Groovy 3 New Features

Lambdas and More

Contact Info

Ken Kousen

Kousen IT, Inc.

ken.kousen@kousenit.com

http://www.kousenit.com

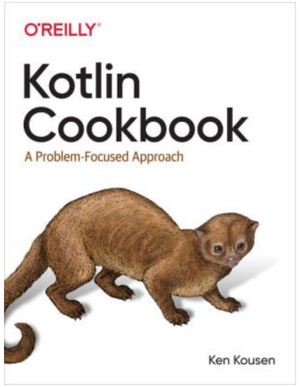
http://kousenit.org (blog)

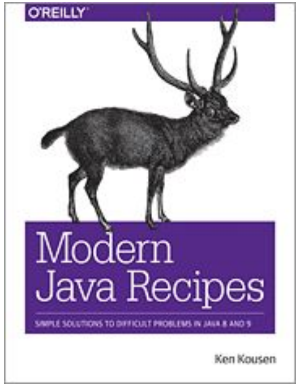
@kenkousen (twitter)

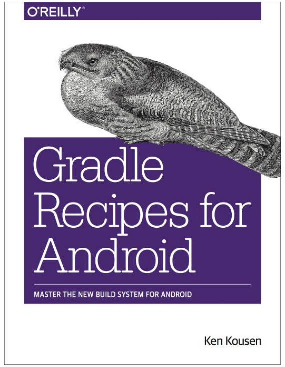
https://kenkousen.substack.com (newsletter)

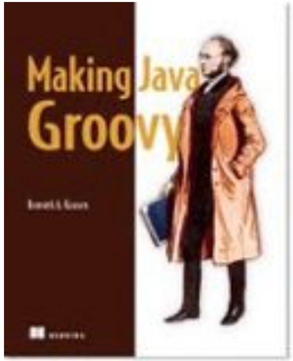
All demos at:

https://github.com/kousen/groovy_3_examples






O'REILLY
Gradle Recipes for Android
MASTER THE NEW BUILD SYSTEM FOR ANDROID
Ken Kousen



Release notes for Groovy 3.0

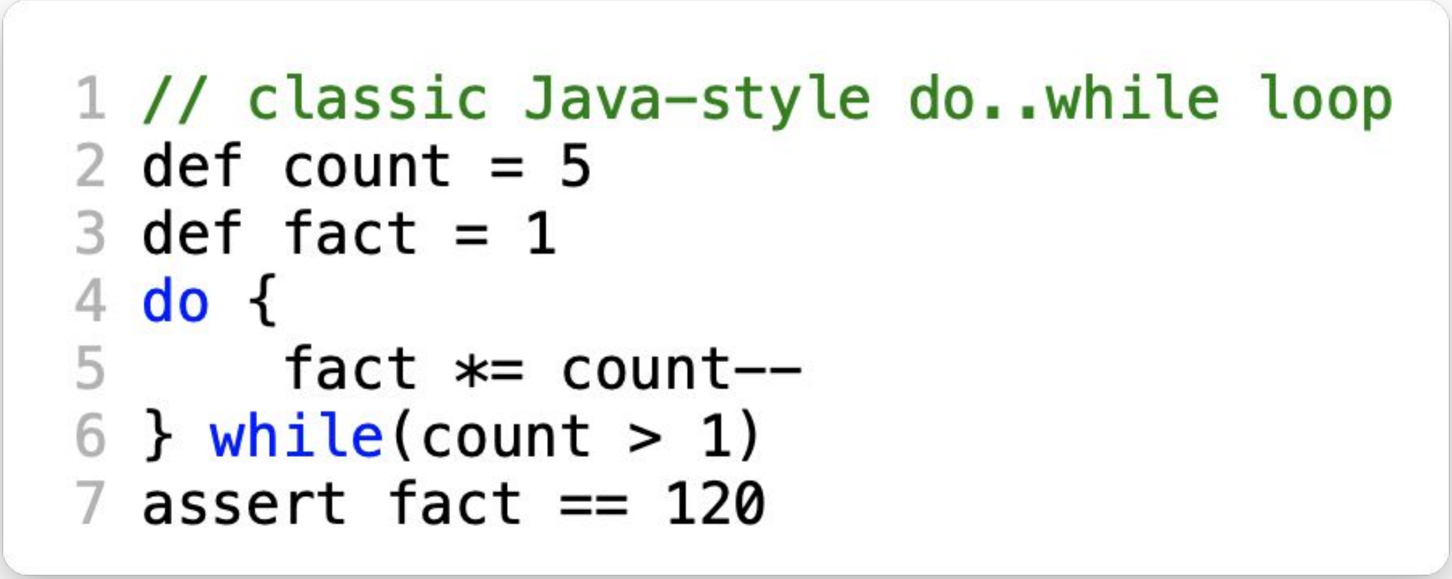
http://groovy-lang.org/releasenotes/groovy-3.0.html

