Talleres UTP 2024

Java 21: Overview

Mg. Díaz Sánchez Fernando

Universidad Tecnológica del Perú

Imagen extraída de: https://static.vecteezy.com/system/resources/previews/004/973/162/original/web-design-and-development-isometric-illustrations-web-design-development-concept-programming-language-program-code-big-data-processing-on-laptop-screen-illustration-vector.jpg

Java 21: Overview

Universidad Tecnológica del Perú

OBJETIVOS DEL TALLER

Familiarizarse con las principales novedades de Java 21 para aplicarlas en el desarrollo de software

Contenido

1. Novedades en Java 21
2. Pattern Matching for Switch
3. Record Patterns
4. Sequenced Collections
5. Virtual Threads

Antes de empezar

Universidad Tecnológica del Perú

1. Link de descarga (Amazon Corretto)

https://docs.aws.amazon.com/corretto/latest/corretto-21-ug/downloads-list.html

2. Código para la sesión de hoy

https://github.com/utpcix/java21_overview.git

Novedades en Java 21

Universidad Tecnológica del Perú

Evolución desde Java 8

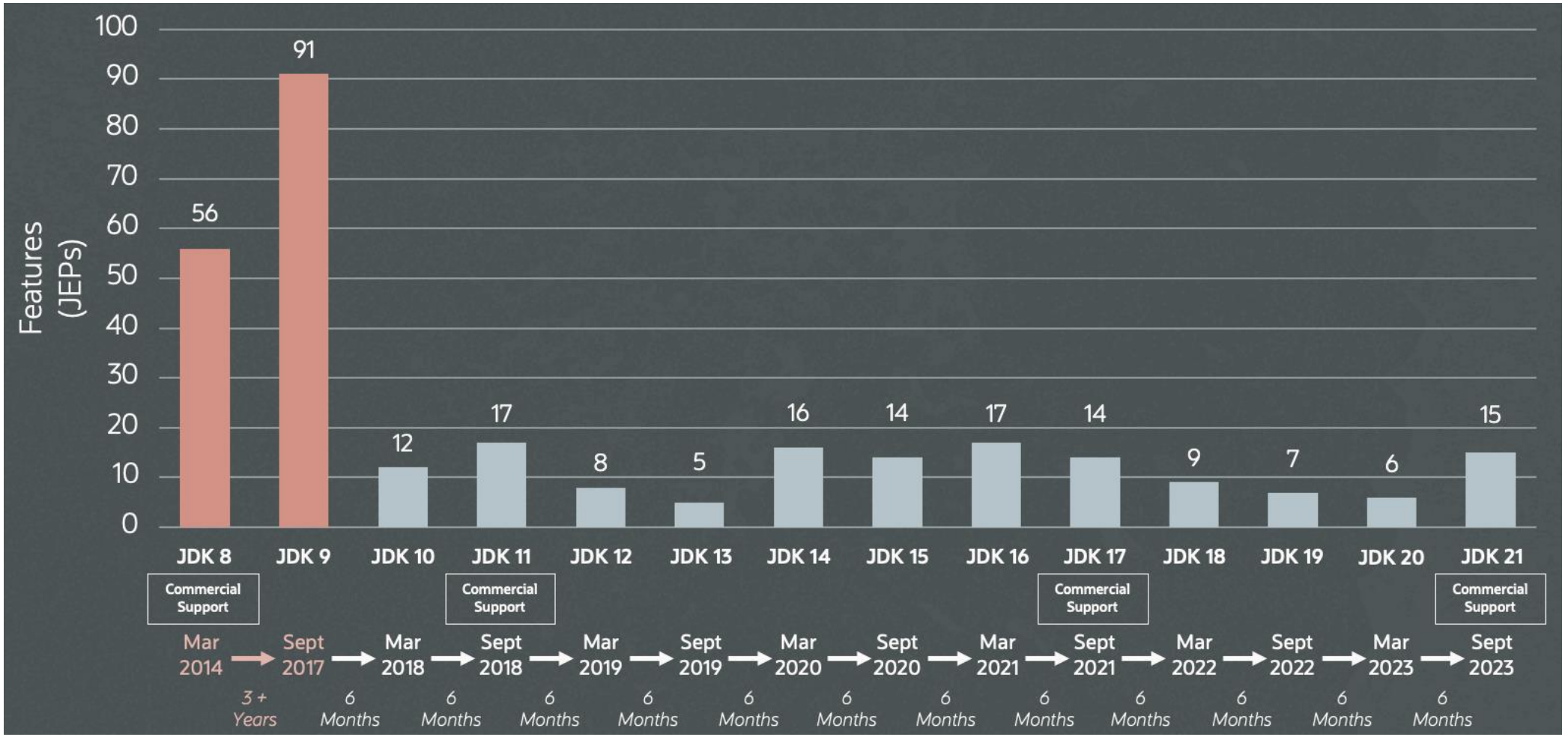


Imagen extraída de: https://inside.java/images/blog/21/JepPerRelease.png

Novedades en Java 21

Universidad Tecnológica del Perú

Fecha de lanzamiento: 19 Set 2023

Soporte LTS: Enero 2032

Principales características (No Previews)

- API Key Encapsulation Mechanism
- Virtual Threads
- Generational ZGC
- Record Patterns
- Pattern Matching for Switch
- Sequenced Collections

Pattern Matching for Switch

Universidad Tecnológica del Perú

- JEP 433: https://openjdk.org/jeps/433
- Mejora sustancialmente la sintaxis de la estructura switch
- Permite extender las expresiones de comparación con distintos patrones
- Ahora detecta valores nulos
- Elimina la necesidad de usar un default en algunos casos

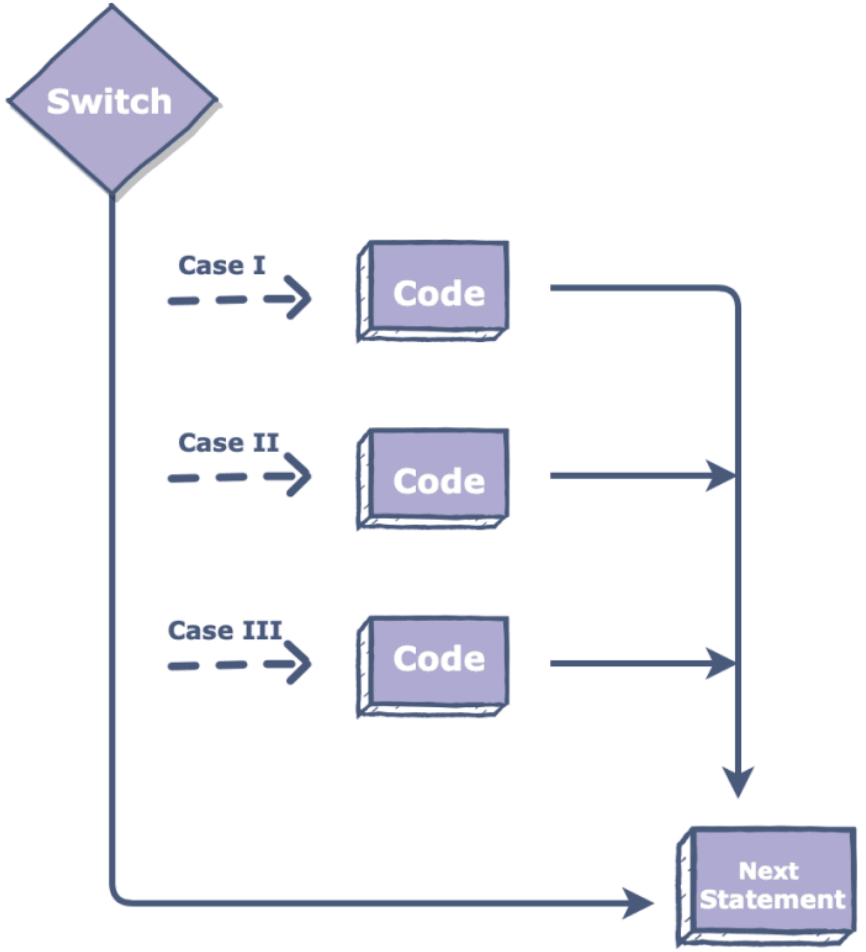


Imagen extraída de: https://thepracticaldev.s3.amazonaws.com/i/e87oxfdbji42pjxarbbr.png

