Java 14: what's new

*Release was on the 17th of March 2020

Changes in Java distribution

Lot of things changed after 16/04/19.
How it affects me?

What's changed?

Oracle always supported **OpenJDK** and **OracleJDK**:

- both were **free**
- had some functional differences

License agreement has changed for **OracleJDK** starting from Java SE 8 updates after January 2019:

You may not: use the Programs for any data processing or any commercial, production, or internal business purposes other than developing, testing, prototyping, and demonstrating your Application;

What's changed?

OpenJDK

Java Development Kit JDK

GNU GPL v2 + CPE

Oracle Binary Code License Agreement

Does this affects JRE?

YES

Both JDK and JRE are affected

Any difference?

OpenJDK

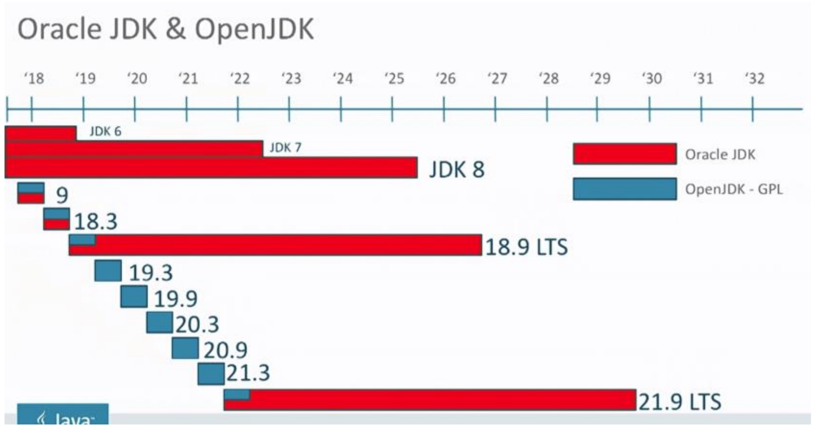
Java Development Kit
JDK

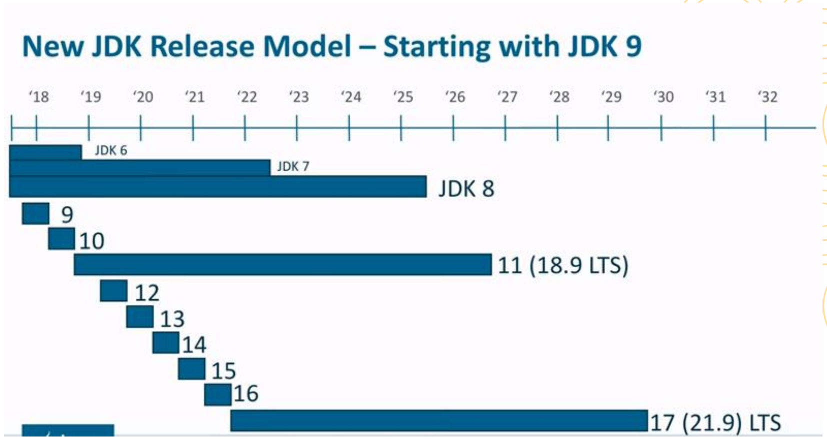
no real technical difference
build process for the Oracle JDK is based on OpenJDK.

- 👍 Can be used in PROD for free
- Performance is good enough for PROD
- 6 months of support

- Should pay for commercial license
- 👍 Performance is better
- 👍 LTS (5 years)

Release cycle





Some more bureocracy

How proposal for features work?

How to make a proposal?

JEP

JDK **E**nhancement **P**roposal

(informal JCP)

JSR

Java **S**pecification **R**equest

Commitee

Best representatives from the best companies. Rotation each year.

JCP

Java **C**ommunity **P**rocess

JCP flow

1

Invitation

JSR reviewed, JSR approved, Expert Group Formed

2

Community review of the draft

1-3 months of community review

3

Public review and approval

Specification, Tech compatibility kit, reference implementation release

4

Maintenance

Maintenance Lead review, executive committee vote

https://www.jcp.org/en/procedures/overview

JEP-12: Preview features

A *preview feature* is a new feature of Java language, that is fully specified, fully implemented, and yet impermanent. Provokes developer feedback based on real world use. Can become permanent in the future.

For example in Java 11 we had experimental options:

```
-XX:+UnlockExperimentalVMOptions -XX:+UseZGC
```

Finally! Let’s move to features

Brief overview

1

Language features

Features that 99% of you can and will use

JEP-359

Records

JEP-359: Records (aka data classes)

- Preview feature
  (-enable-preview -release 14)
- Part of Valhalla Project
- New syntax in language
- Future: https://openjdk.java.net/jeps/384

JEP-359: Records (aka data classes)

```java
final class Person {
    private final String name;
    private final String surname;

    public Person(String name, String surname) {
        this.name = name;
        this.surname = surname;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Person person = (Person) o;
        return Objects.equals(name, person.name) &&
                Objects.equals(surname, person.surname);
    }

    @Override
    public int hashCode() {
        return Objects.hash(name, surname);
    }

    @Override
    public String toString() {
        return "";
    }

    public String getSurname() {
        return surname;
    }

    public String getName() {
        return name;
    }
}
```

