ORACLE

To Java 18 and Beyond!

Billy Korando

Oracle - Java Developer Advocate ☕🥑
@BillyKorando

Important Notes

- Ask questions
- Reach out:
  Twitter: @BillyKorando
  Email: billy.korando@oracle.com
- Link to slides 👉 https://github.com/wkorando/to-java-18-and-beyond

- https://dev.java
- https://inside.java
- https://youtube.com/java
- Inside Java Podcast
- Inside Java Newscast
- #SipOfJava
- #JEPCafé

Agenda

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

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 Updates

Added in Java 14
JEP 361

Switch Updates

```
switch(d){
    case 1:
     System.out.println("Sunday");
     break;
    case 2:
     System.out.println("Monday");
     break;
    case 3:
     System.out.println("Tuesday");
     break;
    case 4:
     System.out.println("Wednesday");
     break;
    case 5:
     System.out.println("Thursday");
     break;
    case 6:
     System.out.println("Friday");

    case 7:
     System.out.println("Saturday");
     break;
}
```


Switch Updates

```
switch(d){
    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");
}
```

Switch Updates

```
String day = switch(d){
     case 1 -> "Sunday";
     case 2 -> "Monday";
     case 3 -> "Tuesday";
     case 4 -> "Wednesday";
     case 5 -> "Thursday";
     case 6 -> "Friday";
     case 7 -> "Saturday";
     default -> throw new IllegalArgumentException();
}
```

Switch Updates

```
String day = switch(d){
     case 1 -> "Sunday";
     case 2 -> "Monday";
     case 3 -> "Tuesday";
     case 4 -> "Wednesday";
     case 5 -> "Thursday";
     case 6 -> {
       System.out.println("Ladies and Gentlemen, the Weekend");
       yield "Friday";
     }
     case 7 -> "Saturday";
     default -> throw new IllegalArgumentException();
}
```


Switch Updates

```
String day = switch(d){
    case 1:
        yield "Sunday";
    case 2:
        yield "Monday";
    case 3:
        yield "Tuesday";
    case 4:
        yield "Wednesday";
    case 5:
        yield "Thursday";
    case 6:
        yield "Friday";
    case 7:
        yield "Saturday";
    default:
        throw new IllegalArgumentException();
}
```


Switch Updates

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

DaysOfWeek dayOfWeek = [someDayOfWeek]

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


Switch Updates

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

String day = switch(dayOfWeek){
    case SUNDAY :
        yield "Sunday";
    case MONDAY :
        yield "Monday";
    case TUESDAY :
        yield  "Tuesday";
    case WEDNESDAY:
        yield "Wednesday";
    case THURSDAY:
        yield "Thursday";
    case FRIDAY:
        yield "Friday";
    case SATURDAY:
        yield "Saturday";
};
```

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);
}
```

Predicate

Pattern Variable

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
```

In scope where compiler knows pattern variable is set

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);
```