Pattern Matching for Switch

Detección de tipos de datos en clausula case

```java
public static String showDataType(Object object){
    // Pattern Matching for data types
    return switch (object){
        case Integer i -> String.format("int %d",i);
        case String s -> String.format("String %s",s);
        case null -> String.format("Null Info");
        case Ticket t -> String.format("Ticket %d:%s", t.id(), t.event());
        default -> String.format("Unknown");
    };
}
```

Pattern Matching for Switch

Universidad Tecnológica del Perú

Mejora de patrones con Guards Conditions

```
public static String rubrica(Integer nota){
    // Using Guards Condition
    return switch (nota){
        case Integer n when (n>= 0 && n<=5) -> "Muy Bajo";
        case Integer n when (n> 5 && n<=10) -> "Bajo";
        case Integer n when (n> 10 && n<=15) -> "Regular";
        case Integer n when (n> 15 && n<=20) -> "Alto";
        default -> "Fuera del rango";
    };
}
```

Pattern Matching for Switch

Eliminación de clausula default mediante Sealing Class

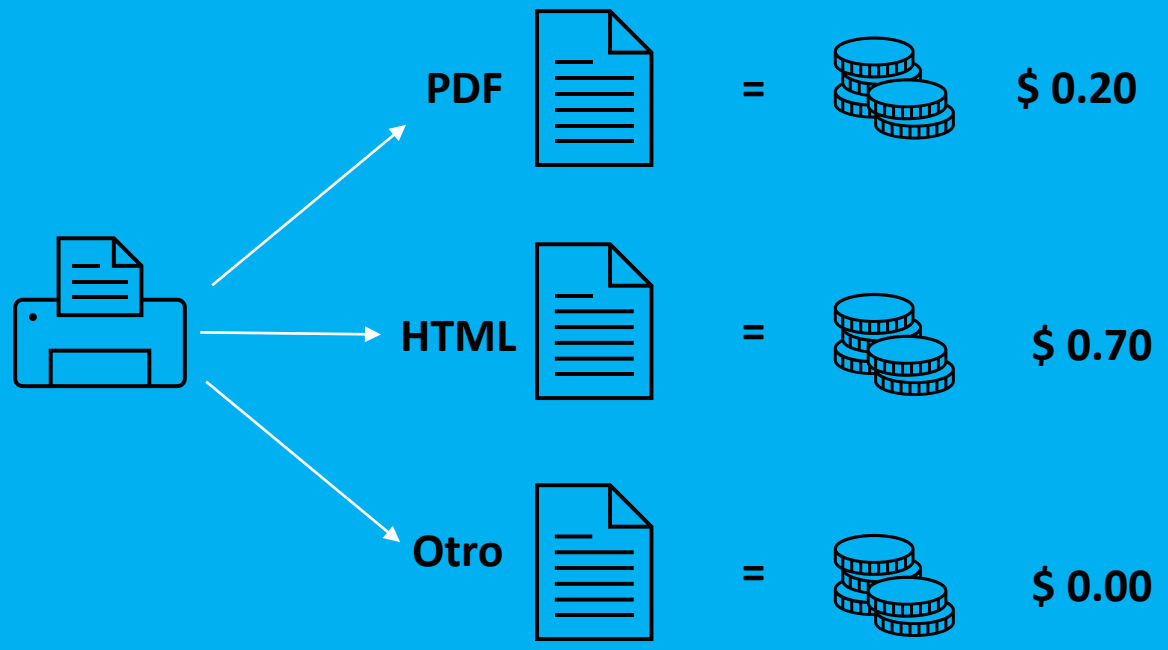
```java
public static String ExecOSCommand(OSCommand cmd){
    // Exhausting Switch with Sealing Class
    return switch (cmd){
        case LinuxCommand linux -> {
            System.out.println("Linux command exec...");
            linux.exec( cmd: "ls");
            yield "linux cmd done";
        }
        case WindowsCommand win -> {
            System.out.println("Windows command exec...");
            win.exec( cmd: "dir");
            yield "windows cmd done";
        }
        // No need for 'default'
    };
}
```