JEP-359: Records (aka data classes)

```
public record Person(String name, String surname){}
```

- All needed methods are generated by compiler
- Final, cannot be abstract

Project Lombok

Scala

Kotlin

JEP-359: Records (aka data classes)

```
public final class Person extends java.lang.Record {
    private final java.lang.String name;
    private final String surname;

    public Person(java.lang.String name, String surname) { /* compiled code */ }

    public java.lang.String toString() { /* compiled code */ }

    public final int hashCode() { /* compiled code */ }

    public final boolean equals(java.lang.Object o) { /* compiled code */ }

    public java.lang.String name() { /* compiled code */ }

    public String surname() { /* compiled code */ }
}
```

JEP-305

Pattern matching

JEP-359: Pattern matching for instanceof

- Preview feature
  (-enable-preview -release 14)
- Future: http://openjdk.java.net/jeps/375
- New syntax in language

JEP-359: Pattern matching for instanceof

Before Java 14

```
Object obj = "Ivan loves instanceof";

if (obj instanceof String) {
    String s = (String) obj;
    System.out.println(s.length());
}
```

JEP-359: Pattern matching for instanceof

After Java 14

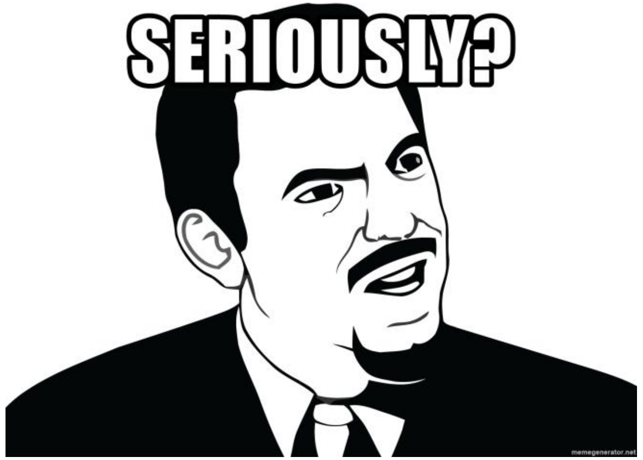
```java
Object obj = "Ivan loves instanceof";

if (obj instanceof String s){
    System.out.println(s.length());
}
```

JEP-358

Helpful NPE

JEP-358: Helpful NullPointerExceptions

- Java removed NPE? – of course **NO!**
- Simply enhanced logging!



JEP-358: Helpful NullPointerExceptions

Before Java 14

```
DemoClass superNpeFeature = init();
superNpeFeature.getData().getSpecialInfo().getName();
```

```
Exception in thread "main" java.lang.NullPointerException at
Main.main(Main.java:13)
```

JEP-358: Helpful NullPointerExceptions

After Java 14

```
DemoClass superNpeFeature = init();
superNpeFeature.getData().getSpecialInfo().getName();
```

```
Exception in thread "main" java.lang.NullPointerException: Cannot invoke
“SpecialInfo().getName()" because the return value of “Data().getSpecialInfo()" is
null at Main.main(Main.java:12)
```

JEP-361

Switch enabled by the default

JEP-361: Switch Expressions (Standard)

- Switch already worked in Java12 and Java13
- Not a preview feature anymore
- Now is a part of permanent standard
- New syntax and new ‘yield’ keyword

JEP-361: Switch Expressions

Before Java 14

```
int numLetters;
switch(day){
   case MONDAY:
   case FRIDAY:
   case SUNDAY:
       numLetters=6;
        break;
   case TUESDAY:
        numLetters=7;
        break;
   case THURSDAY:
   case SATURDAY:
        numLetters=8;
        break;
   case WEDNESDAY:
        numLetters=9;
        break;
   default:
        throw new IllegalStateException("How?: " + day);
}
```

JEP-361: Switch Expressions

After Java 14

```
int numLetters = switch (day) {
        case MONDAY, FRIDAY, SUNDAY -> 6;
        case TUESDAY                -> 7;
        case THURSDAY, SATURDAY     -> 8;
        case WEDNESDAY              -> 9;
};
```

JEP-361: Switch Expressions

Also: yield keyword

```
int j = switch(day) {
        case MONDAY -> 0;
        case TUESDAY -> 1;
        default -> {
            int k=day.toString().length();
            int result=f(k);
             yield result; // <<<<
        }
};
```

JEP-368

Text blocks being changed again!