Parrot Parser

Flexible Parrot parser

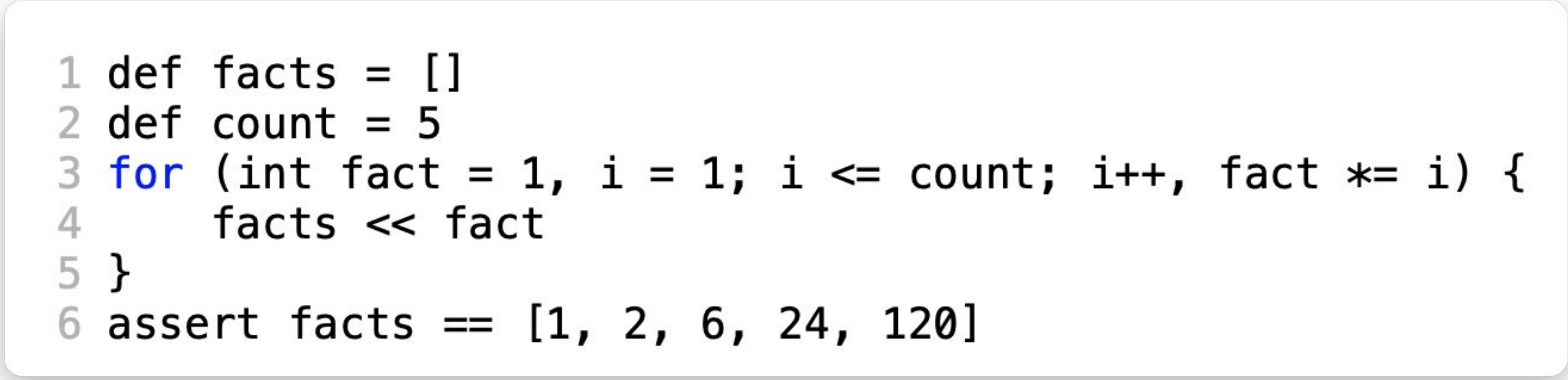
- do-while loops
- enhanced for loops
- lambdas, method references, constructor references
- try-with-resources
- Java array initializers
- default methods in interfaces

do-while loop



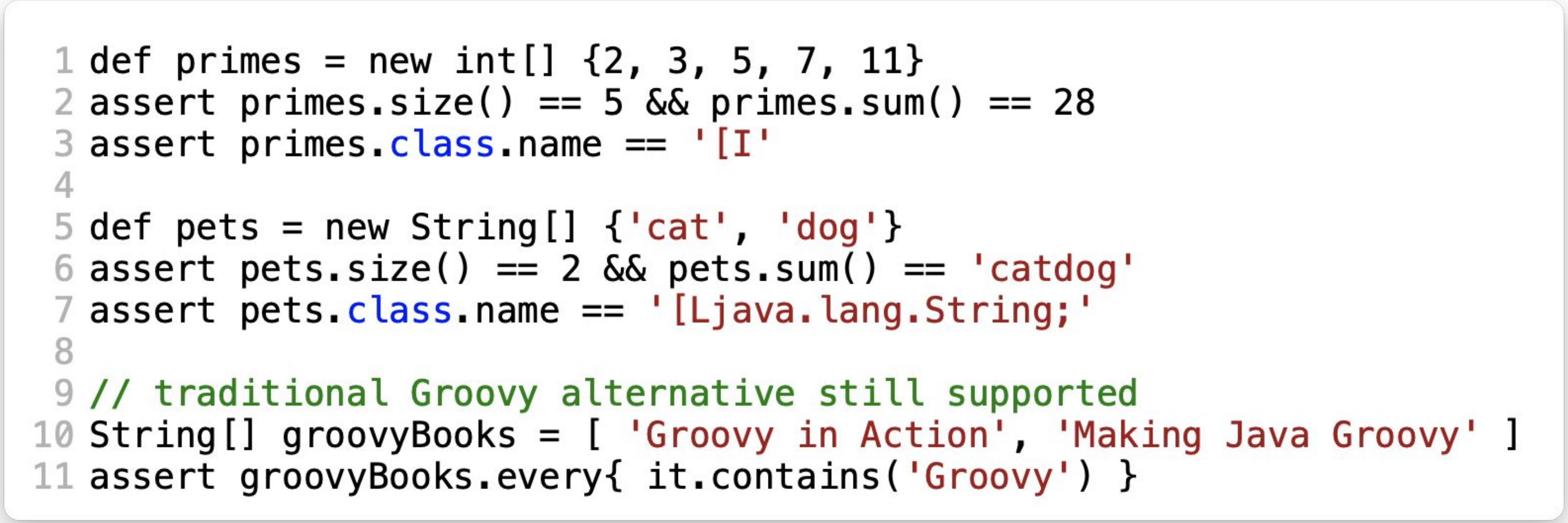
```groovy
// classic Java-style do..while loop
def count = 5
def fact = 1
do {
    fact *= count--
} while(count > 1)
assert fact == 120
```