Java 21 en Acción

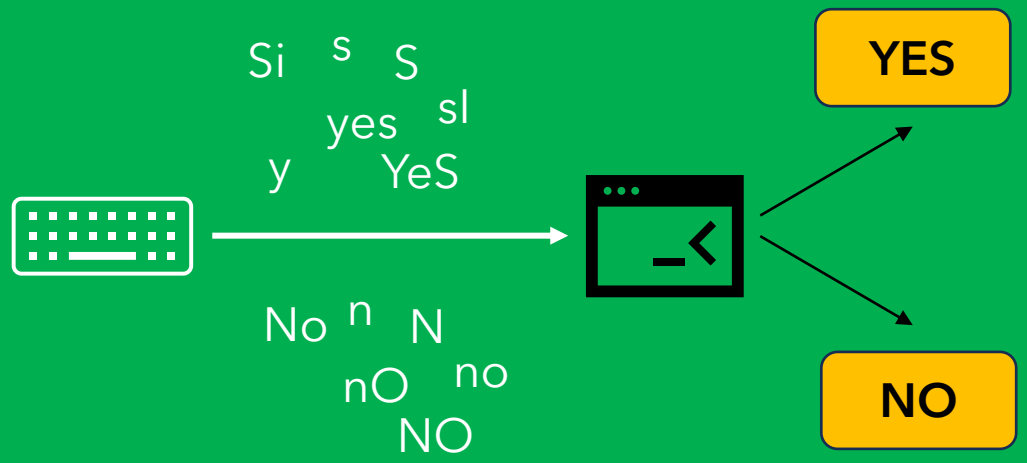
Universidad Tecnológica del Perú

PRINT SERVICE

Implementar un método que permita determinar el monto a pagar por la impresión de documentos, teniendo en cuenta que el precio varía según su tipo



Java 21 en Acción



CONSOLE REQUEST

Implementar un método que permita determinar la respuesta (Si | No) de un usuario en consola de tal forma que se le reconozca los datos de una forma flexible tanto en ingles como en español

Java 21 en Acción

Universidad Tecnológica del Perú

ENGLISH GRADE

Implementar un método que permita determinar la calificación de un estudiante de ingles según su puntuación obtenida

Nota: Usar excepciones para datos inválidos

English Exam

Score	Grade
90-100	Excellent
70-89	Very Good
50-69	Satisfying
30-49	Sufficient
0-29	Unsatisfactory

Record Patterns

- JEP 440: https://openjdk.org/jeps/440
- Mejora el uso de patrones para clases de datos (records)
- Permite una potente y declarativa forma de navegación y procesamiento de datos
- Es un excelente complemento para la sentencia switch

Imagen extraída de: https://www.mindtheproduct.com/wp-content/uploads/2022/03/shutterstock_306380360-scaled.jpg

Record Patterns

Universidad Tecnológica del Perú

Uso de records en sealed interfaces

```
public sealed interface MyAppOption {
    record ShowHelp() implements MyAppOption {}
    record SaveImage(URL url) implements MyAppOption {}
    record SaveImageInDir(URL url, Path path) implements MyAppOption {}
    record NoOption() implements MyAppOption {}
}
```

Nota: Se puede usar selaed interfaces con records para limitar claramente las opciones disponibles de una app

```
usage: MiniWGet [OPTIONS] URL
 -d,--dir <arg>   Directory for files
 -h,--help        Show this help
```

Record Patterns

Universidad Tecnológica del Perú

Uso de record pattern en sentencia switch

```
// Process Cli App Option
switch (opt){
    case MyAppOption.NoOption() -> showMyAppHelp(options);
    case MyAppOption.ShowHelp() -> showMyAppHelp(options);
    case MyAppOption.SaveImage(URL src_url) -> saveImageFromURL(src_url);
    case MyAppOption.SaveImageInDir(URL filename, Path path)
            -> saveImageFromURL(filename,path);
};
```

Nota: El uso de record patterns permite mejorar el mantenimiento de secciones de código clave

