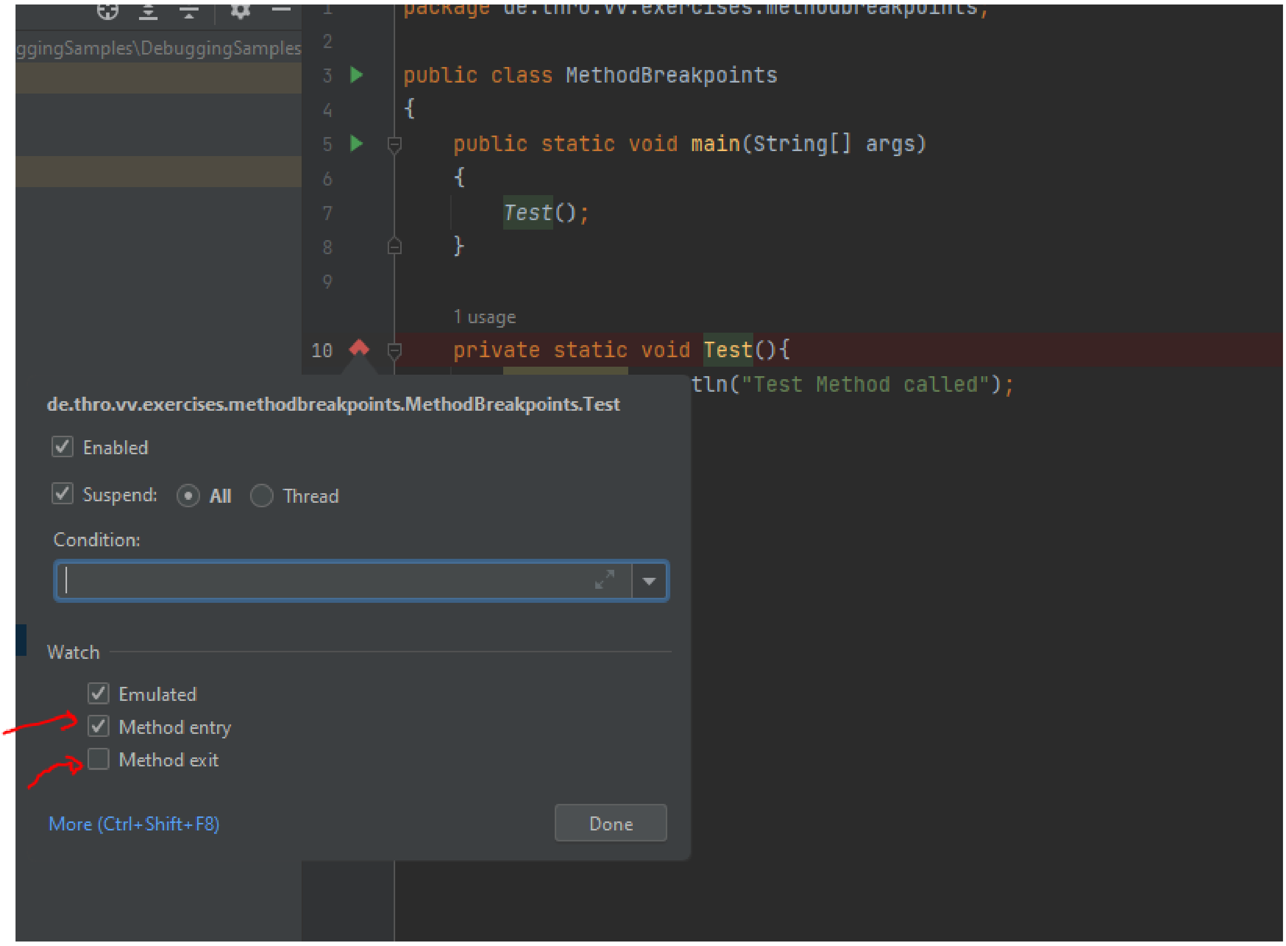
The following types of breakpoints are available in IntelliJ IDEA: (vgl. https://www.jetbrains.com/help/idea/using-breakpoints.html#set-breakpoints)

