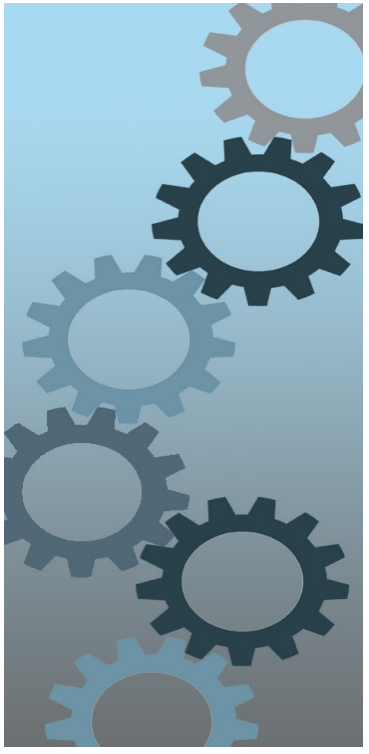
HSLU Hochschule Luzern

Programming Concepts And Paradigms

New Concepts
NIO, Loom, Valhalla, Amber



Marcel Baumann

Content

- NIO and NIO.2
- Local Date and Time
- Auto Closeable *java.lang.AutoCloseable*
- javax.measure
- Project Loom
- Project Valhalla
- Project Amber

