JDK22

WHAT'S IN FOR YOU

azul

ABOUT me...

Gerrit Grunwald | Developer Advocate | Azul

@hansolo_

azul

JDK ENHANCEMENT PROPOSALS IN JDK 22

azul

JEP's

- JEP 423 Region Pinning for G1
- JEP 447 Statements before super()*
- JEP 454 Foreign Function & Memory API
- JEP 456 Unnamed Variables & Patterns
- JEP 457 Class-File API*
- JEP 458 Launch Multi-File Source-Code programs
- JEP 459 String Templates*
- JEP 460 Vector API **
- JEP 461 Stream Gatherers*
- JEP 462 Structured Concurreny*
- JEP 463 Implicitly Declared Classes and Instance Main Methods*
- JEP 464 Scopes Values*

* Preview ** Incubator

azul

423

REGION PINNING FOR G1

[STABLE]

azul

JEP 423

REGION PINNING FOR G1

- Do not disable G1 in presence of JNI critical regions
- Before: when thread was in JNI region, G1 must wait
- Now: G1 pins regions in minor and major collection
- Pinned regions in young gen. will be promoted to old gen
- Pinned regions in old gen won't be evacuated
- As a result garbage collection can be performed normally
  (even in presence of JNI critical regions)

492

STATEMENTS BEFORE SUPER()

[PREVIEW]

azul

JEP 447

STATEMENTS BEFORE SUPER()

- Java provides the ability to extend another (non final) class
- Ability to inherit state and behavior of superclass
- Therefor the call to a superclass constructor must come first
- Testing parameters before calling the superclass constructor is more complex

JEP 447

STATEMENTS BEFORE SUPER()

- Allows certain statements to be executed before super()
- Useful when passing values to base constructor from extended classes
- You cannot access instance variables or execute methods of your derived class before calling super()

JEP 447

STATEMENTS BEFORE SUPER()

```
public class PositiveBigInteger extends BigInteger {

  public PositiveBigInteger(long value) {
    super(value);
    if (value <= 0) {
      throw new IllegalArgumentException("non-positive value");
    }
  }

}
```

Validate in constructor

JEP 447

STATEMENTS BEFORE SUPER()

```
public class PositiveBigInteger extends BigInteger {

  public PositiveBigInteger(long value) {
    super(verifyPositive(value));
  }

  private static long verifyPositive(long value) {
    if (value <= 0) {
      throw new IllegalArgumentException("non-positive value");
    }
    return value;
  }

}
```

Validate in constructor using an extra static method

JEP 447

STATEMENTS BEFORE SUPER()

```
public class PositiveBigInteger extends BigInteger {

  public PositiveBigInteger(long value) {
    if (value <= 0) {
      throw new IllegalArgumentException("non-positive value");
    }
    super(value);
  }

}
```

Validate in constructor using validation before calling super()

454

FOREIGN FUNCTION & MEMORY API

[STABLE]

azul

JEP 454

FOREIGN FUNCTION & MEMORY API

- Enables Java programs to interoperate with code outside the JVM
- New linker option to pass heap segments to downcall method handles
- New Enable-Native-Access manifest attribute for JAR files
  (allows code in executable JAR files to call restricted methods without the --enable-native-access command line option)
- Build C-language function descriptors programmatically
- Enhanced support for variable-length arrays in native memory
- Support for arbitrary charsets for native strings

JEP 454

FOREIGN FUNCTION API

```c
#include <stdio.h>

int add(int a, int b) {
    return a + b;
}
```

libadd.c code to add two integers

```
> gcc -shared -o libadd.so -fPIC libadd.c
```

Compile to a shared library named libadd.so

JEP 454

FOREIGN FUNCTION API

```
public static void main(String[] args) {
  try (var arena = Arena.ofConfined()) {
    var lib          = SymbolLookup.libraryLookup(Path.of("libadd.so"), arena);
    var linker       = Linker.nativeLinker();
    var fd           = FunctionDescriptor.of(ValueLayout.JAVA_INT, ValueLayout.JAVA_INT, ValueLayout.JAVA_INT);
    var addFunc      = lib.find("add").get();
    var methodHandle = linker.downcallHandle(addFunc, fd);
    var sum          = methodHandle.invoke(1, 2);

    System.out.println("sum = " + sum);

  } catch (Throwable e) {
    throw new RuntimeException(e);
  }
}
```

Load the library, lookup the method and invoke it

JEP 456

UNNAMED VARIABLES & PATTERNS

[STABLE]

azul

JEP 456

UNNAMED VARIABLES & PATTERNS

- Allows to substitute with the underscore character _ for
  - Unnecessary type and name of a record component in pattern matching
  - Variables that must be declared but will not be used

JEP 456

UNNAMED VARIABLES & PATTERNS

```
static int count(List<String> names) {
  int total = 0;
  for (String name : names) {
    total++;
  }
  return total;
}
```

name is unused

JEP 456

UNNAMED VARIABLES & PATTERNS

```
static int count(List<String> names) {
  int total = 0;
  for (var _ : names) {
    total++;
  }
  return total;
}
```

and can be replaced with _

JEP 456

UNNAMED VARIABLES & PATTERNS

```
sealed abstract class Ball permits RedBall, BlueBall, GreenBall { }

final class RedBall   extends Ball { }
final class BlueBall  extends Ball { }
final class GreenBall extends Ball { }


public static void filter(Ball ball) {
    switch (ball) {
        case RedBall   red   -> process(ball);
        case GreenBall green -> process(ball);
        case BlueBall  blue  -> process(ball);
    }
}
```

red, green and blue are unused

JEP 456

UNNAMED VARIABLES & PATTERNS

```
sealed abstract class Ball permits RedBall, BlueBall, GreenBall { }

final class RedBall   extends Ball { }
final class BlueBall  extends Ball { }
final class GreenBall extends Ball { }


public static void filter(Ball ball) {
    switch (ball) {
        case RedBall   _ -> process(ball);
        case GreenBall _ -> process(ball);
        case BlueBall  _ -> process(ball);
    }
}
```

and can be replaced with _

CLASS-FILE API

[PREVIEW]

azul

JEP 457

CLASS-FILE API

- A standard class-file API which will evolve with the class-file format.
  (avoiding dependencies from third-party libraries)
- Introducing CodeModels
- Generate class files with builders
- Transforming class files

JEP 457

CREATE CLASS USING CODEBUILDER

- JVM is stack based machine
- Many instructions deal with pushing/popping from the operand stack
- For a "Hello World" example we will need 4 bytecode instructions