Old-style Java for loop



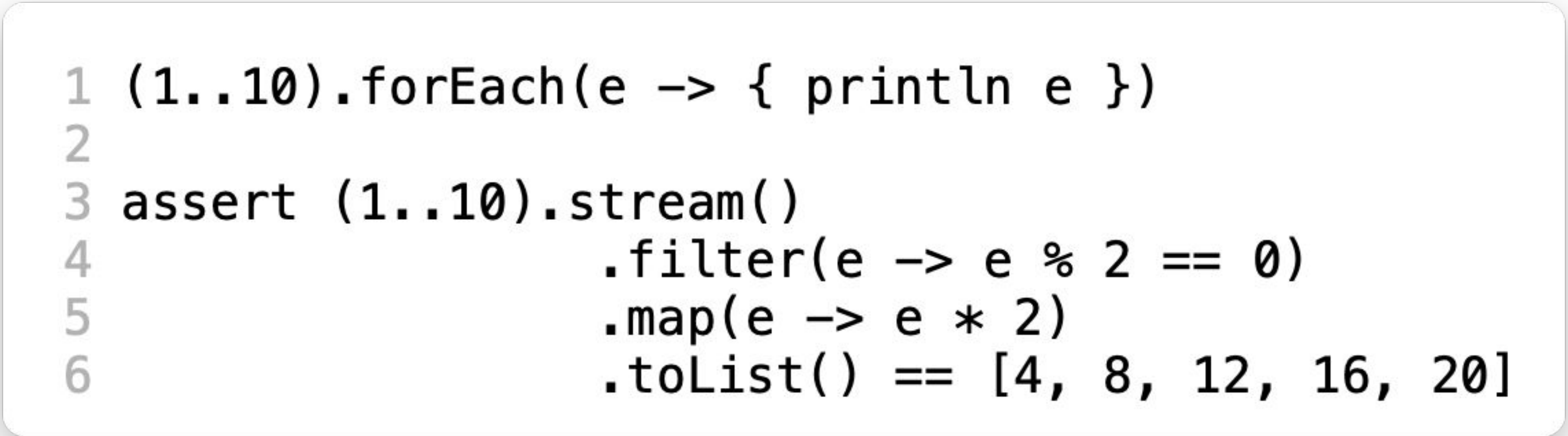
```
def facts = []
def count = 5
for (int fact = 1, i = 1; i <= count; i++, fact *= i) {
    facts << fact
}
assert facts == [1, 2, 6, 24, 120]
```