Java Ecosystem

Kotlin

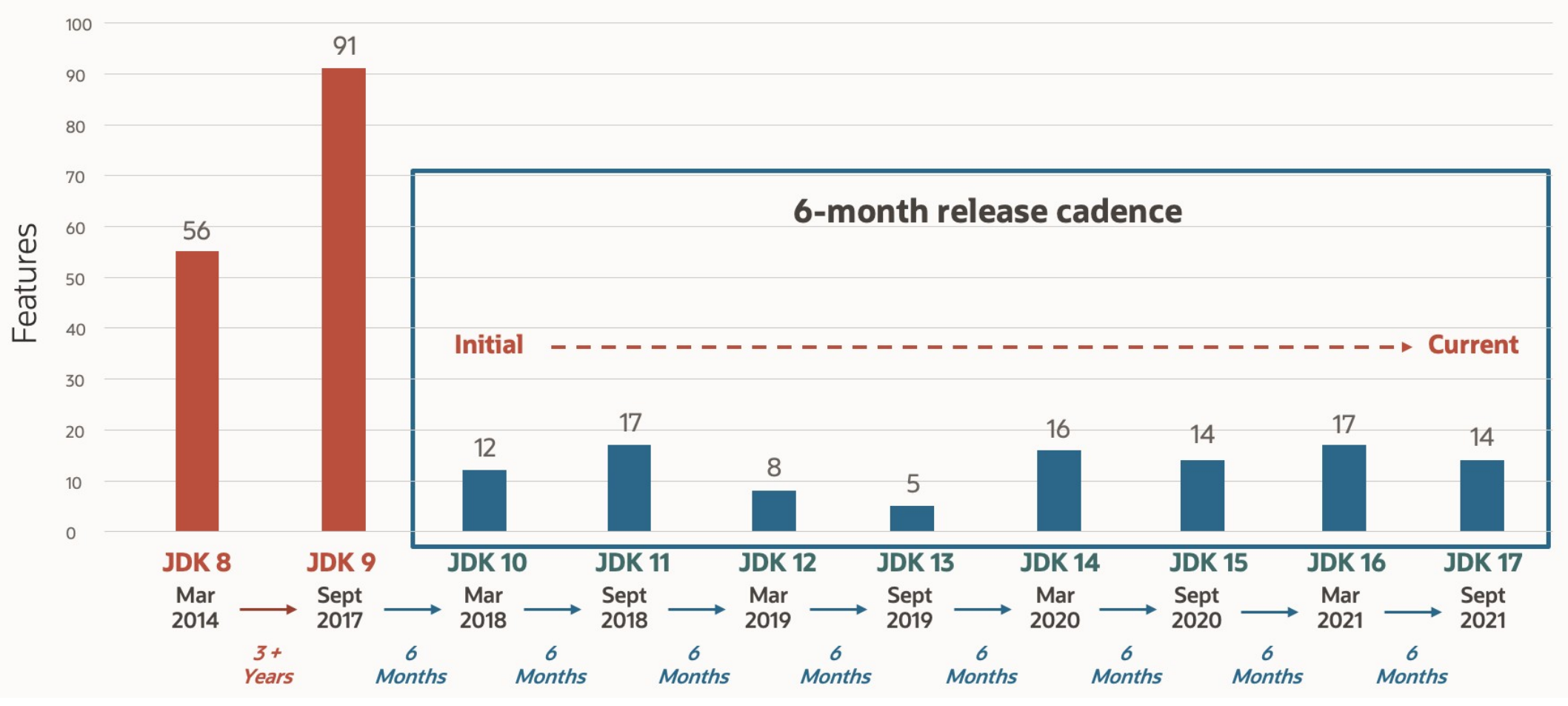
Groovy

Scala

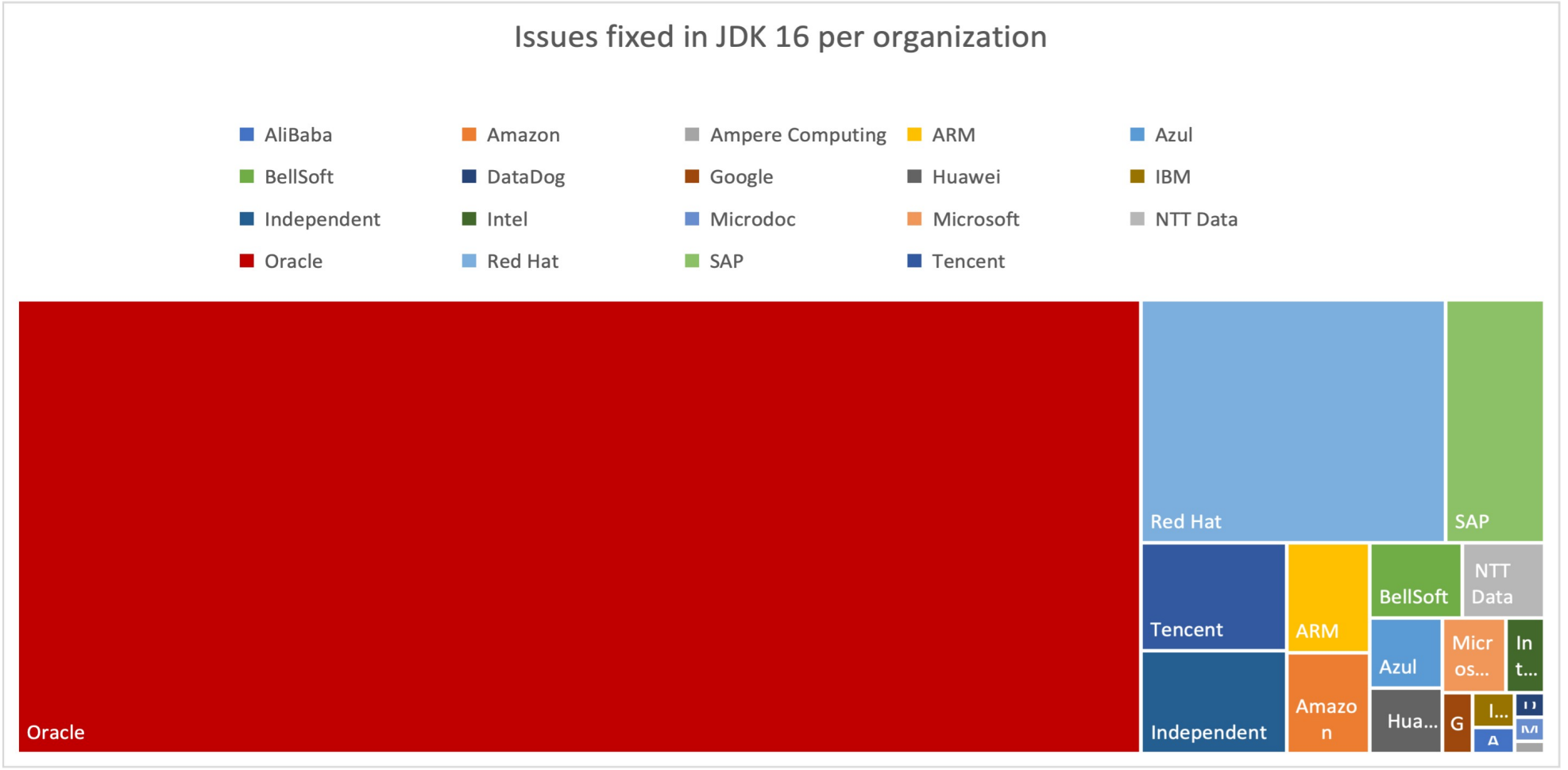
Java

GraalVM

Java Releases



Open Source Effort



New Input Output NIO and NIO.2

- Design error with *java.io.File*
  - *Does not support file system abstraction*
- Correction with *java.nio.file.Path*
- Provides Files and Paths utility class
- Huge improvement with FileSystem concept

Stream Oriented Functions

```
public static Stream<Path> walk(Path start,
      int maxDepth,
      FileVisitOption... options)throws IOException

public static Stream<Path> find(Path start,
      int maxDepth,
      BiPredicate<Path,BasicFileAttributes> matcher,
      FileVisitOption... options) throws IOException

public static Stream<String> lines(Path path,
      Charset cs) throws IOException
```

Date and Time

- History
  - One of the worst class *java.util.Date*
  - Why did they invent *java.sql.Date?*
  - Save the world: Joda library
- Correctness through *java.time* package
  - **Immutable instances**
  - **Date operations**

Date and Time

- Instant
- LocalTime
- LocalDate
- LocalDateAndTime
- ZonedLocalDateAndTime
- *(Year, YearMonth, DayOfWeek, Clock)*
- *(java.time.chrono → Generic API for calendar systems other than the default ISO.)*

Date and Time

- Instant
- LocalTime
- LocalDate
- LocalDateAndTime
- ZonedLocalDateAndTime
- *(Year, YearMonth, DayOfWeek, Clock)*