JEP 457

CREATE CLASS USING CODEBUILDER

INSTRUCTION	STACK BEFORE	STACK AFTER	EXAMPLE	DESCRIPTION
getstatic	`...,`	`..., value`	`getstatic Ljava/lang/System; out`	Pushes a reference to the System.out instance onto the stack
ldc	`...,`	`..., value`	`ldc "Hello World"`	Pushes the constant "Hello World" onto the stack
invokevirtual	`..., objectref, [arg1, arg2, argN]`	`..., [return value]`	`invokevirtual Ljava/io/PrintStream; println(Ljava/lang/String;)V`	Pops the reference to System.out and the "Hello World" string, and executes println
return	`...,`	`empty`	`return`	Returns from a method

JEP 457

CREATE CLASS USING CODEBUILDER

```
ClassFile helloWorldClass = ClassFile.of();

helloWorldClass.buildTo(Path.of("HelloWorld.class"), ClassDesc.of("HelloWorld"), classBuilder -> {

  classBuilder.withMethodBody("main", MethodTypeDesc.ofDescriptor("([Ljava/lang/String;)V"),
                              ACC_PUBLIC | ACC_STATIC, codeBuilder -> {









    });
});
```

Push a reference to the System.out instance onto the stack

JEP 457

CREATE CLASS USING CODEBUILDER

```
ClassFile helloWorldClass = ClassFile.of();

helloWorldClass.buildTo(Path.of("HelloWorld.class"), ClassDesc.of("HelloWorld"), classBuilder -> {

  classBuilder.withMethodBody("main", MethodTypeDesc.ofDescriptor("([Ljava/lang/String;)V"),
                              ACC_PUBLIC | ACC_STATIC, codeBuilder -> {

    codeBuilder.getstatic(ClassDesc.of("java.lang.System"), "out", ClassDesc.of("java.io.PrintStream"))
               .ldc("Hello World")



    });
});
```

Push the constant "Hello World" onto the stack

JEP 457

CREATE CLASS USING CODEBUILDER

```
ClassFile helloWorldClass = ClassFile.of();

helloWorldClass.buildTo(Path.of("HelloWorld.class"), ClassDesc.of("HelloWorld"), classBuilder -> {

  classBuilder.withMethodBody("main", MethodTypeDesc.ofDescriptor("([Ljava/lang/String;)V"),
                              ACC_PUBLIC | ACC_STATIC, codeBuilder -> {

    codeBuilder.getstatic(ClassDesc.of("java.lang.System"), "out", ClassDesc.of("java.io.PrintStream"))
               .ldc("Hello World")
               .invokevirtual(ClassDesc.of("java.io.PrintStream"), "println",
                              MethodTypeDesc.ofDescriptor("(Ljava/lang/Object;)V"))

    });
});
```

Pop reference to System.out and "Hello World" string and execute println

JEP 457

CREATE CLASS USING CODEBUILDER

```
ClassFile helloWorldClass = ClassFile.of();

helloWorldClass.buildTo(Path.of("HelloWorld.class"), ClassDesc.of("HelloWorld"), classBuilder -> {

  classBuilder.withMethodBody("main", MethodTypeDesc.ofDescriptor("([Ljava/lang/String;)V"),
                              ACC_PUBLIC | ACC_STATIC, codeBuilder -> {

    codeBuilder.getstatic(ClassDesc.of("java.lang.System"), "out", ClassDesc.of("java.io.PrintStream"))
               .ldc("Hello World")
               .invokevirtual(ClassDesc.of("java.io.PrintStream"), "println",
                              MethodTypeDesc.ofDescriptor("(Ljava/lang/Object;)V"))
               .return_();
    });
});
```

Return from method

JEP 457

CREATE CLASS USING CODEBUILDER

```
ClassFile helloWorldClass = ClassFile.of();

helloWorldClass.buildTo(Path.of("HelloWorld.class"), ClassDesc.of("HelloWorld"), classBuilder -> {

  classBuilder.withMethodBody("main", MethodTypeDesc.ofDescriptor("([Ljava/lang/String;)V"),
                              ACC_PUBLIC | ACC_STATIC, codeBuilder -> {

    codeBuilder.getstatic(ClassDesc.of("java.lang.System"), "out", ClassDesc.of("java.io.PrintStream"))
               .ldc("Hello World")
               .invokevirtual(ClassDesc.of("java.io.PrintStream"), "println",
                              MethodTypeDesc.ofDescriptor("(Ljava/lang/Object;)V"))
               .return_();
    });
});
```

JEP 457

CREATE CLASS USING CODEBUILDER

```
public class HelloWorld {
    public static void main(String[] var0) {
        System.out.println("Hello World");
    }
}
```

Decompiled HelloWorld.class file

JEP 457

CREATE CLASS USING CODEBUILDER

```
hansolo@Falcon : ls
```

JEP 457

CREATE CLASS USING CODEBUILDER

```
hansolo@Falcon : ls
HelloWorld.class   out   src
hansolo@Falcon :
```

JEP 457

CREATE CLASS USING CODEBUILDER

```
hansolo@Falcon : ls
HelloWorld.class   out   src
hansolo@Falcon : java -version
```

azul

JEP 457

CREATE CLASS USING CODEBUILDER

```
hansolo@Falcon : ls
HelloWorld.class   out   src
hansolo@Falcon : java -version
openjdk version "22-beta" 2024-03-19
OpenJDK Runtime Environment Zulu22+67-CA (build 22-beta+28)
OpenJDK 64-Bit Server VM Zulu22+67-CA (build 22-beta+28, mixed mode, sharing)
hansolo@Falcon :
```

JEP 457

CREATE CLASS USING CODEBUILDER

```
hansolo@Falcon : ls
HelloWorld.class   out   src
hansolo@Falcon : java -version
openjdk version "22-beta" 2024-03-19
OpenJDK Runtime Environment Zulu22+67-CA (build 22-beta+28)
OpenJDK 64-Bit Server VM Zulu22+67-CA (build 22-beta+28, mixed mode, sharing)
hansolo@Falcon : java HelloWorld
```

JEP 457

CREATE CLASS USING CODEBUILDER

```
hansolo@Falcon : ls
HelloWorld.class   out   src
hansolo@Falcon : java -version
openjdk version "22-beta" 2024-03-19
OpenJDK Runtime Environment Zulu22+67-CA (build 22-beta+28)
OpenJDK 64-Bit Server VM Zulu22+67-CA (build 22-beta+28, mixed mode, sharing)
hansolo@Falcon : java HelloWorld
Hello World
hansolo@Falcon :
```

