Java-News

Beyond Java 8

Tim Hegemann

2018-05-16

Outline

jshell

New Syntax

Java Platform Module System (JPMS)

API Changes

New Release Cycle

- Oracle is proposing to increase the release cadence of Java SE to every six months

Release	Availability	End of Life
8	Mar 2014	>2020
9	Sep 2017	Mar 2018
10 (18.3)	Mar 2018	Sep 2018
11 (18.9)	Sep 2018	TBA (LTS)

jshell

jshell

- Java REPL (Read-Eval-Print Loop)
- Included in JDK 9 and newer
- Tab completion for methods, parameters, variable names and documentation

jshell

- Java REPL (Read-Eval-Print Loop)
- Included in JDK 9 and newer
- Tab completion for methods, parameters, variable names and documentation
- `jshell [options] [load-files]`
- `--class-path` *path* to set the classpath
- Predefined scripts: `DEFAULT`, `JAVASE`, `PRINTING`

jshell Example

```
$ jshell
|  Welcome to JShell -- Version 10.0.1
|  For an introduction type: /help intro

jshell> 1 + 2 * 3
$1 ==> 7

jshell> 6 - $1
$2 ==> -1
```

jshell Example

```
jshell> int fib(int n) {
   ...> int a = 0, b = 1;
   ...> for(int i = 0; i < n; ++i) {
   ...> b = b + a;
   ...> a = b;
   ...> }
   ...> return a;
   ...> }
|  created method fib(int)
```

jshell Example

```
jshell> int fib(int n) {
   ...> int a = 0, b = 1;
   ...> for(int i = 0; i < n; ++i) {
   ...> b = b + a;
   ...> a = b;
   ...> }
   ...> return a;
   ...> }
|  created method fib(int)


jshell> fib(13)
$2 ==> 4096
```

jshell Example

```
jshell> int fib(int n) {
   ...> return fibRec(0, 1, n);
   ...> }
|  modified method fib(int), however, it cannot be invoked until method
↪  fibRec(int,int,int) is declared
```

jshell Example

```
jshell> int fib(int n) {
   ...> return fibRec(0, 1, n);
   ...> }
|  modified method fib(int), however, it cannot be invoked until method
↪   fibRec(int,int,int) is declared

jshell> int fibRec(int a, int b, int n) {
   ...> if (n == 0) return a;
   ...> return fibRec(b, a + b, n - 1);
   ...> }
|  created method fibRec(int,int,int)
```

jshell Example

```
jshell> int fib(int n) {
   ...> return fibRec(0, 1, n);
   ...> }
|  modified method fib(int), however, it cannot be invoked until method
↪  fibRec(int,int,int) is declared

jshell> int fibRec(int a, int b, int n) {
   ...> if (n == 0) return a;
   ...> return fibRec(b, a + b, n - 1);
   ...> }
|  created method fibRec(int,int,int)

jshell> fib(13)
$8 ==> 233
```

New Syntax

Private Interface Methods

```
public interface SortedArray<T extends Comparable<T>> {
    T get(int idx);
    int length();

    default int binSearch(T elem) {
        return binSearchRec(elem, 0, length());
    }

    private int binSearchRec(T elem, int start, int end) {
        if (start > end) return -1; // nothing found
        int mid = start + (end - start) / 2;
        if (get(mid).compareTo(elem) == 0) return mid;
        if (get(mid).compareTo(elem) >= 0)
            return binSearchRec(elem, start, mid);
        return binSearchRec(elem, mid, end);
    }
}
```

Local Variable Type Inference

The traditional way:

```
String words = "a b a a b a b c a b c d";

Collector<String, ?, Map<String, Long>> byOccurrence =
    Collectors.groupingBy(Function.identity(), Collectors.counting());

Map<String, Long> wordFreq = Arrays.stream(words.split(" "))
    .collect(byOccurrence);

for (Map.Entry<String, Long> entry : wordFreq.entrySet())
    if (entry.getValue() > 2)
        System.out.println(entry.getKey() + ": " + entry.getValue());

// prints:
// a: 5
// b: 4
```

Local Variable Type Inference

The Java 10 way:

```
var words = "a b a a b a b c a b c d";

var byOccurrence = Collectors.groupingBy(Function.identity(),
↪   Collectors.counting());

var wordFreq = Arrays.stream(words.split(" ")).collect(byOccurrence);

for (var entry : wordFreq.entrySet())
    if (entry.getValue() > 2)
        System.out.println(entry.getKey() + ": " + entry.getValue());

// prints:
// a: 5
// b: 4
```

Local Variable Type Inference – Non-Denotable Types

The traditional way:

```
jshell> Object user = new Object() {
   ...> int id = 0;
   ...> String name = "root";
   ...> }
user ==> 1@685cb137

jshell> user.id
|  Error:
|  cannot find symbol
|    symbol:   variable id
|  user.id
|  ^-----^
```

Local Variable Type Inference – Non-Denotable Types

The Java 10 way:

```
jshell> var user = new Object() {
   ...> int id = 0;
   ...> String name = "root";
   ...> }
user ==> $1@6093dd95

jshell> user.id
$4 ==> 0

jshell> user.name
$5 ==> "root"
```

Java Platform Module System (JPMS)

Project Jigsaw

- Originally proposed in 2005 for Java 7

Project Jigsaw

- Originally proposed in 2005 for Java 7
- A Java module is a uniquely named, reusable group of related packages, as well as resources and a module descriptor.