- *(java.time.chrono → Generic API for calendar systems other than the default ISO.)*

java.time

- *Instant* is a timestamp
- *LocalDate* is a date without a time, or any reference to an offset or time-zone
- *LocalTime* is a time without a date, or any reference to an offset or time-zone
- Composition
  - LocalDateTime combines date and time, but still without any offset or time-zone
  - ZonedDateTime is a "full" date-time with time-zone and resolved offset from UTC/Greenwich
  - OffsetTime, ZoneId and ZoneOffset, Year and YearMonth
  - Period is a time-based amount of time, such as '34.5 seconds'.

AutoCloseable

- Finalizers are obsolete
  - They are a design error because they cannot be implemented efficiently with garbage collectors
- AutoCloseable and try with resources
  - Compatibility by updating Closeable
    Closeable implements AutoCloseable
  - Compiler eliminates verbosity and programmer’s errors
  - Mix API and language extension

Text Blocks & Formatting

- Readable text constants

  ”””
  ....
  “””

- Formatting

```
return """
    LegalEntity[oid=%s, id=%s, name=%s, fromDate=%s, toDate=%s, text=%s, vatNr=%s, tags=%s]
     """.formatted(oid(), id(), name(), fromDate(), toDate(), text(), vatNr(), tags());
```

Modern equals()

```
class T {

   ...

   @Override
   public boolean equals(Object obj) {
       return (obj instanceof T that) &&
           (Objects.equals(x, that.x) && ...;
```

Modern toString()

