ORACLE

To Java 17 and Beyond!

Billy Korando

Oracle - Java Developer Advocate ☕🥑
@BillyKorando

Important Notes

- Ask questions
- Reach out:
  Twitter: @BillyKorando
  Email: billy.korando@oracle.com
- Live code examples at end time permitting 🤞
- Links at the end

Agenda

- New Language Features
- New Runtime Features
- Deprecations
  Removals
  And other changes to know about
- The Future

https://wkorando.github.io/sip-of-java/

New Language Features

Text Blocks

Added in Java 15

JEP 378

Text Blocks

```
String simpleJSONMessage = "{\n" + //
        "\t\"firstName\": \"Billy\",\n" + //
        "\t\"lastName\": \"Korando\",\n" + //
        "\t\"jobTitle\": \"Java Developer Advocate\",\n" + //
        "\t\"twitterHandle\": \"@BillyKorando\"\n" + //
        "}";
```


Text Blocks

```
String simpleJSONMessage = """
        {
                "firstName": "Billy",
                "lastName": "Korando",
                "jobTitle": "Java Developer Advocate",
                "twitterHandle": "@BillyKorando"
        }
        """;
```


Text Blocks

```
String simpleJSONMessage = """
        {
                "firstName": "%s",
                "lastName": "%s",
                "jobTitle": "%s",
                "twitterHandle": "%s"
        }
        """;

System.out.println(simpleJSONMessage.
        formatted("Billy",
                "Korando",
                "Java Developer Advocate",
                "@BillyKorando"));
```


Text Blocks

```
String aReallyLongLine = """
        Lorem ipsum dolor sit amet, consectetur adipiscing elit, \
        sed do eiusmod tempor incididunt ut labore et dolore \
        magna aliqua. \
        """;

System.out.println(aReallyLongLine);
```


Switch Expressions

Added in Java 14

JEP 361

Switch Expressions

```
switch (args[0]) {
case "1" -> System.out.println("Sunday");
case "2" -> System.out.println("Monday");
case "3" -> System.out.println("Tuesday");
case "4" -> System.out.println("Wednesday");
case "5" -> System.out.println("Thursday");
case "6" -> System.out.println("Friday");
case "7" -> System.out.println("Saturday");
default -> System.out.println("Invalid selection, valid choices 1-7");
}
```


Switch Expressions

```
String result = switch (args[0]) {
case "1" -> "Sunday";
case "2" -> "Monday";
case "3" -> "Tuesday";
case "4" -> "Wednesday";
case "5" -> "Thursday";
case "6" -> "Friday";
case "7" -> "Saturday";
default -> "Invalid Choice";
};

System.out.println(result);
```


Switch Expressions

```
String result = switch (args[0]) {
case "1" -> "Sunday";
case "2" -> "Monday";
case "3" -> "Tuesday";
case "4" -> "Wednesday";
case "5" -> "Thursday";
case "6" -> "Friday";
case "7" -> "Saturday";
default -> {
    System.out.println("Invalid selection, valid choices 1-7");
    yield "Invalid Choice";
}
};
```


Switch Expressions

```
enum DaysOfWeek {
    SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY ;
}

public static void main(String[] args) {

    DaysOfWeek dayOfWeek = DaysOfWeek.SUNDAY;
    String result = switch (dayOfWeek) {
    case SUNDAY -> "Sunday";
    case MONDAY -> "Monday";
    case TUESDAY -> "Tuesday";
    case WEDNESDAY -> "Wednesday";
    case THURSDAY -> "Thursday";
    case FRIDAY -> "Friday";
    case SATURDAY -> "Saturday";
    };

    System.out.println(result);
}
```

Pattern Matching for Instanceof

Added in Java 16

JEP 394

Pattern Matching for Instanceof

```
Object actuallyAString = "I'm actually a string!";

if(actuallyAString instanceof String) {//Test if actuallyAString is a String
    String nowImAString = //Assign actuallyAString to a variable
    (String) actuallyAString; //Convert actuallyAString to a String

    System.out.println(nowImAString);
}
```


Pattern Matching for Instanceof

```
Object actuallyAString = "I'm actually a string!";

if(actuallyAString instanceof String nowImAString) {
    System.out.println(nowImAString);
}
```