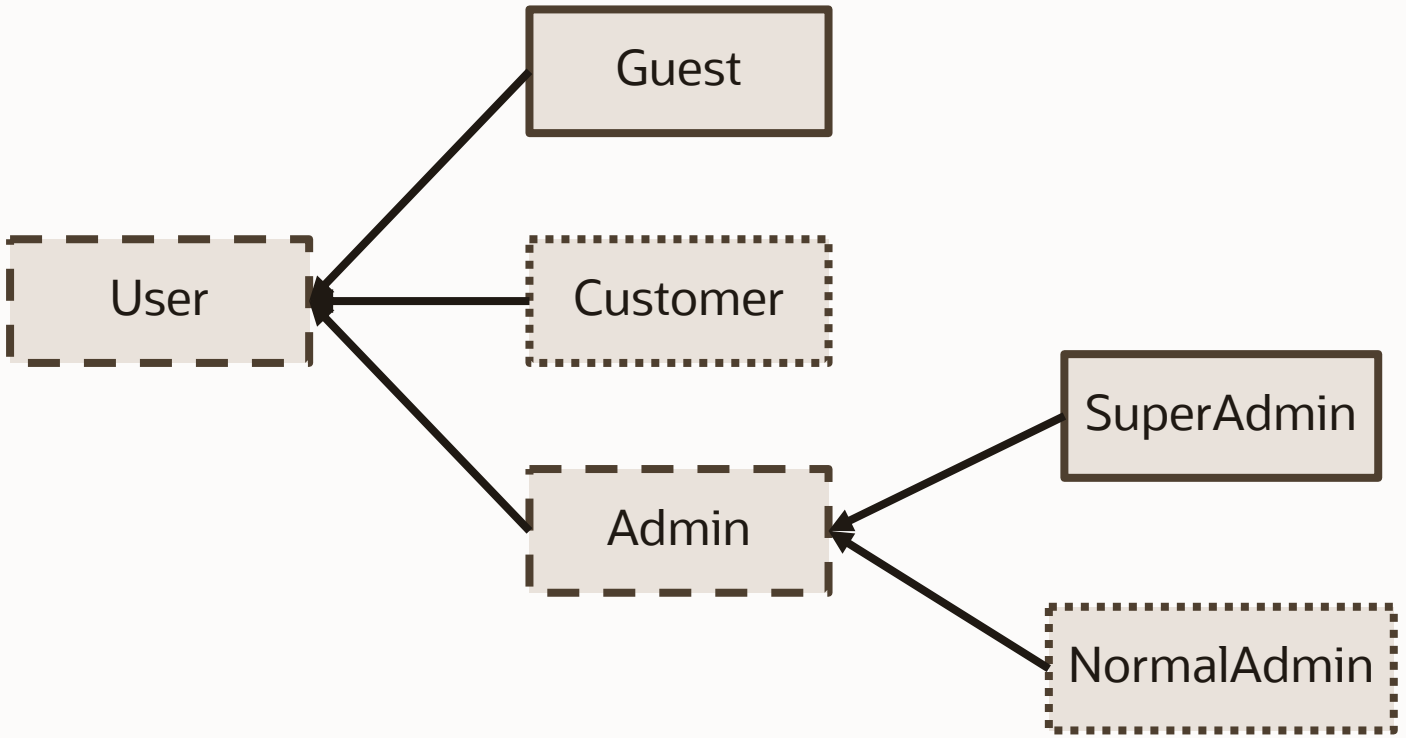

Pattern Matching for Instanceof

```
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

```
public abstract sealed class User
permits Admin, Customer, Guest{

}

public sealed class Admin
extends User permits SuperAdmin, NormalAdmin{

}

public final class Guest extends User {

}

public non-sealed class Customer extends User {

}
```


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
- Can override default implementation, or add your own method definitions

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()));
```


Sealed Classes & Records

```
public sealed interface User{}

public record Admin(...) implements User{}
public record Guest(...) implements User{}
public record Customer(...) implements User{}
```


Code Snippets in Java API Documentation

JDK 18
JEP 413

Code Snippets in Java API Documentation

In-line:

```
/**
 * The following code shows how to use {@code Optional.isPresent}:
 * {@snippet :
 * if (v.isPresent()) {
 *     System.out.println("v: " + v.get());
 * }
 * }
 */
```

External

```
/**
 * The following code shows how to use {@code Optional.isPresent}:
 * {@snippet file="ShowOptional.java" region="example"}
 */
```

```
public class ShowOptional {
    void show(Optional<String> v) {
        // @start region="example"
        if (v.isPresent()) {
            System.out.println("v: " + v.get());
        }
        // @end
    }
}
```

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 (<1 ms 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)

Simple Web Server

JDK 18
JEP 408

Simple Web Server - Basics

- Start the server: `jwebserver`
- Default server address: `http://127.0.0.1:8000`
- Serve files from current directory
- MIME types automatically configured
  - `.html` rendered as `text/html`
  - `.java` rendered as `text/plain`
- MIME types customizable via: `java.net.URLConnection::getFileNameMap` API

Simple Web Server - Customizations

- `-p` configure port
- `-b` configure binding address
- `-d` for the directory to serve
  - Can be a remote directory
- `-o` configure console output `none|info|verbose` default: `info`
- `-h` to display help, which includes available configuration flags

General Performance Improvements

Spring Boot Petclinic 2.50

Java 8 Results (1.8.0_291-b10)
Application startup: 5.265 seconds
Memory: 673M

Java 17 Results (build 17+35-2724)
Application startup: 4.956 seconds
Memory: 282M

Java 18 Results (build 18+36-2087)
Application startup: 4.909 seconds
Memory: 295M