Java-style array initialization



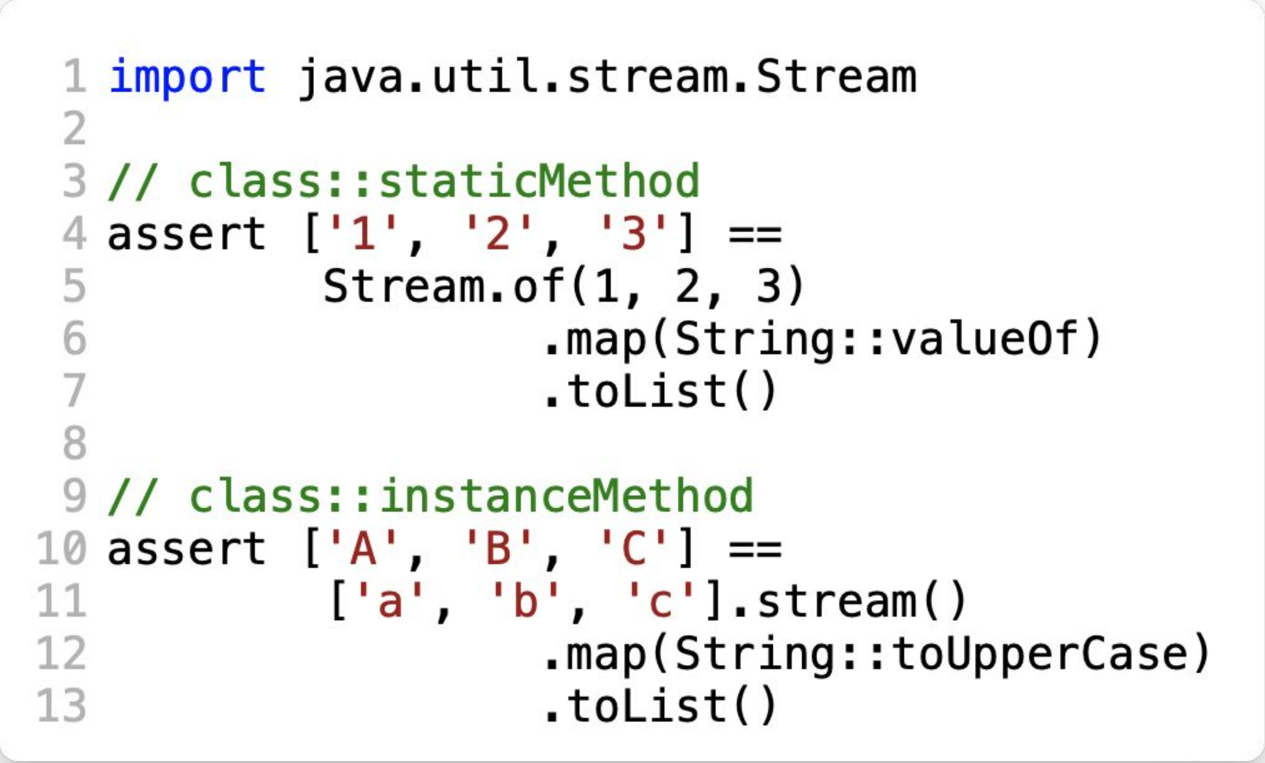
```groovy
def primes = new int[] {2, 3, 5, 7, 11}
assert primes.size() == 5 && primes.sum() == 28
assert primes.class.name == '[I'

def pets = new String[] {'cat', 'dog'}
assert pets.size() == 2 && pets.sum() == 'catdog'
assert pets.class.name == '[Ljava.lang.String;'

// traditional Groovy alternative still supported
String[] groovyBooks = [ 'Groovy in Action', 'Making Java Groovy' ]
assert groovyBooks.every{ it.contains('Groovy') }
```

Java-style lambda syntax



```
(1..10).forEach(e -> { println e })

assert (1..10).stream()
                .filter(e -> e % 2 == 0)
                .map(e -> e * 2)
                .toList() == [4, 8, 12, 16, 20]
```