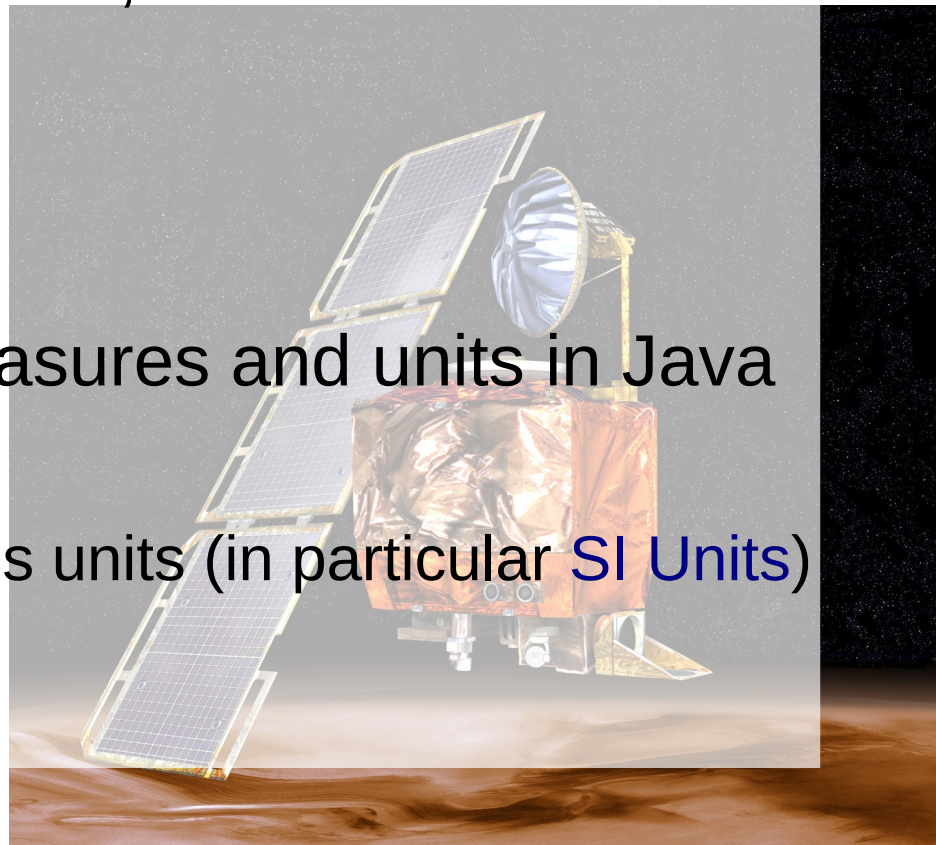
```
class T {

   ...

   @Override
   public String toString() {
        return """
            Activity[date=%s, code=%s]
            """.formatted(date(), code());
   }
```

Control Questions

1. Why was the *java.util.Date* class not removed?
2. How can you discourage developer to use a class or a method in Java?

javax.measure

- JSR-385 *API 2.0* (JSR-363 *API 1.0*, JSR-275 *Unit Specifications*)
  - javax.measure:unit-api:2.1.2
  - tec.units:indriya:2.1.2
- Unified way of representing measures and units in Java
  - Checking of unit compatibility
  - Expression of a quantity in various units (in particular SI Units)
  - Arithmetic operations on units

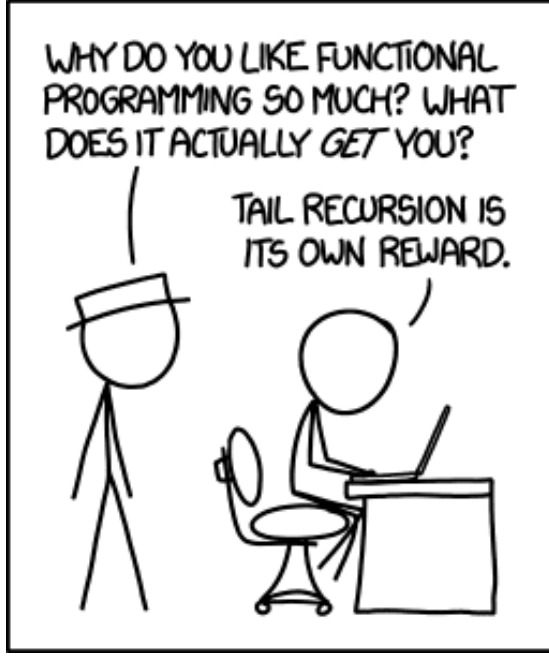


javax.measure

```
double distanceInMeters = 50.0;


UnitConverter metreToKilometre =
   METRE.getConverterTo(MetricPrefix.KILO(METRE));


double distanceInKilometers =
   metreToKilometre.convert(distanceInMeters );
```