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}`

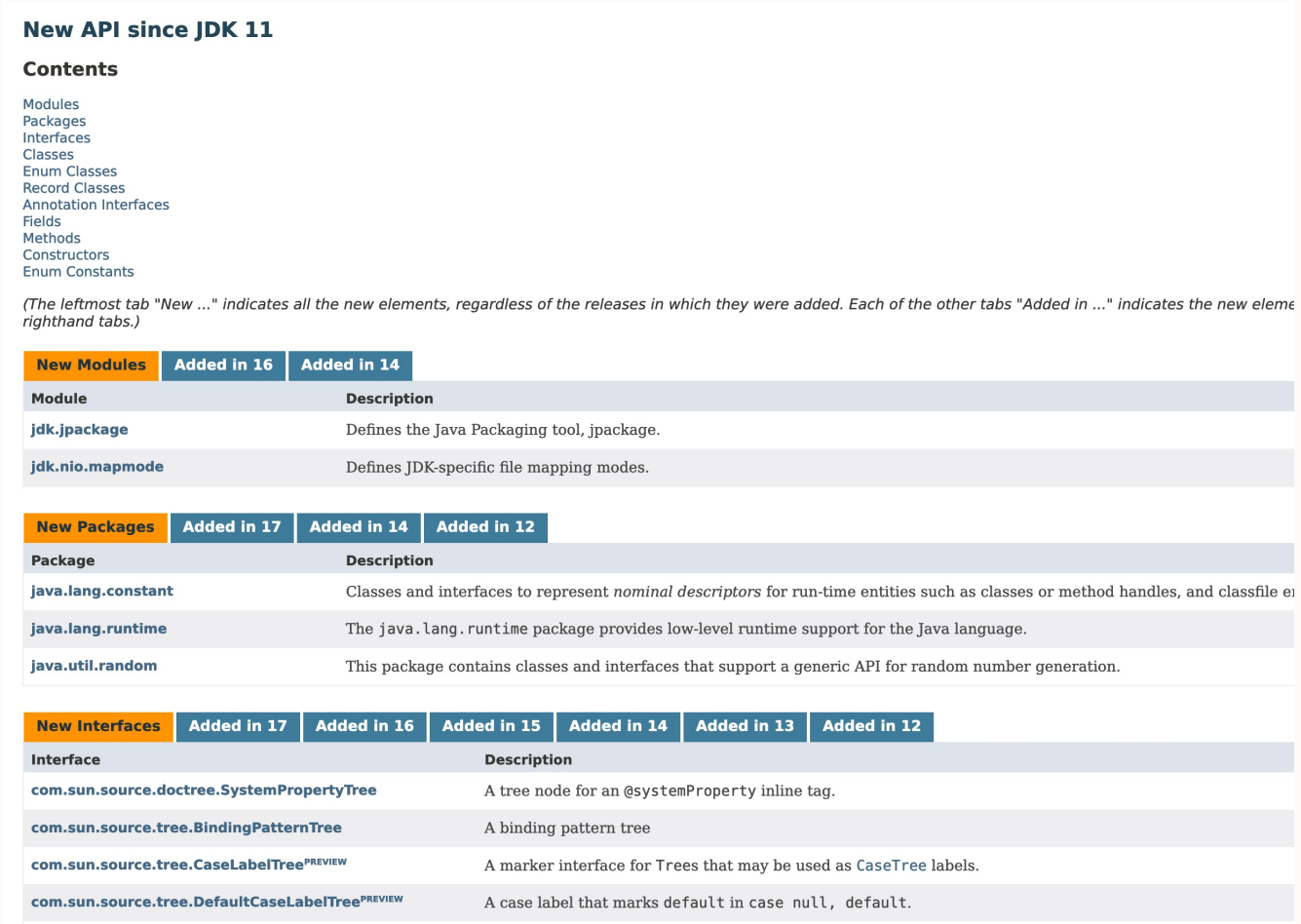
Strong Encapsulation (cont.)

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

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

New Classes, Methods, Fields, etc...

Javadoc New (and Deprecated)



New API since JDK 11

Contents

(The leftmost tab "New ..." indicates all the new elements, regardless of the releases in which they were added. Each of the other tabs "Added in ..." indicates the new eleme
righthand tabs.)

New Modules | **Added in 16** | **Added in 14**

Module	Description
jdk.jpackage	Defines the Java Packaging tool, jpackage.
jdk.nio.mapmode	Defines JDK-specific file mapping modes.