458

LAUNCH MULTI-FILE SOURCE-CODE PROGRAMS

[STABLE]

azul

JEP 458

LAUNCH MULTI-FILE SOURCE-CODE PROGRAMS

- Since JDK 11 one can run a `.java` file directly using the java launcher
  (no need to compile the file with javac first, it will be compiled in memory and then executed)
- Now this also works with multiple `.java` files
- Only `.java` files that are directly referenced in the initial `.java` file will be compiled
- The java launcher requires source files organized according to their package structure
- Makes it easier to getting started with Java

JEP 458

LAUNCH MULTI-FILE SOURCE-CODE PROGRAMS

```
public class Main {

  public static void main(String[] args) {
    String text = Helper.check(args);
    System.out.println("Hello " + text);
  }
}

public class Helper {

  public static final String check(final String[] args) {
    return args.length == 0 ? "World" : args[0];
  }
}
```

JEP 458

LAUNCH MULTI-FILE SOURCE-CODE PROGRAMS

```
hansolo@Falcon : java -version
```

azul

JEP 458

LAUNCH MULTI-FILE SOURCE-CODE PROGRAMS

```
hansolo@Falcon : java -version
openjdk version "22-beta" 2024-03-19
OpenJDK Runtime Environment Zulu22+67-CA (build 22-beta+28)
OpenJDK 64-Bit Server VM Zulu22+67-CA (build 22-beta+28, mixed mode, sharing)
hansolo@Falcon :
```

JEP 458

LAUNCH MULTI-FILE SOURCE-CODE PROGRAMS

```
hansolo@Falcon : java -version
openjdk version "22-beta" 2024-03-19
OpenJDK Runtime Environment Zulu22+67-CA (build 22-beta+28)
OpenJDK 64-Bit Server VM Zulu22+67-CA (build 22-beta+28, mixed mode, sharing)
hansolo@Falcon : java Main.java
```

azul

JEP 458

LAUNCH MULTI-FILE SOURCE-CODE PROGRAMS

```
hansolo@Falcon : java -version
openjdk version "22-beta" 2024-03-19
OpenJDK Runtime Environment Zulu22+67-CA (build 22-beta+28)
OpenJDK 64-Bit Server VM Zulu22+67-CA (build 22-beta+28, mixed mode, sharing)
hansolo@Falcon : java Main.java
Hello World
hansolo@Falcon :
```

azul

JEP 458

LAUNCH MULTI-FILE SOURCE-CODE PROGRAMS

```
hansolo@Falcon : java -version
openjdk version "22-beta" 2024-03-19
OpenJDK Runtime Environment Zulu22+67-CA (build 22-beta+28)
OpenJDK 64-Bit Server VM Zulu22+67-CA (build 22-beta+28, mixed mode, sharing)
hansolo@Falcon : java Main.java
Hello World
hansolo@Falcon : java Main.java Gerrit
```

JEP 458

LAUNCH MULTI-FILE SOURCE-CODE PROGRAMS

```
hansolo@Falcon : java -version
openjdk version "22-beta" 2024-03-19
OpenJDK Runtime Environment Zulu22+67-CA (build 22-beta+28)
OpenJDK 64-Bit Server VM Zulu22+67-CA (build 22-beta+28, mixed mode, sharing)
hansolo@Falcon : java Main.java
Hello World
hansolo@Falcon : java Main.java Gerrit
Hello Gerrit
hansolo@Falcon :
```

JEP 458

"Shebang" files

```
#!/path/to/java --source 10

public class Main {
    public static void main(String[] args) {
        System.out.println("Hello " + (args.length == 0 ? "World" : args[0]));
    }
}
```

Save to file "hello"

JEP 458

"Shebang" files

```
hansolo@Falcon : chmod +x hello
hansolo@Falcon :
```

azul

JEP 458

"Shebang" files

```
hansolo@Falcon : chmod +x hello
hansolo@Falcon : ./hello
Hello World
hansolo@Falcon :
```

azul

JEP 458

"Shebang" files

```
hansolo@Falcon : chmod +x hello
hansolo@Falcon : ./hello
Hello World
hansolo@Falcon : ./hello Gerrit
Hello World
hansolo@Falcon :
```

459

STRING TEMPLATES

[SECOND PREVIEW]

azul

JEP 459

STRING TEMPLATES

- Enables developers to concatenate text with embedded expressions
- Reduces boilerplate code
- Eliminates risk of string interpolation

JEP 459

STRING TEMPLATES (e.g. FORMATTED JSON OUTPUT)

```
public record Person(String firstName, String lastName, String street, String zip, String city) {

  @Override public String toString() {
    return new StringBuilder().append("{\n")
                              .append("  \"firstName\":\"").append(firstName).append("\",\n")
                              .append("  \"lastName\":\"").append(lastName).append("\",\n")
                              .append("  \"street\":\"").append(street).append("\",\n")
                              .append("  \"zip\":\"").append(zip).append("\",\n")
                              .append("  \"city\":\"").append(city).append("\"\n")
                              .append("}")
                              .toString();
  }
}
```

JEP 459

STRING TEMPLATES (e.g. FORMATTED JSON OUTPUT)

```
public record Person(String firstName, String lastName, String street, String zip, String city) {

  @Override public String toString() {
    return STR."""
        {
          "firstName":"\{firstName}",
          "lastName":"\{lastName}",
          "street":""\{street}",
          "zip":"\{zip}",
          "city":"\{city}"
        }
        """;

  }
}
```

⚠ Template Processors will be removed ⚠

460

VECTOR API

[INCUBATOR]

azul

JEP 460

VECTOR API

- Helps to increase performance of vector computations.
  (which are superior to the equivalent scalar alternatives)
- Short version…process more data within a single instruction
- The JVM does auto-vectorization already for some time
  (can be switched off by -XX:-UseSuperWord)
- But now you can decide how to use vectorization

JEP 460

VECTOR API (e.g. add two arrays)

```
int[] a = { 1, 2, 3, 4, 5, 6, 7, 8 }
int[] b = { 8, 7, 6, 5, 4, 3, 2, 1 }
int[] c = new int[8];

for(int i = 0 ; i < 8 ; i++) {
    c[i] = a[i] + b[i];
}
```

Every addition will
take 1 CPU cycle
(here 8 CPU cycles)

SISD
(Single Instruction, Single Data)

JEP 460

VECTOR API (e.g. add two arrays)

```
int[] a = { 1, 2, 3, 4, 5, 6, 7, 8 }
int[] b = { 8, 7, 6, 5, 4, 3, 2, 1 }
int[] c = new int[8];

for(int i = 0 ; i < 8 ; i++) {
    c[i] = a[i] + b[i];
}
```