Methd references



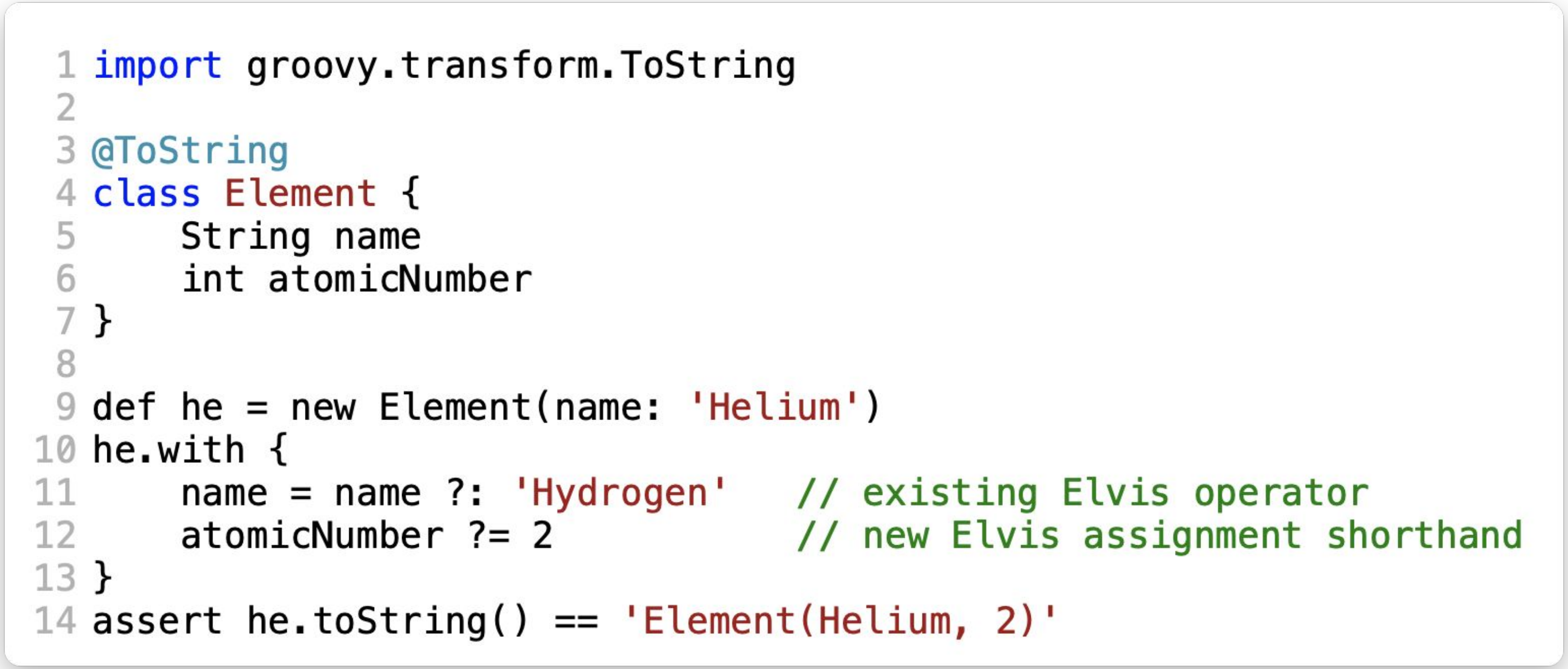
```
import java.util.stream.Stream

// class::staticMethod
assert ['1', '2', '3'] ==
        Stream.of(1, 2, 3)
                .map(String::valueOf)
                .toList()

// class::instanceMethod
assert ['A', 'B', 'C'] ==
        ['a', 'b', 'c'].stream()
                .map(String::toUpperCase)
                .toList()
```