Pattern Matching for Instanceof

```
Object actuallyAString = "I'm actually a string!";

if(actuallyAString instanceof String nowImAString) {
    System.out.println(nowImAString);
}

System.out.println(nowImAString); //Compiler error, nowImAString not in scope

boolean isAString = (actuallyAString instanceof String nowImAString);

System.out.println(nowImAString); //Compiler error, nowImAString not in scope
```


Pattern Matching for Instanceof

```
Object actuallyAString = "I'm actually a string!";

if(actuallyAString instanceof String nowImAString
    || nowImAString.endsWith("!") //Compiler error, nowImAString not in scope
) {
    System.out.println(nowImAString);
}
```


Pattern Matching for Instanceof

```
Object actuallyAString = "I'm actually a string!";

if(actuallyAString instanceof String nowImAString
    && nowImAString.endsWith("!")) {
    System.out.println(nowImAString);
}
```


Pattern Matching for Instanceof

```
Object actuallyAString = "I'm actually a string!";

if (!(actuallyAString instanceof String nowImAString)) {
//...
} else {
    System.out.println(nowImAString);
}

if (!(actuallyAString instanceof String nowImAString)) {
    throw new IllegalArgumentException("Must be a string!");
}

System.out.println(nowImAString);
```


Pattern Matching for Instanceof

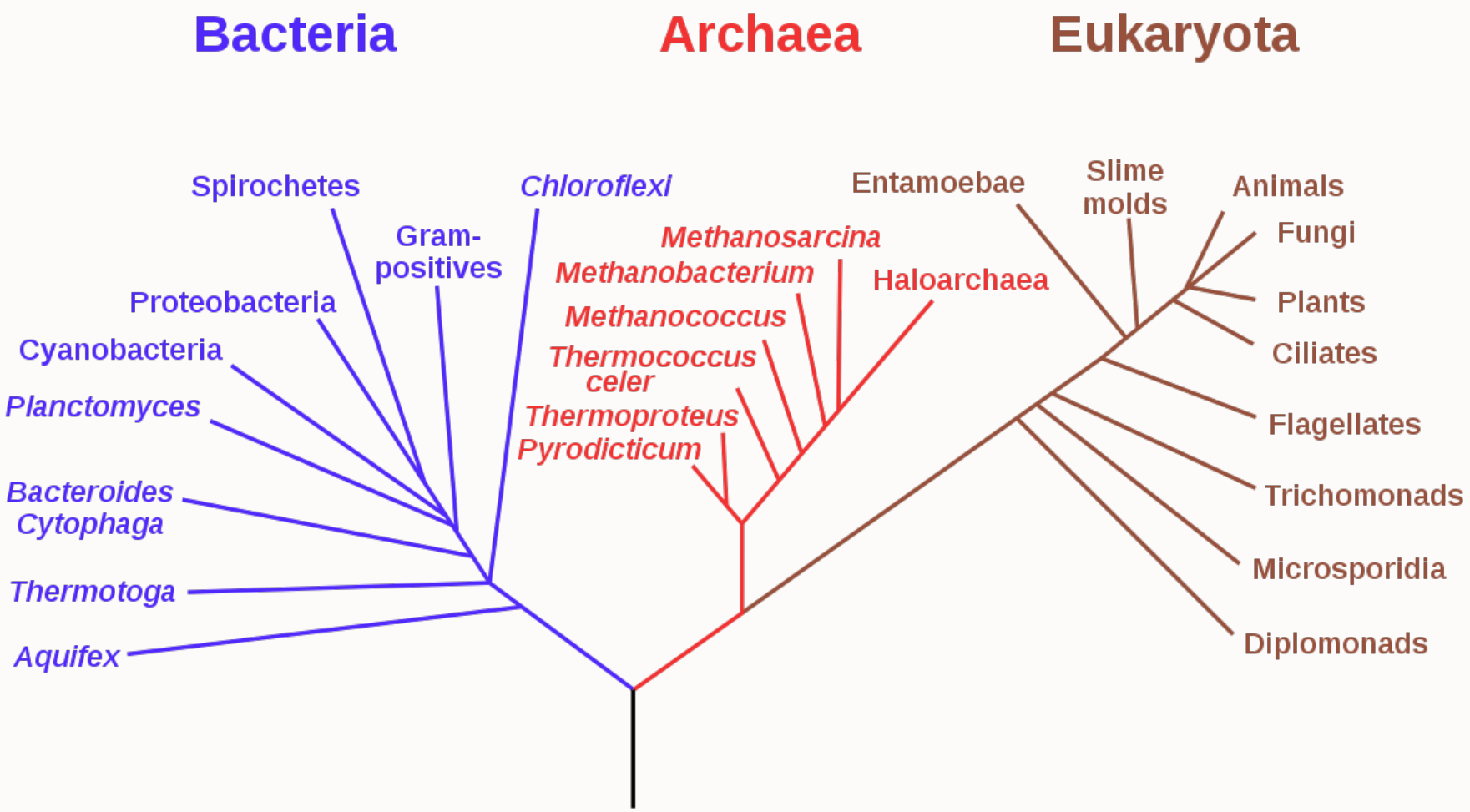
```
Object actuallyAString = "I'm actually a string!";
String aString = null;

if(actuallyAString instanceof String nowImAString) {
    aString  = nowImAString;
}

System.out.println(aString);
```