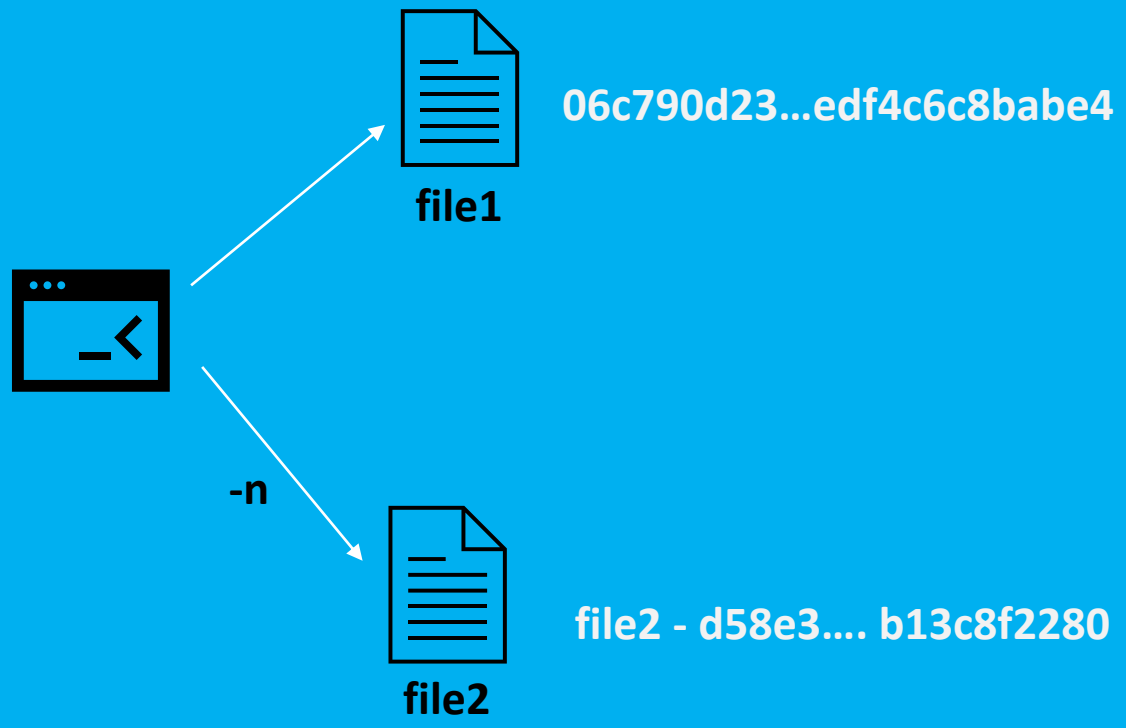
Java 21 en Acción

Universidad Tecnológica del Perú

MD5 CLI APP

Implementar un método que reciba argumentos CLI para generar el md5 de un archivo.
Se debe recibir de forma obligatoria el nombre de un archivo.

Si se recibe el argumento –n se
debe incluir el nombre del archivo como parte del resultado



Sequenced Collections

- JEP 431: https://openjdk.org/jeps/431
- Introduce nuevas interfaces para representar colecciones con un “encounter order” definido
- Uniformiza operaciones relacionadas al “encounter order” entre las clases Collection
- Permite iterar en “reversa” de manera clara y uniforme entre las clases Collection

Imagen extraída de: https://skatesradar.com/wp-content/uploads/2017/09/shutterstock_1868610.jpg

Sequenced Collections

Universidad Tecnológica del Perú

	First element	Last element
List	list.get(0)	list.get(list.size() - 1)
Deque	deque.getFirst()	deque.getLast()
SortedSet	sortedSet.first()	sortedSet.last()
LinkedHashSet	lhs.iterator().next()	// custom
SequencedCollection	seq.getFirst()	seq.getLast()