Operators

!in and !instanceof

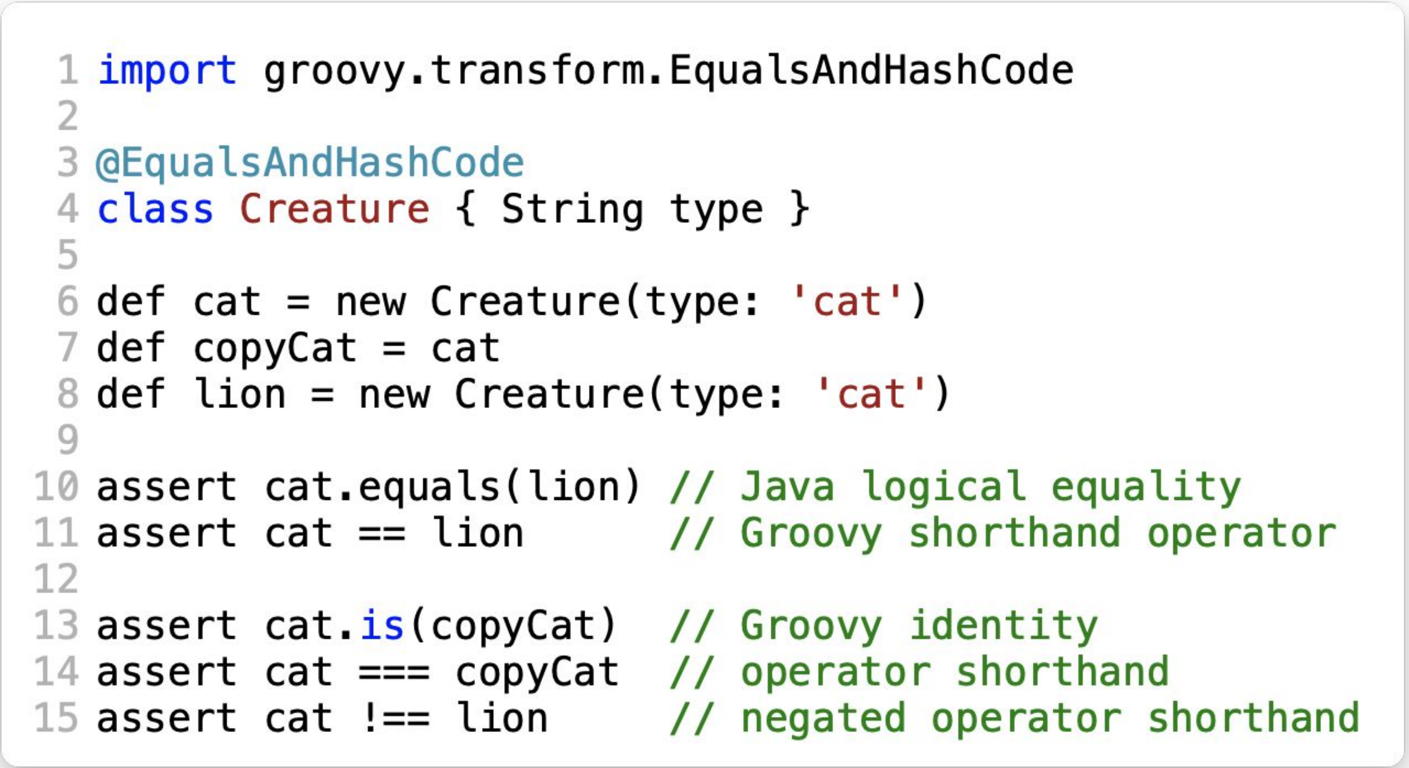
```
assert 42 !instanceof LocalDate

assert 4 !in [1, 3, 5, 7]
```

Elvis assignment



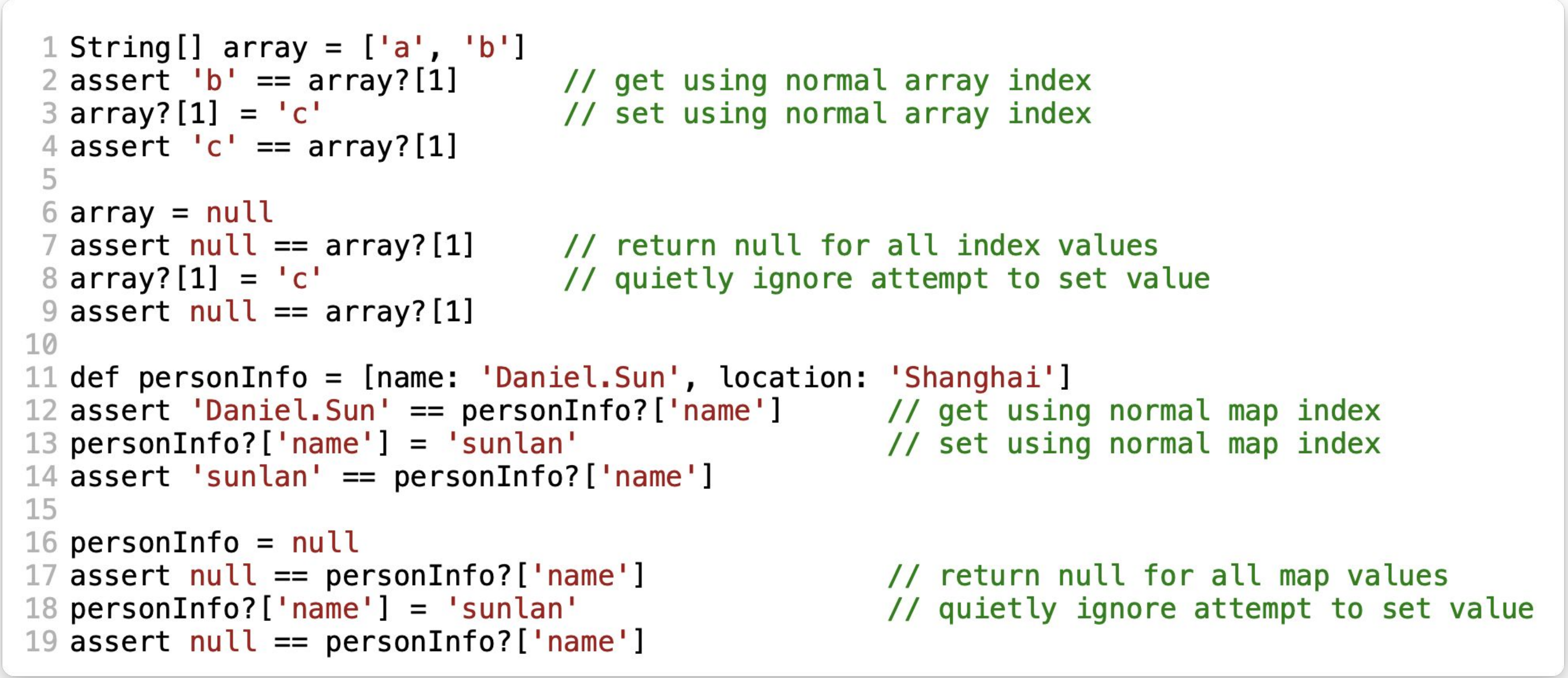
```groovy
import groovy.transform.ToString

@ToString
class Element {
    String name
    int atomicNumber
}

def he = new Element(name: 'Helium')
he.with {
    name = name ?: 'Hydrogen'   // existing Elvis operator
    atomicNumber ?= 2           // new Elvis assignment shorthand
}
assert he.toString() == 'Element(Helium, 2)'
```