Sealed Classes

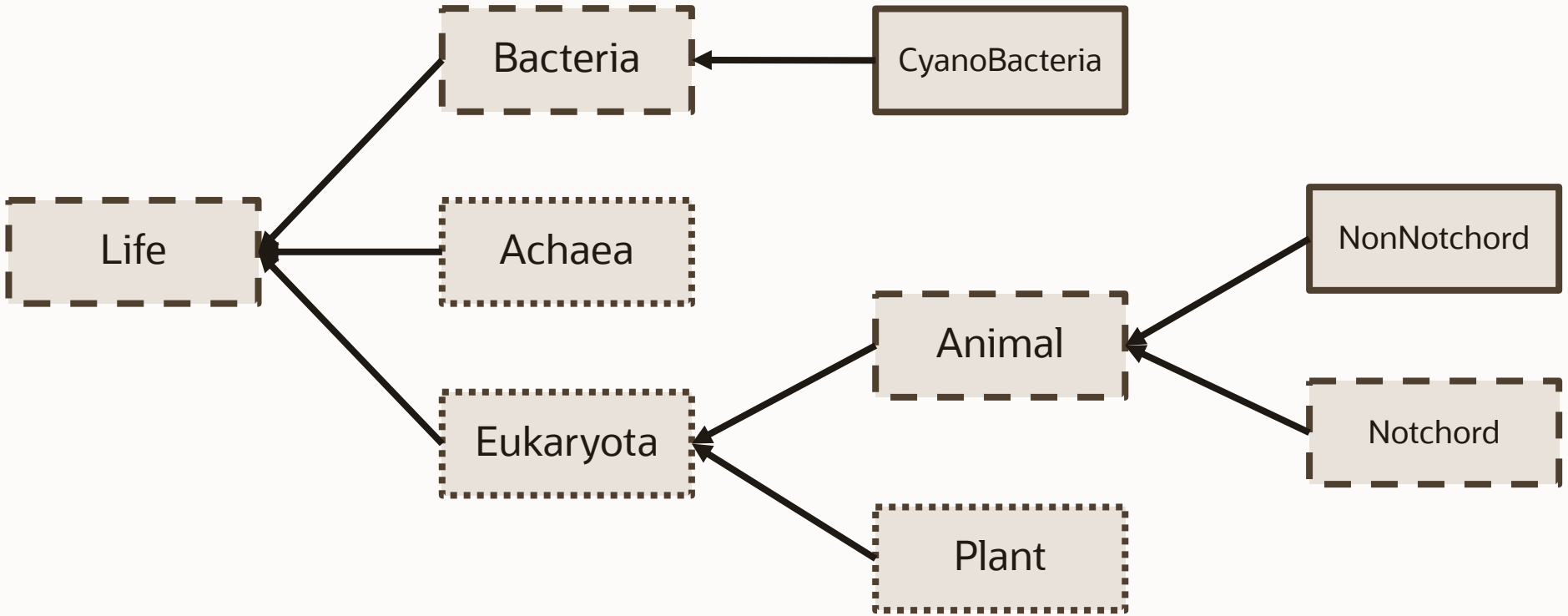
Added in Java 17

JEP 409

Sealed Classes



Sealed Classes



Records

Added in Java 16

JEP 395

Records

```
String firstName1 = "Billy";
String lastName1 = "Korando";
String title1 = "Java Developer Advocate";
String twitterHandle1 = "@BillyKorando";

String firstName2 = "Sharat";
String lastName2 = "Chander";
String title2 = "Java Developer Advocate";
String twitterHandle2 = "@Sharat_Chander";

class Person{
    private String firstName;
    private String lastName;
    private String title;
    private String twitterHandle;
    public Person(String firstName, String lastName, String title, String twitterHandle) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.title = title;
        this.twitterHandle = twitterHandle;
    }
    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + ((firstName == null) ? 0 : firstName.hashCode());
        result = prime * result + ((lastName == null) ? 0 : lastName.hashCode());
        result = prime * result + ((title == null) ? 0 : title.hashCode());
        result = prime * result + ((twitterHandle == null) ? 0 : twitterHandle.hashCode());
        return result;
    }
    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        Person other = (Person) obj;
        if (firstName == null) {
            if (other.firstName != null)
                return false;
        } else if (!firstName.equals(other.firstName))
            return false;
        if (lastName == null) {
            if (other.lastName != null)
                return false;
        } else if (!lastName.equals(other.lastName))
            return false;
        if (title == null) {
            if (other.title != null)
                return false;
        } else if (!title.equals(other.title))
            return false;
        if (twitterHandle == null) {
            if (other.twitterHandle != null)
                return false;
        } else if (!twitterHandle.equals(other.twitterHandle))
            return false;
        return true;
    }
    @Override
    public String toString() {
        return "Person [firstName=" + firstName + ", lastName=" + lastName + ", title=" + title
                + ", twitterHandle=" + twitterHandle + "]";
    }
}

var persons = Stream.of(new Person(firstName1, lastName1, title1, twitterHandle1),
new Person(firstName2, lastName2, title2, twitterHandle2));

persons.forEach(System.out::println);
```

Records

```
String firstName1 = "Billy";
String lastName1 = "Korando";
String title1 = "Java Developer Advocate";
String twitterHandle1 = "@BillyKorando";