Project Jigsaw

- Originally proposed in 2005 for Java 7
- A Java module is a uniquely named, reusable group of related packages, as well as resources and a module descriptor.
- Goals:
  - Reliable configuration
  - Strong encapsulation
  - Scalable Java platform
  - Greater platform integrity

Module Declarations

```
$ tree .
.
└── src
    └── de.uniwue.rudi
        ├── de
        │   └── uniwue
        │       └── rudi
        │           └── Main.java
        └── module-info.java
```

Module Declarations

```
$ tree .
.
└── src
    └── de.uniwue.rudi
        ├── de
        │   └── uniwue
        │       └── rudi
        │           └── Main.java
        └── module-info.java
```

```
$ cat src/de.uniwue.rudi/module-info.java
module de.uniwue.rudi {
}
```

More Module Declarations

```
module de.uniwue.rudi {
    // public members of package rudi are visivble to other modules.
    exports de.uniwue.rudi;

    // qualified export - visability is limited to packages marie and
    ↪  herbert.
    exports de.uniwue.rudi.internal to de.uniwue.marie,
    ↪  de.uniwue.herbert;

    // module dependency to module ingrid.
    requires de.uniwue.ingrid;

    // modules that depend on rudi implicitly depend on karl too.
    requires transitive de.uniwue.karl;

    // further reading: uses, provides, opens, ...
}
```

API Changes

Enhanced Deprecation (since 9)

```java
/**
 * Not implemented, does nothing.
 *
 * @deprecated
 * This method was intended to control instruction tracing.
 * It has been superseded by JVM-specific tracing mechanisms.
 * This method is subject to removal in a future version of Java SE.
 *
 * @param on ignored
 */
@Deprecated(since="9", forRemoval=true)
public void traceInstructions(boolean on) { }
```

Long Overdue Stuff (since 9)

In class `InputStream`

- `byte[] readAllBytes()`

Long Overdue Stuff (since 9)

In class `InputStream`

- `byte[] readAllBytes()`

In class `java.time.Duration`

- `long toMinutesPart()`

```
jshell> var d = java.time.Duration.ofSeconds(12345);
d ==> PT3H25M45S
jshell> d.toMinutes()
$1 ==> 205
jshell> d.toMinutesPart()
$2 ==> 25
jshell> d.toSecondsPart()
$3 ==> 45
```

Long Overdue Stuff (since 9)

In class `InputStream`

- `byte[] readAllBytes()`

In class `java.time.Duration`

- `long toMinutesPart()`

In class `Stream<T>`

- `Stream<T> takeWhile(Predicate<? super T> p)`
- `Stream<T> dropWhile(Predicate<? super T> p)`

Optional-like (since 9)

In class `Optional<T>`

- `Stream<T> stream()`

```
// flatten a stream of optionals discarding empty elements
Stream<Optional<T>> os = ...
Stream<T> s = os.flatMap(Optional::stream);
```

Optional-like (since 9)

In class `Optional<T>`

- `Stream<T> stream()`
- `T orElseThrow()`

Optional-like (since 9)

In class `Optional<T>`

- `Stream<T> stream()`
- `T orElseThrow()`

In class `Objects`

- `static <T> T requireNonNullElse(T obj, T orElse)`

Array Comparison (since 9)

In class `Arrays`

- `<T extends Comparable<? super T>> int compare(T[] a, T[] b)`
- `int compareUnsigned(int[] a, int[] b)`
- `int mismatch(Object[] a, Object[] b)`

Unmodifiable Collection Factories

```
// since 9
var original = new ArrayList<>(List.of(1, 2, 3, 4, 5, 6, 7));

var view = Collections.unmodifiableList(original);

// since 10
var copy = List.copyOf(original);

original.remove(3);    // removes the 4 :(

System.out.println(view.contains(4));    // false
System.out.println(copy.contains(4));    // true
```

Unmodifiable Collection Factories

```
// since 9
var original = new ArrayList<>(List.of(1, 2, 3, 4, 5, 6, 7));

var view = Collections.unmodifiableList(original);

// since 10
var copy = List.copyOf(original);

original.remove(3);    // removes the 4 :(

System.out.println(view.contains(4));    // false
System.out.println(copy.contains(4));    // true
```

See also `Collectors.toUnmodifiableList()`.

See also `Map.ofEntries(Map.Entry<K, V>[])` and `Map.entry(K, V)`.