CPUs have special registers called SIMD
(Single Instruction, Multiple Data)

Operations on SIMD registers only take 1 CPU cycle

JEP 460

VECTOR API (e.g. add two arrays)

```
int[] { 1, 2, 3, 4, 5, 6, 7, 8 }
int[] { 8, 7, 6, 5, 4, 3, 2, 1 }
```

1 Integer == 32 bit

azul

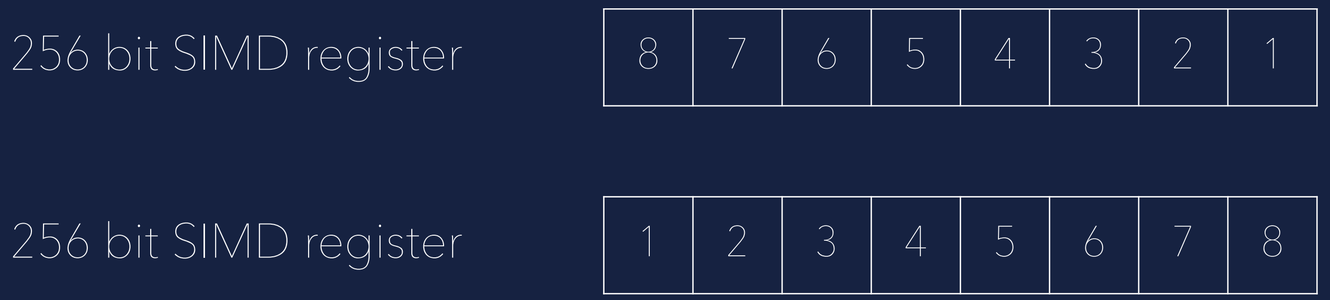
JEP 460

VECTOR API (e.g. add two arrays)

```
int[] { 1, 2, 3, 4, 5, 6, 7, 8 }
int[] { 8, 7, 6, 5, 4, 3, 2, 1 }
```

1 Integer == 32 bit

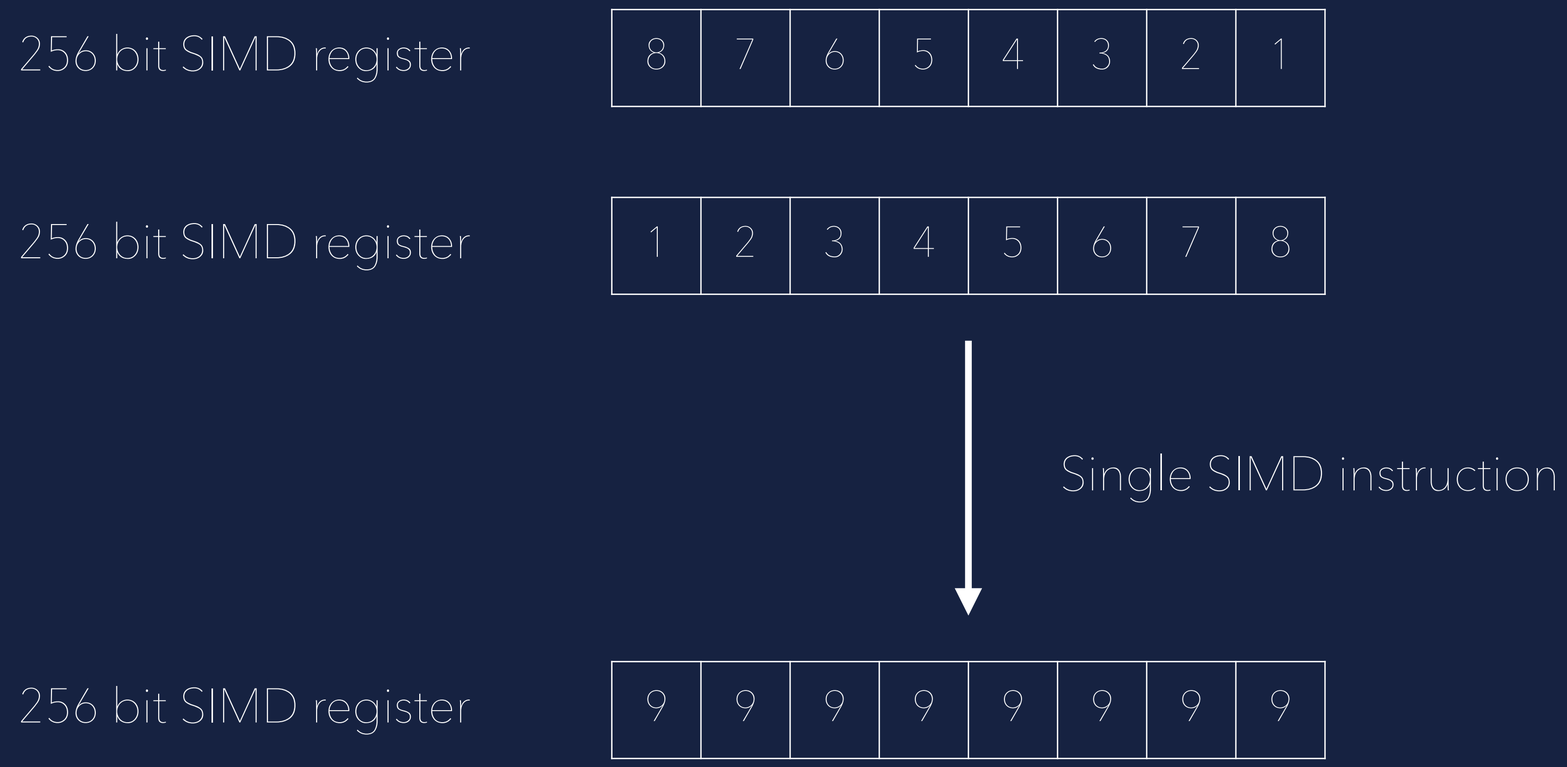
Whole array fits into 1 SIMD register



azul

JEP 460

VECTOR API (e.g. add two arrays)



JEP 460

VECTOR API (e.g. add two arrays)

```
int[] a = { 1, 2, 3, 4, 5, 6, 7, 8 }
int[] b = { 8, 7, 6, 5, 4, 3, 2, 1 }
int[] c = new int[8];

// Scalar addition
for(int i = 0 ; i < 8 ; i++) {
    c[i] = a[i] + b[i];
}


// Vectorized SIMD addition
IntVector vectorA = IntVector.fromArray(IntVector.SPECIES_PREFERRED, a, 0);
IntVector vectorB = IntVector.fromArray(IntVector.SPECIES_PREFERRED, b, 0);
IntVector vectorC = aVector.add(bVector);

vectorC.intoArray(c, 0);
```

JEP 460

VECTOR API

DATA THAT DOES NOT FIT INTO SIMD REGISTERS WILL BE STORED/RESTORED IN MAIN MEMORY -> SLOW !