- Line Breakpoints: Suspend the program upon reaching the line of code where the breakpoint was set. This type of breakpoints can be set on any executable line of code.



```java
package de.thro.vv.exercises.linebreakpoints;

public class LineBreakpoints
{
    public static void main(String[] args)
    {
        /*Exercise: Set Breakpoint --> CTRL + F8*/
        boolean firstRun = true;
        /*Step trough debugger --> F8 */
        System.out.println("Hello World");
        /*Step into method --> F7*/
        DetailedMethod(firstRun);
        System.out.println("Hello World 2");

    }

    private static void DetailedMethod(boolean boolValue){
        boolValue = false;

        /*Step out method --> SHIFT + F8 */
        var shouldNotBeViewedInDebugger = false;
    }
}
```

- Method Breakpoints: Suspend the program upon entering or exiting the specified method or one of its implementations, allowing you to check the method's entry/exit conditions.



```
package de.thro.vv.exercises.methodbreakpoints;

public class MethodBreakpoints
{
    public static void main(String[] args)
    {
        Test();
    }

    1 usage
    private static void Test(){
        tln("Test Method called");
```

de.thro.vv.exercises.methodbreakpoints.MethodBreakpoints.Test

☑ Enabled

☑ Suspend: ● All ○ Thread

Condition:

