



Java™

VS

8

11

11

- ✓ Java 11, 25 Eylül 2018'de piyasada herkesin kullanımına sunuldu. Java SE platform sürüm 11'in açık kaynaklı bir referans uygulamasıdır. Java 11, Java 8'in yayınlanmasından dört yıl sonra piyasaya sürüldü.
- ✓ Java 11, daha fazla işlevsellik sağlamak için yeni özelliklerle birlikte gelir. Bu iki sürüm arasındaki dört buçuk yılda eklenen özellikleri ele aldım.

# Modularization

- ✓ Modülerlik, uygulamamızı kolayca test etmemize ve ayrıca kodumuzun hangi bölümlerinin herkese açık olacağını yönetmemize olanak tanır.
- ✓ Modülerlik özelliği Java 8 de mevcut değildir,Java 11 de mevcuttur.
- ✓ Modüller arası erişilebilirlik ise modül ana dizini seviyesinde bulunan *module-info.java* sınıfı içerisinde tanımlanır.
- ✓ **sql** modülümüz **crud** isimli paketimiz olsun. **Sql** modülü içerisindeki **crud** paketini erişilebilir yapmak istiyorsak *module-info.java* sınıfı aşağıdaki gibi olmalıdır.

```
module sql {  
  exports com.demo.sql.crud;  
}
```

# Var Keyword

- ✓ Java 10 da tanıtılıp Java 11 de geliştirilen var keywordü, standart kodlamayı azaltmamıza yardımcı olan geliştirici dostu bir anahtar sözcüktür.
- ✓ Artık parametre tanımlarken parametre tipi belirtmeden, sadece `var` keyword ile tanımlama yapabiliyoruz.

```
var testWord = "Hello, Test";  
var age = 22;  
var isAvailable = true;  
for(var i = 0; i < 10; i++)  
for(var user: users)
```

# Local-Variable Syntax for Lambda Parameters

- ✓ Java 11'de lambda parametrelerinde local syntax ( *var* anahtar sözcüğü) kullanma desteği eklendi.
- ✓ Lambda tanımlamaları daha önce explicitly ve implicitly olarak 2 şekilde yapılabilmekteydi. Artık implicitly olarak ``var`` kullanarak tanımlayabiliyoruz.
- ✓ (Item a, int b) -> a.create(b); // Explicitly  
(a,b) -> a.create(b); // Implicitly  
(var a, var b) -> a.create(b); // Implicitly with var

# New String Methods

- ✓ Java 11 , *String* sınıfına birkaç yeni metod ekledi :isBlank, lines, strip, stripLeading, stripTrailing ve repeat.

```
var myName = " \u205F ";  
System.out.println(myName.isBlank()); // içeriğe bakar  
System.out.println(myName.isEmpty()); // uzunluğa bakar  
  
var myName1 = "  Bella Brown  ";  
System.out.println(myName1.strip().length());  
var myName2 = "  Bella Brown";  
System.out.println(myName2.stripLeading().length());  
System.out.println(myName2.stripTrailing().length());
```

# Launching Single Source File Programs

- ✓ Java 11'den önce main metoduna sahip bir java class'ını console üzerinde çalıştırabilmek için ilk önce bu sınıfı compile etmek daha sonra da çalıştırmak gerekiyordu.

```
$javac -d classes Test.java  
$java cp classes Test
```

- ✓ Java 11 ile artık Test sınıfımızı compile etmeden aşağıdaki gibi tek satırda çalıştırabiliyoruz.

```
$java Test.java
```

# Multi-jar sürümleri

- ✓ Multi-jar lar Java 8 istemcileri için destek ve işlevselliği korurken, kodu Java 11'in en son özellikleriyle kullanmaya devam etmenin bir yolunu sağlar.