String firstName2 = "Sharat";
String lastName2 = "Chander";
String title2 = "Java Developer Advocate";
String twitterHandle2 = "@Sharat_Chander";

record Person(String firstName, String lastName, String title, String twitterHandle) {}

var persons = Stream.of(new Person(firstName1, lastName1, title1, twitterHandle1),
        new Person(firstName2, lastName2, title2, twitterHandle2));

persons.forEach(System.out::println);
```

Records

- Transparent modeling of data as data
- Superclass always `java.lang.Record`
- Cannot be extend, abstract, and implicitly final
- All fields are final (shallowly immutable)
- Cannot declare instance fields
- Accessors, hashcode, toString, equals, automatically generated

Records

```
String firstName1 = "Billy";
String lastName1 = "Korando";
String title1 = "Java Developer Advocate";
String twitterHandle1 = "@BillyKorando";

String firstName2 = "Sharat";
String lastName2 = "Chander";
String title2 = "Java Developer Advocate";
String twitterHandle2 = "@Sharat_Chander";

record Person(String firstName, String lastName, String title, String twitterHandle) {
    public String toString() {
        return lastName + ", " + firstName +
        " twitter handle: " + twitterHandle +
        " job title: " + title;
    }
}

var persons = Stream.of(new Person(firstName1, lastName1, title1, twitterHandle1),
        new Person(firstName2, lastName2, title2, twitterHandle2));

persons.forEach(System.out::println);
```


Records

```
record Person(String firstName, String lastName, String title, String twitterHandle) {
    public String toJSON() {
        return """
                {
                    "firstName" : "%s",
                    "lastName" : "%s",
                    "title" : "%s",
                    "twitterHandle" : "%s"
                }
               """.formatted(firstName, lastName, title, twitterHandle);
    }
}
var persons = Stream.of(new Person(firstName1, lastName1, title1, twitterHandle1),
        new Person(firstName2, lastName2, title2, twitterHandle2));

persons.forEach(p -> System.out.println(p.toJSON()));
```


New Runtime Features and Improvements

CDS Updates

JDK 12
Default CDS Archive: JEP 341

JDK 13
Dynamic CDS Archive: JEP 350

AppCDS

java -XX:ArchiveClassesAtExit=dynamic-archive.jsa -jar <application>

java -XX:SharedArchiveFile=dynamic-archive.jsa -jar <application>

Z GC

JDK 15 (JEP 377)

- From scratch re-design of GC on Java
- Single generation
- Low Latency (<10ms pause times)
- Scalable (multi-terabyte heaps)
- Get started: `-XX:+UseZGC -Xmx<size> -Xlog:gc`