Loom

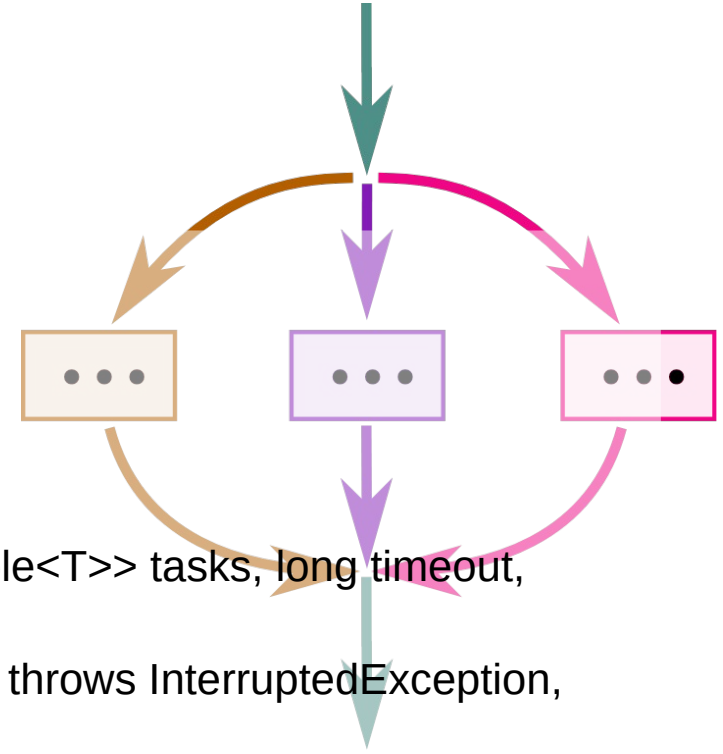
- Virtual threads
- Delimited continuations
- Tail-call elimination
- *All three features are available in Scheme for decades ;-)*
- *Virtual threads exist in Smalltalk, Erlang, Go, etc.*



Loom

```
Thread thread = Thread.startVirtualThread(runnable);

ExecutorService executor = Executors.newVirtualThreadExector();


executor.submit(runnable);
executor.submit(callable);
```

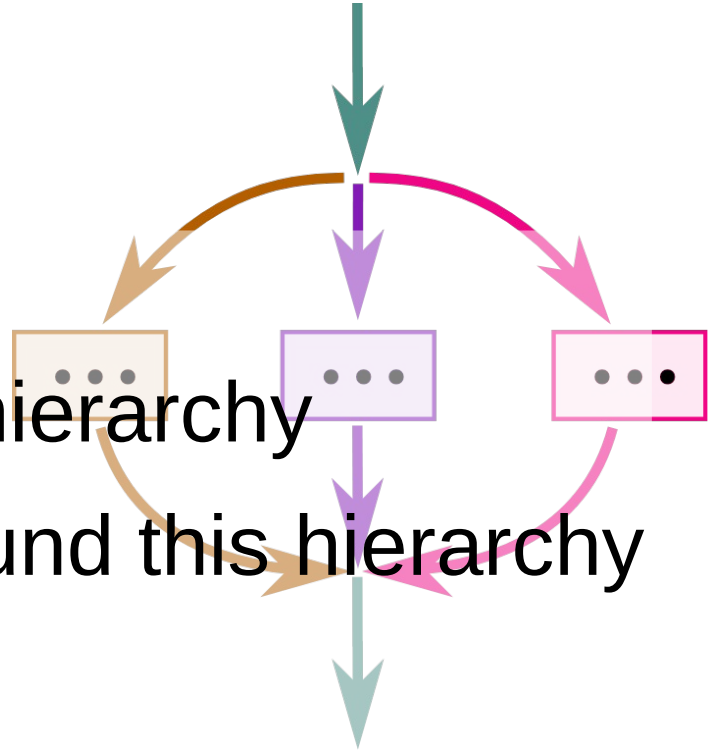
Structured Concurrency



- Synchronization Tools
  - Fork And Join
  - Future
  - ExecutorService
    - <T> List<Future<T>> invokeAll(Collection<? extends Callable<T>> tasks, long timeout, TimeUnit unit) throws InterruptedException
    - <T> T invokeAny(Collection<? extends Callable<T>> tasks) throws InterruptedException, ExecutionException
  - CompletableFuture
- The goal is
  - No complex synchronization to wait for completion
  - Follow the flow of the program, do not create a new flow (see for example async for a new flow with rules such as only async calls)