New Packages | **Added in 17** | **Added in 14** | **Added in 12**

Package	Description
java.lang.constant	Classes and interfaces to represent *nominal descriptors* for run-time entities such as classes or method handles, and classfile e
java.lang.runtime	The `java.lang.runtime` package provides low-level runtime support for the Java language.
java.util.random	This package contains classes and interfaces that support a generic API for random number generation.

New Interfaces | **Added in 17** | **Added in 16** | **Added in 15** | **Added in 14** | **Added in 13** | **Added in 12**

Interface	Description
com.sun.source.doctree.SystemPropertyTree	A tree node for an `@systemProperty` inline tag.
com.sun.source.tree.BindingPatternTree	A binding pattern tree
com.sun.source.tree.CaseLabelTree PREVIEW	A marker interface for `Trees` that may be used as `CaseTree` labels.
com.sun.source.tree.DefaultCaseLabelTree PREVIEW	A case label that marks `default` in `case null, default`.

docs.oracle.com/en/java/javase/18/docs/api/index.html

The Future

Pattern Matching for Switch

Currently in Preview 2
JEP 420
Soon in Preview 3
JEP 8282272 (Draft)

Pattern Matching for Switch

```
return switch (o) {
      case Integer i -> String.format("int %d", i);
      case Long l    -> String.format("long %d", l);
      case Double d  -> String.format("double %f", d);
      case String s  -> String.format("String %s", s);
      default        -> o.toString();
};
```

Pattern Matching for Switch

```
return switch (o) {
     case Integer i when i > 0 && i / 2 -> System.out.println("Positive & even" + i);
     case Integer i when i > 0 -> System.out.println("Positive & odd" + i);
     case Integer i -> System.out.println("Zero or negative" + i);
     default        -> o.toString();
};
```


Pattern Matching for Switch

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


Pattern Matching for Switch

```
public sealed interface User{}

public record Admin(...) implements User{}
public record Guest(...) implements User{}
public record Customer(...) implements User{}

User u = ...

User someUser = switch(u){
    case Admin a -> ...;
    case Guest g -> ...;
    case Customer c -> ...;
}
```


Pattern Matching for Switch

```
public sealed interface User{}

public record Admin(...) implements User{}
public record Guest(...) implements User{}
public record Customer(...) implements User{}

User u = ...

switch(u){  //Error not exhaustive!
    case Admin a -> ...;
    case Customer c -> ...;
}
```


Record Patterns

Currently in candidate status
JEP 405

Record Patterns

```
record Point(int x, int y) {}
enum Color { RED, GREEN, BLUE }
record ColoredPoint(Point p, Color c) {}
record Rectangle(ColoredPoint upperLeft, ColoredPoint lowerRight) {}

static void printColorOfUpperLeftPoint(Rectangle rectangle) {
    if(rectangle != null){
        ColoredPoint ul = rectangle.ul();
        if (ul != null) {
            Color c = ul.c();
            System.out.println(c);
        }
    }
}
```


Record Patterns

```
record Point(int x, int y) {}
enum Color { RED, GREEN, BLUE }
record ColoredPoint(Point p, Color c) {}
record Rectangle(ColoredPoint upperLeft, ColoredPoint lowerRight) {}

static void printColorOfUpperLeftPoint(Rectangle rectangle) {
    if(rectangle instaceof Rectangle r){
        ColoredPoint ul = r.ul();
        if (ul != null) {
            Color c = ul.c();
            System.out.println(c);
        }
    }
}
```


Record Patterns

```
record Point(int x, int y) {}
enum Color { RED, GREEN, BLUE }
record ColoredPoint(Point p, Color c) {}
record Rectangle(ColoredPoint upperLeft, ColoredPoint lowerRight) {}

static void printColorOfUpperLeftPoint(Rectangle rectangle) {
    if(rectangle instaceof Rectangle(ColoredPoint ul, ColoredPoint lr){
        Color c = ul.c();
        System.out.println(c);
    }
}
```


Record Patterns

```
record Point(int x, int y) {}
enum Color { RED, GREEN, BLUE }
record ColoredPoint(Point p, Color c) {}
record Rectangle(ColoredPoint upperLeft, ColoredPoint lowerRight) {}

static void printColorOfUpperLeftPoint(Rectangle rectangle) {
    if(rectangle instaceof Rectangle(ColoredPoint(Point p, Color c), ColoredPoint lr){
        System.out.println(c);
    }
}
```


Early Access Builds

https://jdk.java.net/19/
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-18-and-beyond

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

https://jdk.java.net/18/release-notes

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

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

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

Thank you

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


ORACLE