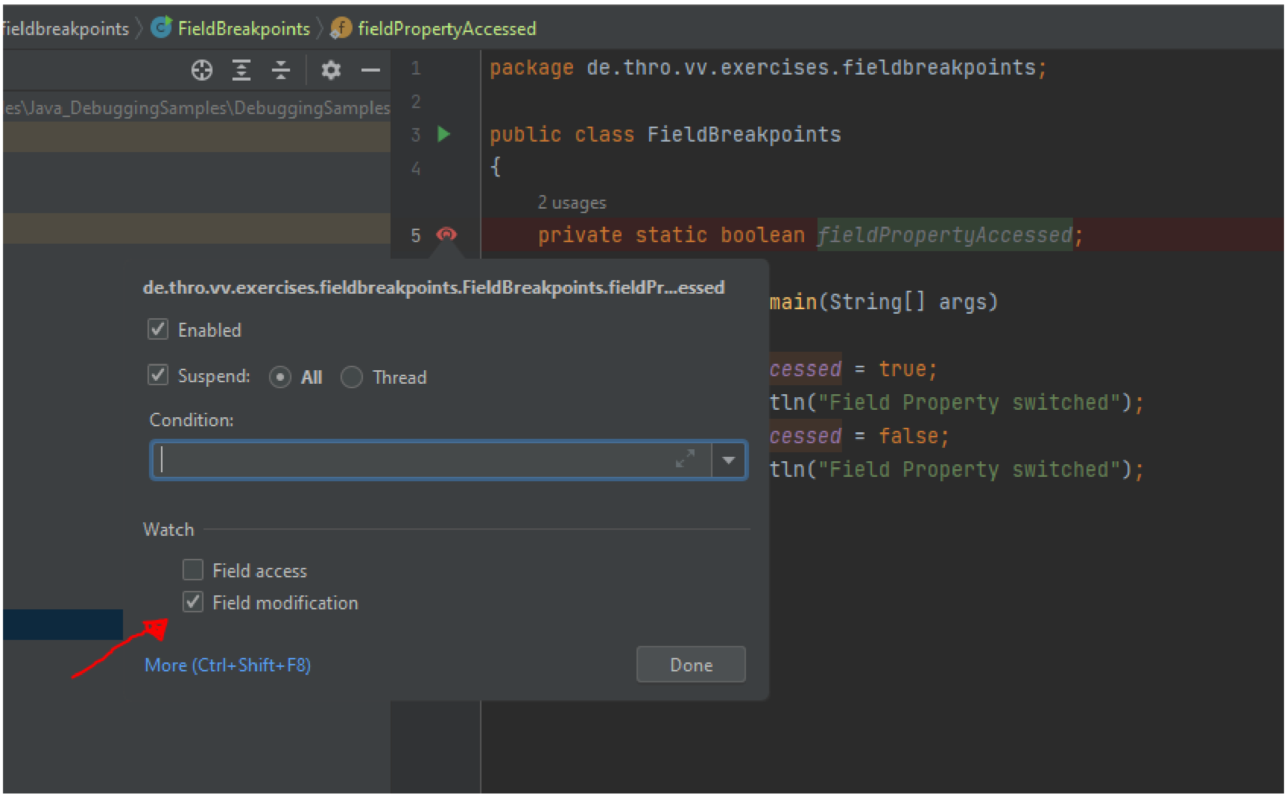
Watch

☑ Emulated

☑ Method entry

☐ Method exit

- Field Watchpoints: To detect access and modification of fields if caller source is unknown.


ieldbreakpoints 〉 FieldBreakpoints 〉 fieldPropertyAccessed
es\Java_DebuggingSamples\DebuggingSamples

```
package de.thro.vv.exercises.fieldbreakpoints;

public class FieldBreakpoints
{
    2 usages
    private static boolean fieldPropertyAccessed;

                        main(String[] args)

                        cessed = true;
                        tln("Field Property switched");
                        cessed = false;
                        tln("Field Property switched");
```

de.thro.vv.exercises.fieldbreakpoints.FieldBreakpoints.fieldPr...essed

☑ Enabled

☑ Suspend: ● All ○ Thread

Condition:

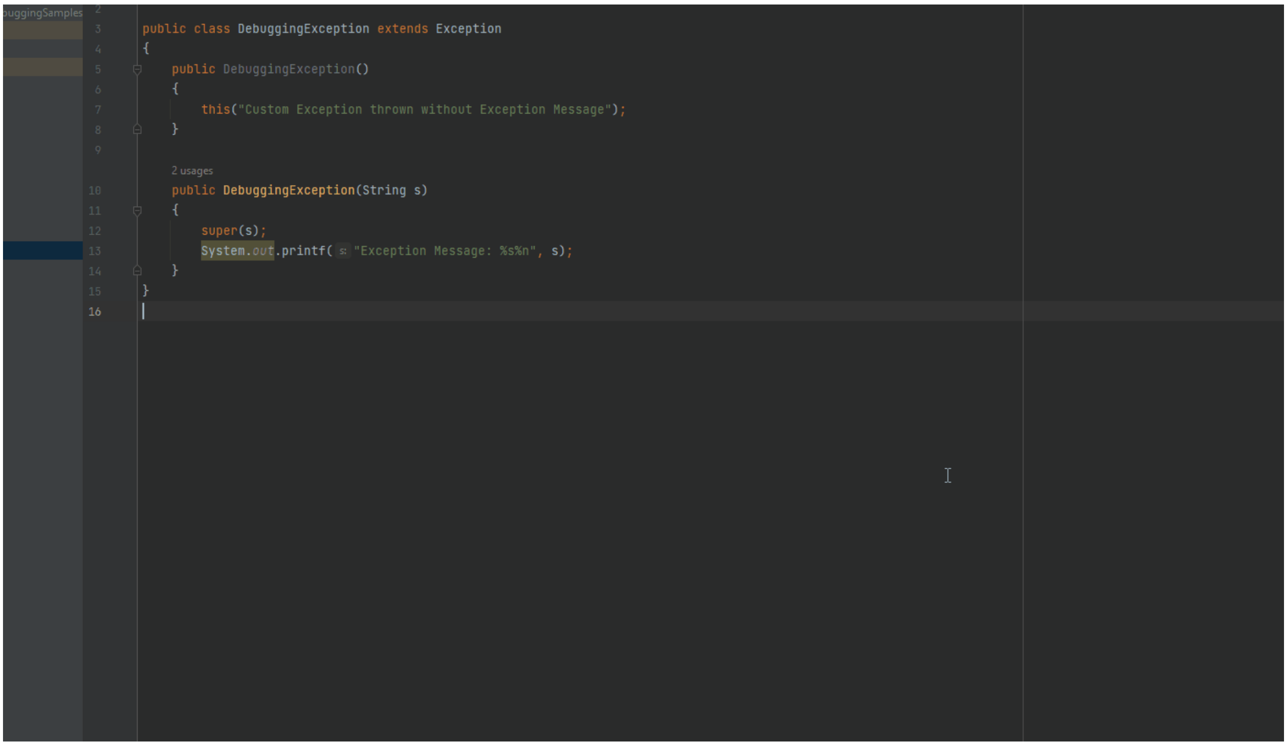
Watch

☐ Field access

☑ Field modification

- Exception Breakpoints: Hold application if specific exception occurs

```
> CTRL + SHIFT + F8  --> View Breakpoints, ALT + Insert (Exception)
```