JEP 460

VECTOR API (e.g. count how many cells of 1st array are ==, < or > than cells of 2nd array)

```
int lowerThan   = 0;
int equal       = 0;
int greaterThan = 0;

long start = System.nanoTime();

for (int i = 0; i < arraySize; i++) {
    if (a[i] == b[i]) {
        equal++;
    } else if (a[i] > b[i]) {
        greaterThan++;
    } else {
        lowerThan++;
    }
}

System.out.println("Scalar: " + arraySize + "  " + equal + "  " + lowerThan + "   " + greaterThan + " -> " +
((System.nanoTime() - start) / 1_000_000) + "ms");
```

JEP 460

VECTOR API (e.g. count how many cells of 1st array are ==, < or > than cells of 2nd array)

```
VectorSpecies<Float> SPECIES     = FloatVector.SPECIES_PREFERRED;
int                  lowerThan   = 0;
int                  equal       = 0;
int                  greaterThan = 0;

long                 start       = System.nanoTime();

for (int i = 0; i < arraySize; i += SPECIES.length()) {
    FloatVector vectorA           = FloatVector.fromArray(SPECIES, a, i);
    FloatVector vectorB           = FloatVector.fromArray(SPECIES, b, i);
    int         lowerThanCounter = vectorA.lt(vectorB).trueCount();
    int         equalCounter     = vectorA.eq(vectorB).trueCount();
    equal       += equalCounter;
    lowerThan   += lowerThanCounter;
    greaterThan += SPECIES.length() - lowerThanCounter - equalCounter;
}

System.out.println("Vector: " + arraySize + "  " + equal + "  " + lowerThan + "   " + greaterThan + " -> " +
((System.nanoTime() - start) / 1_000_000) + "ms");
```

azul

JEP 460

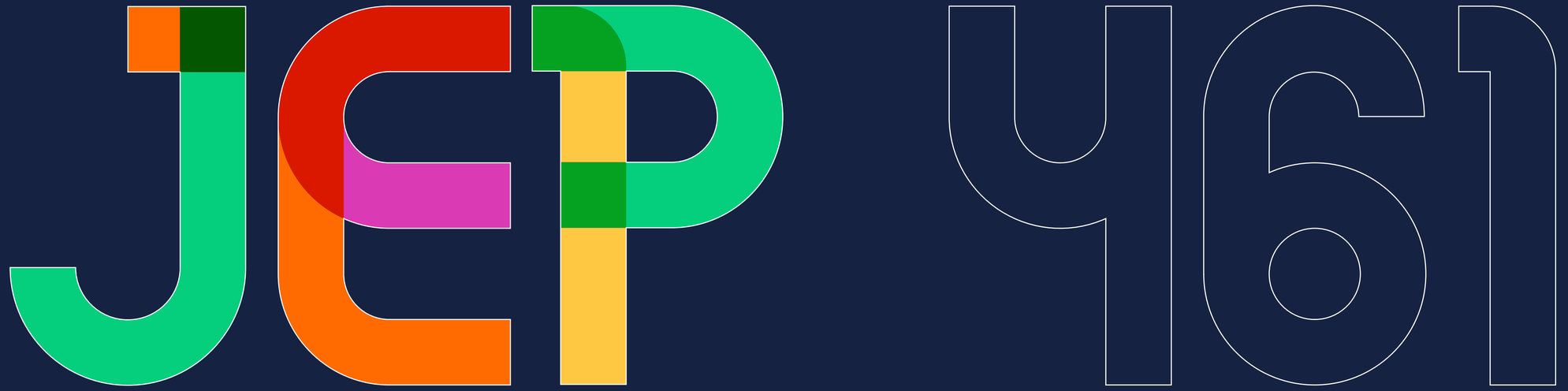
VECTOR API (e.g. count how many cells of 1st array are ==, < or > than cells of 2nd array)

```
int     arraySize = 134_217_728;
float[] a         = new float[arraySize];
float[] b         = new float[arraySize];

for (int i = 0 ; i < arraySize ; i++) {
    a[i] = RND.nextFloat(32);
    b[i] = RND.nextFloat(32);
}

scalar(arraySize, a, b);
vector(arraySize, a, b);


Scalar: 134217728  5   67104661   67113062 -> 661ms
Vector: 134217728  5   67104661   67113062 -> 60ms
```



STREAM GATHERERS

[PREVIEW]

azul

JEP 461

STREAM GATHERERS

- Stream Source -> provide elements
- Stream Intermediate operation -> transform elements
- Stream Terminal operation -> produce value or side effect
- How to intermediate operations like folds, unfolds, barriers, windowing...

JEP 461

STREAM GATHERERS

- Enhancement of Stream API with custom intermediate operations
- New operation Stream::gather(Gatherer)
- Gatherer can transform elements of a stream
  (four functions: initializer, integrator, combiner, finisher)
- Enable developers to create more flexible and expressive streams

JEP 461

STREAM GATHERERS (FIXED WINDOW)

How to convert to List<Point>?

"0,0,0,10,-20,20,80,20,100,10,100,0,0,0"

String with polygon coordinates

JEP 461

STREAM GATHERERS (FIXED WINDOW)

Fixed Window

'[0,0]0,10,-20,20,80,20,100,10,100,0,0,0"

String with polygon coordinates

```
new Point(0,0)
```

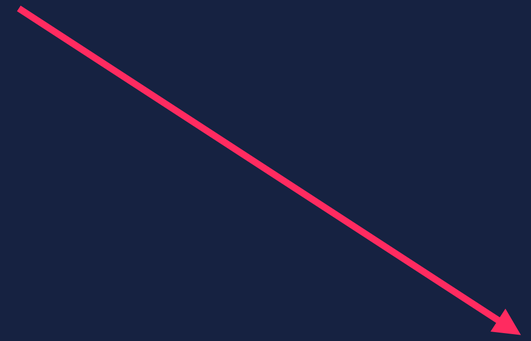
JEP 461

STREAM GATHERERS (FIXED WINDOW)