Identity comparison operators



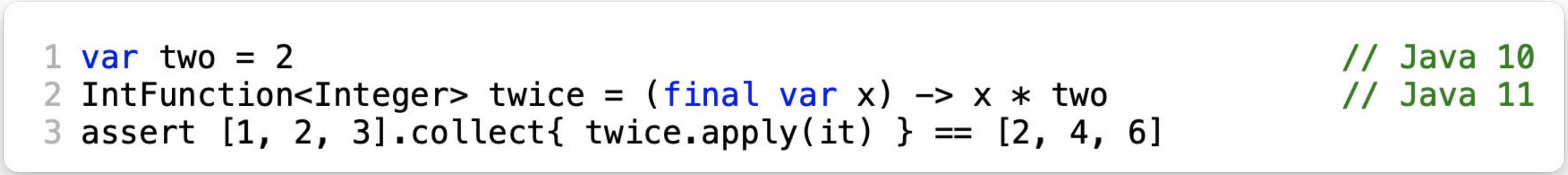
```groovy
import groovy.transform.EqualsAndHashCode

@EqualsAndHashCode
class Creature { String type }

def cat = new Creature(type: 'cat')
def copyCat = cat
def lion = new Creature(type: 'cat')

assert cat.equals(lion) // Java logical equality
assert cat == lion      // Groovy shorthand operator

assert cat.is(copyCat)  // Groovy identity
assert cat === copyCat  // operator shorthand
assert cat !== lion     // negated operator shorthand
```

Safe indexing



```
String[] array = ['a', 'b']
assert 'b' == array?[1]      // get using normal array index
array?[1] = 'c'              // set using normal array index
assert 'c' == array?[1]

array = null
assert null == array?[1]     // return null for all index values
array?[1] = 'c'              // quietly ignore attempt to set value
assert null == array?[1]

def personInfo = [name: 'Daniel.Sun', location: 'Shanghai']
assert 'Daniel.Sun' == personInfo?['name']      // get using normal map index
personInfo?['name'] = 'sunlan'                  // set using normal map index
assert 'sunlan' == personInfo?['name']

personInfo = null
assert null == personInfo?['name']              // return null for all map values
personInfo?['name'] = 'sunlan'                  // quietly ignore attempt to set value
assert null == personInfo?['name']
```

var reserved type



```
var two = 2                                                      // Java 10
IntFunction<Integer> twice = (final var x) -> x * two            // Java 11
assert [1, 2, 3].collect{ twice.apply(it) } == [2, 4, 6]
```

Interface default methods

- Can add default methods to interfaces, as in Java 8+
- Still probably better to use traits

New GDK methods

- `average()` on arrays and iterables
- `takeBetween()` on `String` (including Groovy strings) and `CharSequence`
- `shuffle()` and `shuffled()` on arrays and iterables
- `minus()` on `LocalDate`

AST transforms

- @NullCheck