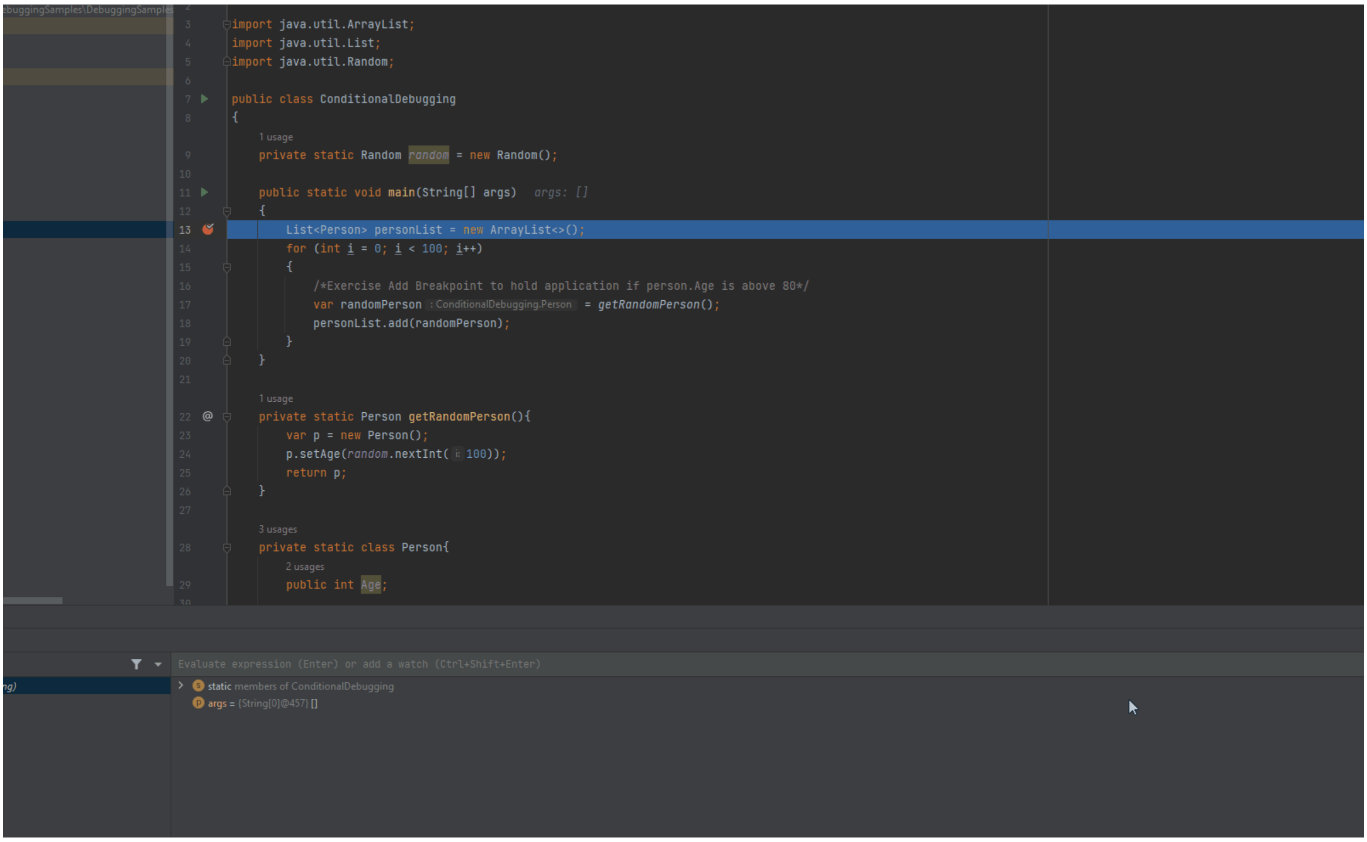


```java
public class DebuggingException extends Exception
{
    public DebuggingException()
    {
        this("Custom Exception thrown without Exception Message");
    }

    2 usages
    public DebuggingException(String s)
    {
        super(s);
        System.out.printf( s: "Exception Message: %s%n", s);
    }
}
```

Alternatively, check "Any Exception" to stop the application if an exception occurs.