JEP-368: Text Blocks (Second Preview)

- Still a preview feature
- Added some changes to **JEP-355**: Text Blocks (Preview)
- Will be different from the Java 13 version

JEP-368: Text Blocks (Second Preview)

Before Java 13

```
String html = "<html>\n" +
              "    <body>\n" +
              "        <p>Hello, world</p>\n" +
              "    </body>\n" +
              "</html>\n";
```

JEP-368: Text Blocks (Second Preview)

In Java 13

```
String html = """
              <html>
                  <body>
                      <p>Hello World</p>
                  </body>
              </html>
              """;
```

JEP-368: Text Blocks (Second Preview)

After Java 14

```
String text = """
              Here I want spaces in the end not being trimmed:   \
              Here I will use the second version of this feature:    \s
              This will work? Nice!
              """;
```

- \ - added for newline
- **\s** – added for explicit space

2

Tools and libs

Features that 10% of you will use,
speedrun

JEP-343

Incubator

JEP-343: Packaging Tool (Incubator)

- Simple packaging tool
- Similar to JavaFX: javapackager
- Supports natural packaging formats
- tools/jpackage (jdk.incubator.jpackage)

JEP-343: Packaging Tool (Incubator)

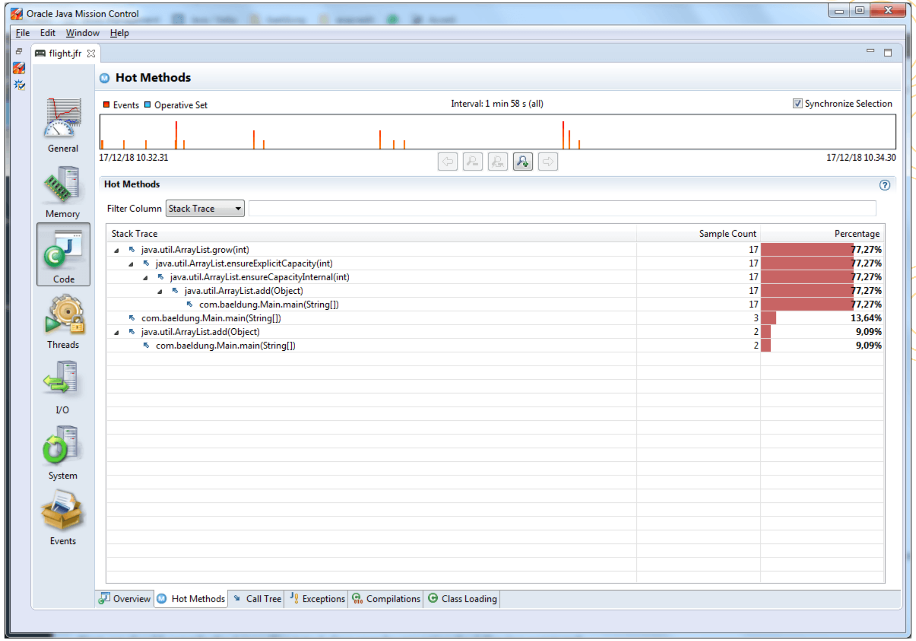
- Example of usage:
  ```
  $ jpackage --name myapp --input lib --main-jar main.jar --type exe
  ```
- msi/exe on Windows
- pkg/dmg on macOS
- deb/rpm on Linux.

JEP-349

Java Flight Recorder

JEP 349: JFR Event Streaming

- JFR – monitoring tool that can be used as plugin in Java Mission Control
- Continuous monitoring with 1% overhead
- I love good old jconsole, never used JMC/JFR



Hot Methods

Events Operative Set Interval: 1 min 58 s (all) ☑ Synchronize Selection

17/12/18 10.32.31 — 17/12/18 10.34.30

Hot Methods

Filter Column: Stack Trace

Stack Trace	Sample Count	Percentage
java.util.ArrayList.grow(int)	17	77,27%
java.util.ArrayList.ensureExplicitCapacity(int)	17	77,27%
java.util.ArrayList.ensureCapacityInternal(int)	17	77,27%
java.util.ArrayList.add(Object)	17	77,27%
com.baeldung.Main.main(String[])	17	77,27%
com.baeldung.Main.main(String[])	3	13,64%
java.util.ArrayList.add(Object)	2	9,09%
com.baeldung.Main.main(String[])	2	9,09%

JEP-364-365

ZGC on Mac/Win

JEP 364-365: ZGC on Mac/Win

- ZGC introduced for Win/Mac
- New experimental GC from Java 13
- Concurrent GC with stop the world <10ms that can support multi-terabytes heap
- XX:+UnlockExperimentalVMOptions -XX:+UseZGC
- In Russia we love Shenandoah GC

3

Things that you will not even notice

Features that 0% of you will notice

You won’t notice it

- G1 finally supports Non-uniform mem access architecture (performance)
  http://openjdk.java.net/jeps/345

- MappedByteBuffer support access to non-volatile memory (stability)
  https://openjdk.java.net/jeps/352

- Finally Solaris/SPARC, Solaris/x64 and Linux/SPARC ports deprecated
  https://openjdk.java.net/jeps/362

- Concurrent Mark Sweep (CMS) removed
  https://openjdk.java.net/jeps/363

- Pack200 (1.5 archiver) finally removed
  http://openjdk.java.net/jeps/367

Thanks!

DO YOU HAVE ANY QUESTIONS?