Helpful NullPointerExceptions

JDK 14

JEP 358

Helpful NullPointerExceptions

```
Exception in thread "main" java.lang.NullPointerException
    at NPEService.main(NPEService.java:7)
```

```
Exception in thread "main" java.lang.NullPointerException:
Cannot invoke "String.length()" because "<local1>" is null
at NPEService.main(NPEService.java:7)
```


AArch64 Support

AArch64 (Arm Arch 64 support)

- Linux JDK 9 (JEP 237)
- Windows JDK 16 (JEP 388)
- macOS JDK 17 (JEP 391)

General Performance Improvements

Spring Boot Petclinic 2.4.5

Requests: 500 (serial)
Heap size: 48 MB
Garbage collector: G1

Java 8 Results (1.8.0_291-b10)
Application startup: 8.665 seconds
Load test execution time: 137 seconds

Java 16 Results (16.0.1+9-24)
Application startup: 6.452 seconds
Load test execution time: 130 seconds

Deprecations
Removals
And other changes to know about

Deprecations

Security Manager
JDK 17
JEP 411

Applet API
JDK 17
JEP 398

Removals

Nashorn (JavaScript Engine)
JDK 15
JEP 372

CMS Garbage Collector
JDK 14
JEP 363

Other Changes

Strongly Encapsulate JDK Internals
JDK 17
JEP 403

No longer relax strong encapsulation with single argument (`i.e. --illegal-access=permit`)

Some critical internal APIs remain available:

- `sun.misc.{Signal,SignalHandler}`
- `sun.misc.Unsafe` (The functionality of many of the methods in this class is available via *variable handles* (JEP 193).)
- `sun.reflect.Reflection::getCallerClass(int)` (The functionality of this method is available in the stack-walking API defined by JEP 259.)
- `sun.reflect.ReflectionFactory`
- `com.sun.nio.file.{ExtendedCopyOption,ExtendedOpenOption, ExtendedWatchEventModifier,SensitivityWatchEventModifier}`

Strong Encapsulation (cont.)

Eventually `--illegal-access=permit` will be removed

Can still declare `--add-opens` to command line and Add-Opens to JAR-Manifest

The Future

Project Amber & Pattern Matching

Java 17

JEP 406

Pattern Matching for switch (Preview)

```
switch (s) {
    case null         -> System.out.println("Oops");
    case "Foo", "Bar" -> System.out.println("Great");
    default           -> System.out.println("Ok");
}
```

```
switch (s) {
    case Triangle t && (t.calculateArea() > 100) ->
        System.out.println("Large triangle");
    case Triangle t ->
        System.out.println("Small triangle");
    default ->
        System.out.println("Non-triangle");
}
```


Project Amber & Pattern Matching

Java 18

JEP 405

Record Patterns & Array Patterns (Preview)

Project Amber & Pattern Matching

Array Patterns:

Old:

```
if (o instanceof String[] sa && sa.length >= 2) {
    String s1 = sa[0];
    String s2 = sa[1];
    System.out.println(s1 + s2);
}
```

New:

```
if (o instanceof String[] { String s1, String s2, ... }){
    System.out.println(s1 + s2);
}
```


Project Amber & Pattern Matching

Records Patterns:

Defined record:

```
record MultiColoredPoint(int i, int j, Color... cols) {}
```

Current way to assign values:

```
MultiColoredPoint(var a, var b, Color[] { var firstColor, var secondColor, ... })
```

Assigning values with Record Patterns:

```
MultiColoredPoint(var a, var b, var firstColor, var secondColor, ...)
```


Project Amber & Pattern Matching

- Pattern Matching for Sealed classes
- Deconstruction Patterns

Read more:
https://cr.openjdk.java.net/~briangoetz/amber/pattern-match.html

Early Access Builds

https://jdk.java.net/17/
https://jdk.java.net/18/
https://jdk.java.net/loom/
https://jdk.java.net/panama/
https://jdk.java.net/valhalla/

Links and Code Examples

Slide deck: https://github.com/wkorando/to-java-17-and-beyond

https://openjdk.java.net/jeps/0

https://github.com/wkorando/sip-of-java
https://wkorando.github.io/sip-of-java/

https://github.com/wkorando/spring-petclinic-records

https://carlmastrangelo.com/blog/the-impossible-java-11

Thank you

Twitter: @BillyKorando
Email: billy.korando@oracle.com


ORACLE