Conditional Debugging

Hit breakpoint only if the specified condition is met. Especially useful for loops.

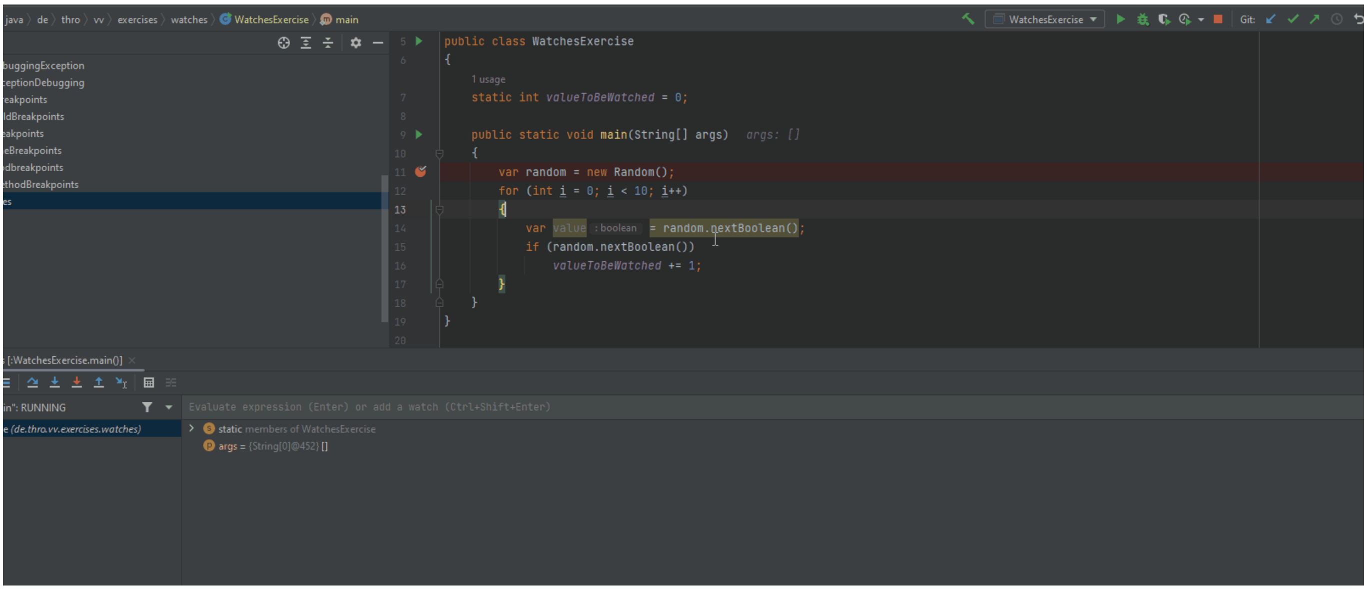


```java
import java.util.ArrayList;
import java.util.List;
import java.util.Random;

public class ConditionalDebugging
{
    1 usage
    private static Random random = new Random();

    public static void main(String[] args)   args: []
    {
        List<Person> personList = new ArrayList<>();
        for (int i = 0; i < 100; i++)
        {
            /*Exercise Add Breakpoint to hold application if person.Age is above 80*/
            var randomPerson : ConditionalDebugging.Person = getRandomPerson();
            personList.add(randomPerson);
        }
    }

    1 usage
    private static Person getRandomPerson(){
        var p = new Person();
        p.setAge(random.nextInt( i: 100));
        return p;
    }

    3 usages
    private static class Person{
        2 usages
        public int Age;
```

```
Evaluate expression (Enter) or add a watch (Ctrl+Shift+Enter)
> static members of ConditionalDebugging
  args = {String[0]@457} []
```