Fixed Window

"0,0[0,10]-20,20,80,20,100,10,100,0,0,0"

String with polygon coordinates



```
new Point(0,10)
```

JEP 461

STREAM GATHERERS (FIXED WINDOW)

Fixed Window

"0,0,0,10[-20,20]80,20,100,10,100,0,0,0"

String with polygon coordinates

```
new Point(-20,20)
```

JEP 461

STREAM GATHERERS (FIXED WINDOW)

Fixed Window

"0,0,0,10,-20,20[80,20]100,10,100,0,0,0"

String with polygon coordinates

```
new Point(80,20)
```

JEP 461

STREAM GATHERERS (FIXED WINDOW)

Fixed Window

"0,0,0,10,-20,20,80,20[100,10]100,0,0,0"

String with polygon coordinates

```
new Point(100,10)
```

JEP 461

STREAM GATHERERS (FIXED WINDOW)

Fixed Window

"0,0,0,10,-20,20,80,20,100,10[100,0]0,0"

String with polygon coordinates

```
new Point(100,0)
```

JEP 461

STREAM GATHERERS (FIXED WINDOW)

Fixed Window

"0,0,0,10,-20,20,80,20,100,10,100,0[0,0]"

String with polygon coordinates

```
new Point(0,0)
```

JEP 461

STREAM GATHERERS (e.g.convert string of polygon coordinates to List<Point>)

```
record Point(int x, int y) {};

// String containing polygon points
String              polygonPoints = "0,0,0,10,20,20,80,20,100,10,100,0,0,0";

// Create a list of points from the String
List<String>        numbers       = Arrays.stream(polygonPoints.split(","))
                                          .toList();

// Map Strings to Integer, create a fixed window of 2 and collect it to a List of List<Integer>
List<List<Integer>> windows       = numbers.stream()
                                           .mapToInt(n -> Integer.valueOf(n))
                                           .boxed()
                                           .gather(Gatherers.windowFixed(2))
                                           .toList();

// Map each List<Integer> to a List<Point>
List<Point>         points        = windows.stream()
                                           .map(p -> new Point(p.get(0), p.get(1)))
                                           .toList();
```

JEP 461

STREAM GATHERERS (e.g.convert string of polygon coordinates to List<Point>)

```
record Point(int x, int y) {};

// String containing polygon points
String polygonPoints = "0,0,0,10,20,20,80,20,100,10,100,0,0,0";

// Create a list of points from the String
List<Point> points = Arrays.stream(polygonPoints.split(","))
                           .mapToInt(n -> Integer.valueOf(n))
                           .boxed()
                           .gather(Gatherers.windowFixed(2))
                           .map(p -> new Point(p.get(0), p.get(1)))
                           .toList();
```

JEP 461

STREAM GATHERERS (e.g.convert string of polygon coordinates to List<Point>)

```
record Point(int x, int y) {};

// String containing polygon points
String polygonPoints = "0,0,0,10,20,20,80,20,100,10,100,0,0,0";

// Create a list of points from the String
List<Point> points = Arrays.stream(polygonPoints.split(","))
                           .mapToInt(n -> Integer.valueOf(n))
                           .boxed()
                           .gather(Gatherers.windowFixed(2))
                           .map(p -> new Point(p.get(0), p.get(1)))
                           .toList();

points.forEach(p -> System.out.println("x: " + p.x + ", "y: " + p.y));
```

```
x: 0, y: 0
x: 0, y: 10
x: -20, y: 20
x: 80, y: 20
x: 100, y: 10
x: 100, y: 0
x: 0, y: 0
```

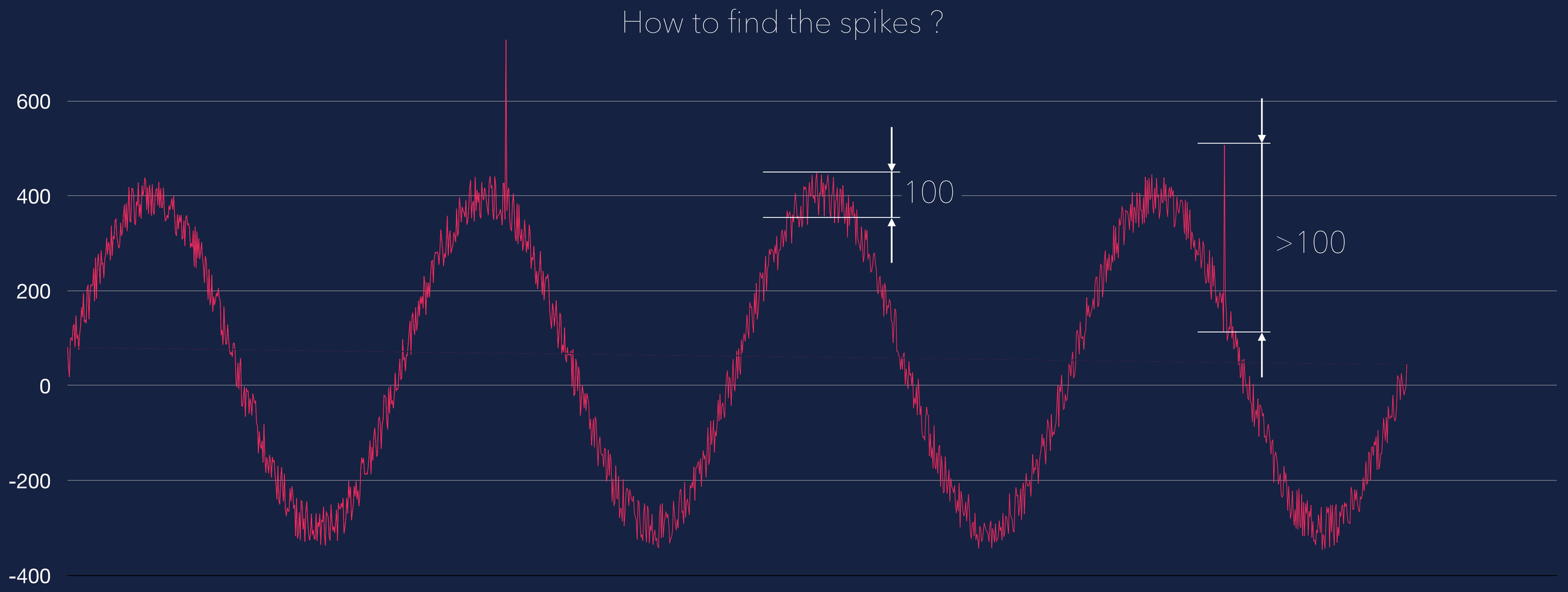
JEP 461

STREAM GATHERERS (SLIDING WINDOW)

azul

JEP 461

STREAM GATHERERS (SLIDING WINDOW)



azul

JEP 461

STREAM GATHERERS (SLIDING WINDOW)