Structured Concurrency

- Bind thread lifetime to a scope
- Interpret this as a parent – child hierarchy
- Build programming concepts around this hierarchy



Structured Concurrency

```
var NTASKS = 1000;

try (ExecutorService exec = Executors.newVirtualThreadExecutor()) {
   for (int i = 0; i < NTASKS; i++) {
      exec.submit(() -> {
          try { TimeUnit.SECONDS.sleep(1);}
          catch (InterruptedException e) {}});
   }
}

// Blocks until all threads completed,
// ExecutorService implements AutoClosable
```

```
ExecutorService exec =
Executors.newVirtualThreadExecutor().withDeadline(
   Instant.now().plus(30, ChronoUnit.SECONDS));
```

```
CompletableFuture<Integer> future = exec.submitTask(callable);
```

Valhalla

- Numeric Type System Unification → everything is an object
- But not all objects have an identity
  - So the object does no more follow object-oriented axioms of: *state, identity and behavior*

Valhalla Steps

- Warning for value based class
  - Do not use constructors for value based classes such as Boolean, Byte, Short, Character, Integer, Long, Float, Double.
  - Value classes are marked with *@jdk.internal.ValueBased* annotation
  - Warning if you synchronize *synchronized* on an instance of such a class. Anyway stop using synchronized and move to concurrency package.

Valhalla Next Steps

- Primitive Objects (JEP 401)
  - primitive class Point implements Shape { … }
  - *Implicit final class with implicit final instance variables → immutable objects*
  - *== on fields equality and not on object equality*
- Unify Basic Primitives with Objects (JEP 402)
  - int and Integer will be the same

Graal VM

- Polyglot environment
  - Seamlessly use multiple languages and libraries
  - Support to compile own language
- Self-hosted Java
- Ahead of Time Compilation AOT (*next try)*
  - Increase application throughput and reduce latency
  - Compile applications into small self-contained native binaries

REPL in Java: Jshell & JBang

- Support of exploratory programming
  - REPL is a tool to test interactively small programs. Jshell was introduced in JDK 9
  - Embedded Web Server is a tool to test small web programs in JDK 18

Lombok Approach

- Annotation approach
  - @Data
  - @Value
  - @Wither
  - @Builder, @SuperBuilder
  - @Getter, @Setter, @Log
  - @Accessors

Amber

- *Records and Sealed Types*
- *Pattern matching for **instanceof** in if*
- Pattern matching for switch statements
- Deconstruction of records and arrays
- Concise method body

Guarded Pattern With switch

```
static void test(Object o) {
    switch (o) {
        case String s && (s.length() == 1) -> …
        case String s                      -> …
    }
}
```

Pattern Matching With if and switch

```
if ((obj instanceof String s) && (s.length() > 5)) {
    flag = s.contains("jdk");
}

String formatted = 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 Record

```
static void printXCoordOfUpperLeftPointWithPatterns(
   Rectangle r) {
    if (r instanceof Rectangle(ColoredPoint(
         Point(var x, var y), var c), var lr)) {
        System.out.println("Upper-left corner: " + x);
    }
}
```

Pattern Matching Arrays

```
static void printSumOfFirstTwoXCoords(Object o) {
    if (o instanceof Point[] { Point(var x1, var y1),
Point(var x2, var y2), ... }) {
        System.out.println(x1 + x2);
    }
}
```

Control Questions

1. What is a virtual (also called green) thread?
2. What is the meaning if tail-call optimization?
3. Why are pattern matching features so hype?
4. What is a deconstruction pattern

Exercises

- Implements toString, equals, hashCode and Comparable
- javax.measure example