Watches

Add any variable as Watch. This value then remains fixed in the "Debugger window".



```
public class WatchesExercise
{
    static int valueToBeWatched = 0;

    public static void main(String[] args)
    {
        var random = new Random();
        for (int i = 0; i < 10; i++)
        {
            var value = random.nextBoolean();
            if (random.nextBoolean())
                valueToBeWatched += 1;
        }
    }
}
```

```
Evaluate expression (Enter) or add a watch (Ctrl+Shift+Enter)
> static members of WatchesExercise
  args = {String[0]@452} []
```

RemoveOnceHit

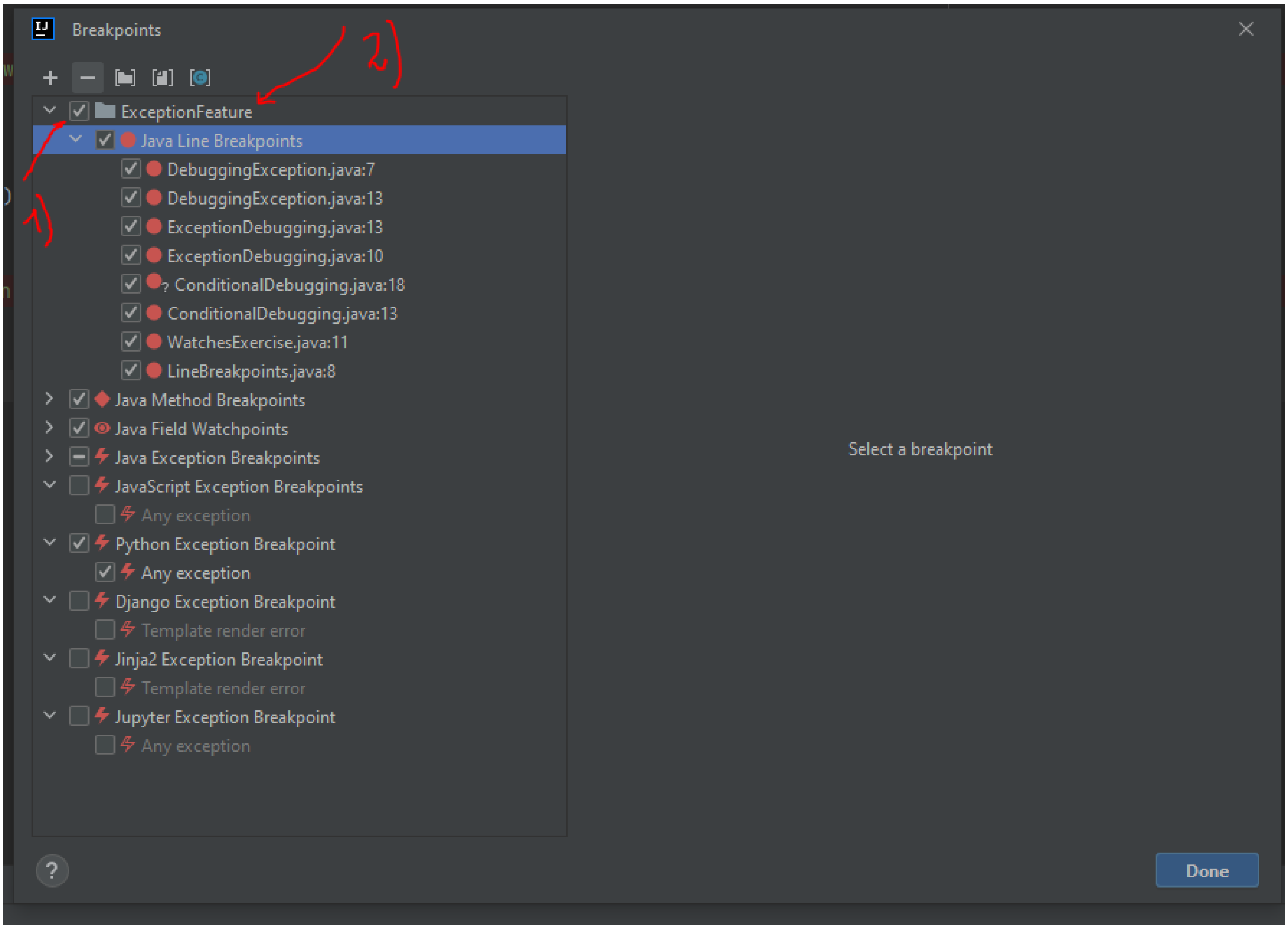
Sometimes a one time breakpoint is useful. For example if the developer has to be sure his method is hit from the correct method. Set the 'Remove Once Hit' Checkbox.