# Yeni Dosya Yöntemleri

✓ *Files* sınıfından yeni *readString* ve *writeString* metodları eklendi.

```
Path filePath = Files.writeString(Files.createTempFile(tempDir, "demo", ".txt"),  
"Sample text");  
String fileContent = Files.readString(filePath);
```

# Collection to an Array

- ✓ *Java.util.Collection* arayüzüne [toArray\(IntFunction\)](#) eklendi . Koleksiyon öğelerinin istenen çalışma zamanı türünde yeni oluşturulan bir diziye aktarılmasını sağlayan bir metod.
- ✓ Bu, bir koleksiyondan doğru türde bir dizi oluşturmayı kolaylaştırır:

// Java 11

```
List<String> list = Arrays.asList("foo", "bar", "baz");  
String[] array = list.toArray(String[]::new);
```

// The above is equivalent to:

```
String[] array2 = list.toArray(new String[0]);
```

# JShell

- ✓ Jshell sayesinde artık komut satırı üzerinde java kodu yazıp çalıştırabiliyoruz. Bu sayede kod yazdığımız dosyaları derleme ve sonrasında çalıştırma süreci yerine, hemen terminal üzerinden yazıp kodumuzun çıktısını görebiliyoruz.

```
jshell> double volume(double radius) { ...> return 4.0 / 3.0 * PI *  
cube(radius); ...> }
```

| created method volume(double), however, it cannot be invoked until variable PI, and method cube(double) are declared

## Optional - Yenilikler

- ✓ **ifPresentOrElse():** Optional.isPresent() metoduna göre if-else yazmak yerine tek metod ile if ve else koşullarındaki çalışmasını istediğimiz metodları tanımlayabiliyoruz.



```
if (optionalPrice.isPresent()) {  
    updatePrice(optionalPrice.get());  
} else {  
    finishOperation();  
}  
-----  
optionalPrice.ifPresentOrElse(this::updatePrice, this::finishOperation);
```

- ✓ **or():** Optional dönen bir metod çağırıldığında, sonucun Optional.empty() olması durumuna göre davranış tanımlanabilmesini sağlamaktadır.

```
Optional<Person> optionalPerson = findPersonById(1L).or(() -> Optional.of(new  
Person()));
```

# Kullanımdan Kaldırmalar

- ✓ Java tarayıcı eklentileri desteğinin kaldırılması nedeniyle, Applet API kullanımdan kaldırılmıştır (ancak henüz kaldırılmamıştır).
- ✓ CMS, yani Concurrent Mark Sweep Garbage Collector şu anda desteklenmemektedir. Kaldırılmasının nedeni, GCC üssünün bakım yükünü azaltmaktır.
- ✓ ECMAScript API'sinde yapılan hızlı değişikliklerle, Nashorn JavaScript motorunun bakım miktarı nedeniyle kaldırıldı.
- ✓ Java EE Platform Spesifikasyonu ile çakışması nedeniyle, Java EE modülü kullanımdan kaldırıldı ve kaldırıldı.

	 Java 8 (LTS)	 Java 11 (LTS)
Major Features	<b>Lambdas</b> <code>s -&gt; do(s)</code> Eg. <code>arr.forEach((String s) -&gt;                      System.out.print(s));  arr.forEach(s-&gt;                      System.out.print(s));  arr.forEach(                      System.out::print);</code>	<b>Var on Lambdas</b> <code>var s -&gt; do(s)</code>
APIs Features	<b><u>Lambdas compatibility</u></b> <code>arrays.forEach(s -&gt; print(s))</code> Consumer, Function, Supplier, Predicate, Operator interfaces  <b><u>Stream, Parallel Stream</u></b> <code>myList.stream().filter(s -&gt;  s.startsWith("B"))  .map(String::toLowerCase)  .sorted()  .forEach(System.out::println)</code> <b><u>String</u></b> <code>String.join("B", list);</code> <b><u>Optional</u></b> <code>Optional&lt;String&gt; optional =  Optional.of(str);</code>	<a href="https://www.knowps.com">https://www.knowps.com</a>  <b><u>String</u></b> <code>String.strip()  String.isBlank()  String.readString(path file)</code>  <b><u>Stream, Parallel Stream</u></b> <code>lines.stream()  .filter(Predicate.not(String::isBlank))</code>  <b><u>Optional</u></b> <code>Optional.of(obj).isEmpty()</code>
JVM/Lang.	<b>Interface:</b> default method and static	<b>GC</b> <ul style="list-style-type: none"> <li>Epsilon GC : the GC no-opt</li> </ul> <b>JAVA EE:</b> deprecated <code>java HelloWorld.java</code> (without compilation) and shebang <b>Interface:</b> nested class