Index	Timestamp	Value
450	526342694458125	410.1954269269
451	526342694458916	432.5650369737
452	526342694459666	371.6015057674
453	526342694460541	439.0635656007
454	526342694461250	427.2644502460
455	526342694465208	732.6881587951
456	526342694466666	409.0010799259
457	526342694467333	385.4080178193
458	526342694467916	416.3920385787
459	526342694468583	395.0509979460

azul

JEP 461

STREAM GATHERERS (SLIDING WINDOW)

Sliding Window

Index	Timestamp	Value
450	526342694458125	410.1954269269
451	526342694458916	432.5650369737
452	526342694459666	371.6015057674
453	526342694460541	439.0635656007
454	526342694461250	427.2644502460
455	526342694465208	732.6881587951
456	526342694466666	409.0010799259
457	526342694467333	385.4080178193
458	526342694467916	416.3920385787
459	526342694468583	395.0509979460

Δ = 22.37

azul

JEP 461

STREAM GATHERERS (SLIDING WINDOW)

Sliding Window

Index	Timestamp	Value
450	526342694458125	410.1954269269
451	526342694458916	432.5650369737
452	526342694459666	371.6015057674
453	526342694460541	439.0635656007
454	526342694461250	427.2644502460
455	526342694465208	732.6881587951
456	526342694466666	409.0010799259
457	526342694467333	385.4080178193
458	526342694467916	416.3920385787
459	526342694468583	395.0509979460

Δ = -60.96

azul

JEP 461

STREAM GATHERERS (SLIDING WINDOW)

Sliding Window

Index	Timestamp	Value
450	526342694458125	410.1954269269
451	526342694458916	432.5650369737
452	526342694459666	371.6015057674
453	526342694460541	439.0635656007
454	526342694461250	427.2644502460
455	526342694465208	732.6881587951
456	526342694466666	409.0010799259
457	526342694467333	385.4080178193
458	526342694467916	416.3920385787
459	526342694468583	395.0509979460

$\Delta = 67.46$

azul

JEP 461

STREAM GATHERERS (SLIDING WINDOW)

Sliding Window

Index	Timestamp	Value
450	526342694458125	410.1954269269
451	526342694458916	432.5650369737
452	526342694459666	371.6015057674
453	526342694460541	439.0635656007
454	526342694461250	427.2644502460
455	526342694465208	732.6881587951
456	526342694466666	409.0010799259
457	526342694467333	385.4080178193
458	526342694467916	416.3920385787
459	526342694468583	395.0509979460

Δ = -11.80

azul

JEP 461

STREAM GATHERERS (SLIDING WINDOW)

Sliding Window

Index	Timestamp	Value
450	526342694458125	410.1954269269
451	526342694458916	432.5650369737
452	526342694459666	371.6015057674
453	526342694460541	439.0635656007
454	526342694461250	427.2644502460
455	526342694465208	732.6881587951
456	526342694466666	409.0010799259
457	526342694467333	385.4080178193
458	526342694467916	416.3920385787
459	526342694468583	395.0509979460

Δ = 305.42

azul

JEP 461

STREAM GATHERERS (SLIDING WINDOW)

Index	Timestamp	Value
450	526342694458125	410.1954269269
451	526342694458916	432.5650369737
452	526342694459666	371.6015057674
453	526342694460541	439.0635656007
454	526342694461250	427.2644502460
455	526342694465208	732.6881587951
456	526342694466666	409.0010799259
457	526342694467333	385.4080178193
458	526342694467916	416.3920385787
459	526342694468583	395.0509979460

Sliding Window

Δ = 305.42 (>100)

azul

JEP 461

STREAM GATHERERS (SLIDING WINDOW)

Index	Timestamp	Value
450	526342694458125	410.1954269269
451	526342694458916	432.5650369737
452	526342694459666	371.6015057674
453	526342694460541	439.0635656007
454	526342694461250	427.2644502460
455	526342694465208	732.6881587951
456	526342694466666	409.0010799259
457	526342694467333	385.4080178193
458	526342694467916	416.3920385787
459	526342694468583	395.0509979460

Sliding Window

Δ = -323.69

azul

JEP 461

STREAM GATHERERS (SLIDING WINDOW)

Index	Timestamp	Value
450	526342694458125	410.1954269269
451	526342694458916	432.5650369737
452	526342694459666	371.6015057674
453	526342694460541	439.0635656007
454	526342694461250	427.2644502460
455	526342694465208	732.6881587951
456	526342694466666	409.0010799259
457	526342694467333	385.4080178193
458	526342694467916	416.3920385787
459	526342694468583	395.0509979460

Sliding Window

Δ = -23.59

azul

JEP 461

STREAM GATHERERS (SLIDING WINDOW)

Index	Timestamp	Value
450	526342694458125	410.1954269269
451	526342694458916	432.5650369737
452	526342694459666	371.6015057674
453	526342694460541	439.0635656007
454	526342694461250	427.2644502460
455	526342694465208	732.6881587951
456	526342694466666	409.0010799259
457	526342694467333	385.4080178193
458	526342694467916	416.3920385787
459	526342694468583	395.0509979460

Sliding Window

$\Delta = 30.98$

azul

JEP 461

STREAM GATHERERS (SLIDING WINDOW)

Index	Timestamp	Value
450	526342694458125	410.1954269269
451	526342694458916	432.5650369737
452	526342694459666	371.6015057674
453	526342694460541	439.0635656007
454	526342694461250	427.2644502460
455	526342694465208	732.6881587951
456	526342694466666	409.0010799259
457	526342694467333	385.4080178193
458	526342694467916	416.3920385787
459	526342694468583	395.0509979460

Sliding Window

Δ = -21.34

azul

JEP 461

STREAM GATHERERS (e.g. find spikes in data series)

```
record Data(int index, long timestamp, double value) {};
record Spike(Data data, double delta) {};

// List with data points
List<Data> timeseries = ...

// Threshold for delta between values
double threshold = 100;

// Find all spikes in timeseries and save them to list
List<Spike> spikes = timeseries.stream()
                               .gather(Gatherers.windowSliding(2))
                               .filter(dataList -> (dataList.get(1).value - dataList.get(0).value) > threshold
                               .map(dataList -> new Spike(dataList.get(1), (dataList.get(1).value - dataList.get(0).value)))
                               .toList();
```