Tipps and Tricks

Named Breakpoints

Sometimes a lot of breakpoints are needed to understand the correct behavior of the application. The developer can quickly lose the overview and gets stuck on "wrong" breakpoints.
The breakpoints can be easily "sorted" here and grouped per feature / package. For example, for the "Exception Feature" under point 1) all breakpoints of this feature can be deactivated with one action. Point 2)

demonstrates the named breakpoints.



☑ ExceptionFeature
- ☑ Java Line Breakpoints
  - ☑ DebuggingException.java:7
  - ☑ DebuggingException.java:13
  - ☑ ExceptionDebugging.java:13
  - ☑ ExceptionDebugging.java:10
  - ☑ ConditionalDebugging.java:18
  - ☑ ConditionalDebugging.java:13
  - ☑ WatchesExercise.java:11
  - ☑ LineBreakpoints.java:8

☑ Java Method Breakpoints

☑ Java Field Watchpoints

⊟ Java Exception Breakpoints

□ JavaScript Exception Breakpoints
- □ Any exception

☑ Python Exception Breakpoint
- ☑ Any exception

□ Django Exception Breakpoint
- □ Template render error

□ Jinja2 Exception Breakpoint
- □ Template render error

□ Jupyter Exception Breakpoint
- □ Any exception

IntelliJ IDEA Shortcuts

> Toggle line/method/field Breakpoint Shortcut --> CTRL + F8
>
> Edit breakpoint properties --> CTRL + SHIFT + F8
>
> Further settings with --> CTRL + SHIFT + F8
>
> Step further --> F8
>
> Run to next breakpoint --> F9

Google Search Engine

- Use Placeholder:

  > Wrong
  >
  > > could not cast value of type DemoHomeClass to DemoClass
  > >
  > > > About 7 results (0,56 seconds)
  >
  > Better:
  >
  > > Could not cast value of type _ to type _

> > > About 1.550.000.000 results (0,51 seconds)
>
> ### Further Improvement:
>
> > site: stackoverflow.com solved Could not cast value of type _ to type _
> >
> > > About 4.420.000 results (1,02 seconds) --> StackOverflow Entry with solved question is first result

- Exclude Search Results:

  > Mustang -Car Ford
  >
  > > Show only horse results

- Must Have Search Terms:

  > Mustang "Ford"
  >
  > > Will search for Cars, excluded horses

- Combinations:

  > -site: udemy "java spring boot"
  >
  > > Show results for Java Spring Boot Courses on Udemy
  >
  > -site: udemy "java RestApi*"
  >
  > > Search on Udemy for Java RestAPI results
  >
  > blog: java spring boot "jpa" 2020..2022
  >
  > > Search for blog entries that can contain java spring boot and must contain "JPA" in the period from 2020 to 2022.

Exercises

Clone this Repository: https://github.com/Thomas-Mildner/DebuggingSamples

- LineBreakpoints: *de.thro.vv.exercises.linebreakpoints.LineBreakpoints --> Main*
- MethodBreakpoints: *de.thro.vv.exercises.methodbreakpoints.MethodBreakpoints --> Main*
- FieldBreakpoints: *de.thro.vv.exercises.fieldbreakpoints.FieldBreakpoints --> Main*
- ConditionalDebugging: *de.thro.vv.exercises.conditionaldebugging.ConditionalDebugging --> Main*
- ExceptionDebugging: *de.thro.vv.exercises.exceptions.ExceptionDebugging --> Main*