JEP 461

STREAM GATHERERS (e.g. find spikes in data series)

```
record Data(int index, long timestamp, double value) {};
record Spike(Data data, double delta) {};

// List with data points
List<Data> timeseries = ...;

// Threshold for delta between values
double threshold = 100;

// Find all spikes in timeseries and save them to list
List<Spike> spikes = timeseries.stream()
                               .gather(Gatherers.windowSliding(2))
                               .filter(dataList -> (dataList.get(1).value - dataList.get(0).value) > threshold
                               .map(dataList -> new Spike(dataList.get(1), (dataList.get(1).value - dataList.get(0).value)))
                               .toList();

spikes.forEach(spike -> System.out.println("index: " + spike.data.index + ", timestamp: " +
                spike.data.timestamp + ", value: " + spike.data.value + ", delta: " + spike.delta));

index: 371, timestamp: 528673514912333, value: 314.7681441638435, delta: 189.7209433886203
index: 1128, timestamp: 528673515412958, value: 494.44187909367434, delta: 148.90840118680734
```

462

STRUCTURED CONCURRENCY

[SECOND PREVIEW]

azul

JEP 462

STRUCTURED CONCURRENCY

- Useful extension to observe and manage virtual threads
- Subtasks will be forked and then joined in the parent tasks code block
- Eliminates risks related to cancellation, shutdown and promoting reliability of concurrent applications

JEP 462

STRUCTURED CONCURRENCY

```
public Response fetch(long id) throws ExecutionException, InterruptedException {
  Future<AccountDetails> accountDetailsFuture = executorService.submit(() -> getAccountDetails(id));
  Future<List<Account>>  linkedAccountsFuture = executorService.submit(() -> fetchLinkedAccounts(id));
  Future<UserDetails>    userDetailsFuture    = executorService.submit(() -> fetchUserDetails(id));

  // Start threads independently
  return new Response(accountDetailsFuture.get(), linkedAccountsFuture.get(), userDetailsFuture.get());
}
```

JEP 462

STRUCTURED CONCURRENCY

```
public Response fetch(long id) {
  try (var scope = new StructuredTaskScope.ShutdownOnFailure()) {
    Supplier<AccountDetails> accountDetailsFuture = scope.fork(() -> getAccountDetails(id));
    Supplier<List<Account>>  linkedAccountsFuture = scope.fork(() -> fetchLinkedAccounts(id));
    Supplier<UserDetails>    userDetailsFuture    = scope.fork(() -> fetchUserDetails(id));

    scope.join();                                  // Join all subtasks
    scope.throwIfFailed(RuntimeException::new);  // Handle error when any subtask fails

    // Response is received from all workers, subtasks have completed by now so process the result
    return new Response(accountDetailsFuture.get(), linkedAccountsFuture.get(), userDetailsFuture.get());
  } catch (InterruptedException e) {
    throw new RuntimeException(e);
  }
}
```

JEP 462

STRUCTURED CONCURRENCY

- Bind blocks of concurrent code to a scope
- Preserve the relationship between the tasks
- Subtasks are executed in their own thread, forked individually and joined as a unit
- Subtasks perform action in behalf of a task
- Task awaits the result of subtasks and observes them for failures

463

IMPLICIT DECLARED CLASSES & INSTANCE MAIN METHODS

[SECOND PREVIEW]

azul

JEP 463

IMPLICITLY DECLARED CLASSED & INSTANCE MAIN METHODS

- Enables developers to write simple programs
- Simplifies learning Java

JEP 463

IMPLICITLY DECLARED CLASSED & INSTANCE MAIN METHODS

```
public class Main {

  public static void main() {
    System.out.println("Hello World");
  }

}
```

JEP 463

IMPLICITLY DECLARED CLASSED & INSTANCE MAIN METHODS

```
public class Main {

  void main() {
    System.out.println("Hello World");
  }

}
```

JEP 463

IMPLICITLY DECLARED CLASSED & INSTANCE MAIN METHODS

```
void main() {
  System.out.println("Hello World");
}
```

azul

SCOPED VALUES

[SECOND PREVIEW]

azul

JEP 464

SCOPED VALUES

- Enables developers to share immutable data within and accross threads
  (should be preferred to thread-local variables)

azul

JEP 464

SCOPED VALUES

```
class Server {
    private void serve(Request request) {
        ...
        User user = authenticateUser(request);
        restAdapter.processRequest(request, user);
        ....
    }
}

class RestAdapter {
    public void processRequest(Request request, User loggedInUser) {
        ...
        UUID id = extractId(request);
        useCase.invoke(id, loggedInUser);
        ....
    }
}
```

User forwarded to RestAdapter

JEP 464

SCOPED VALUES

```
class UseCase {
    public void invoke(UUID id, User loggedInUser) {
        ...
        Data data = repository.getData(id, loggedInUser);
        ....
    }
}

class Repository {
    public Data getData(UUID id, User loggedInUser) {
        ...
        Data data = findById(id);
        if (loggedInUser.isAdmin()) {
            enrichDataWithAdminInfos(data);
        }
        ....
    }
}
```

User forwarded to RestAdapter without using it it UseCase

Only time loggedInUser is used

JEP 464

SCOPED VALUES

```
class Server {
    public final static ScopedValue<User> LOGGED_IN_USER = ScopedValue.newInstance();
    private void serve(Request request) {
        ...
        User user = authenticateUser(request);
        ScopedValue.where(LOGGED_IN_USER, loggedInUser)
                   .run(() -> restAdapter.processRequest(request));
        ....
    }
}

class RestAdapter {
    public void processRequest(Request request) {
        ...
        UUID id = extractId(request);
        useCase.invoke(id);
        ....
    }
}
```

Bind scoped value to user object and make it available to the RestAdapter

No need to forward the user

JEP 464

SCOPED VALUES

```
class UseCase {
    public void invoke(UUID id) {
        ...
        Data data = repository.getData(id);
        ....
    }
}

class Repository {
    public Data getData(UUID id) {
        ...
        Data data = findById(id);
        User loggedInUser = Server.LOGGED_IN_USER.get();
        if (loggedInUser.isAdmin()) {
            enrichDataWithAdminInfos(data);
        }
        ....
    }
}
```

No need to forward the user

Get the logged in user from the scope value in the Server

JEP 464

SCOPED VALUES

NO NEED TO PASS VARIABLES THROUGH METHODS LEADS TO BETTER READABILITY


CODE

CODE

https://github.com/HanSolo/jdk22

azul

azul