ANTES

AHORA

Sequenced Collections

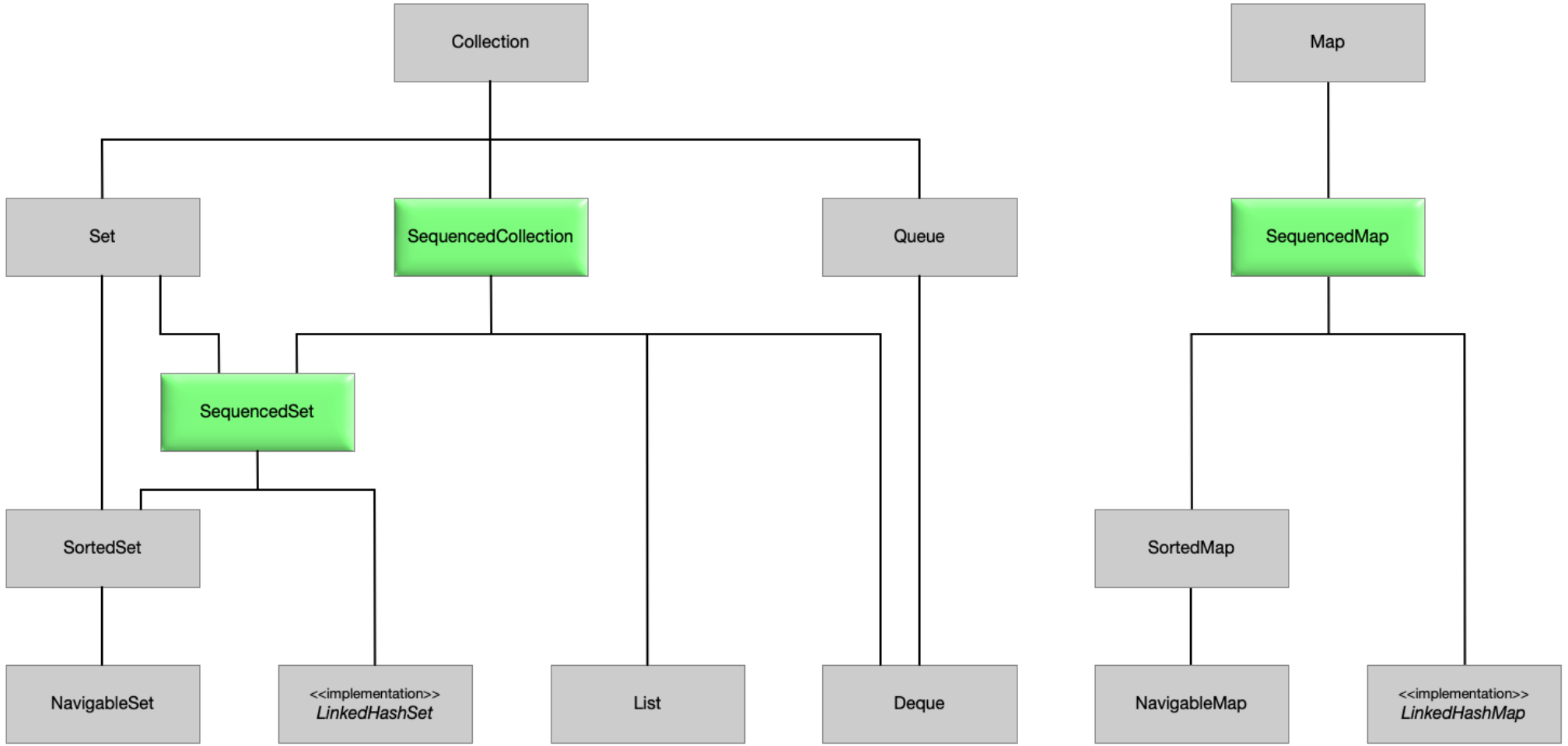


Imagen extraída de: https://cr.openjdk.org/~smarks/collections/SequencedCollectionDiagram20220216.png

Sequenced Collections

Universidad Tecnológica del Perú

```
interface SequencedCollection<E> extends Collection<E> {
    // methods promoted from Deque
    void addFirst(E);
    void addLast(E);
    E getFirst();
    E getLast();
    E removeFirst();
    E removeLast();
    // new method
    SequencedCollection<E> reversed();
}
```

Sequenced Collections


```
interface SequencedMap<K,V> extends Map<K,V> {
    // methods promoted from NavigableMap
    Entry<K, V> firstEntry();
    Entry<K, V> lastEntry();
    Entry<K, V> pollFirstEntry();
    Entry<K, V> pollLastEntry();
    // new methods
    SequencedMap<K,V> reversed();
    SequencedSet<K> sequencedKeySet();
    SequencedCollection<V> sequencedValues();
    SequencedSet<Entry<K,V>> sequencedEntrySet();
    V putFirst(K, V);
    V putLast(K, V);
}
```

Java 21 en Acción

Universidad Tecnológica del Perú

IP Address

192	168	0	100

Reverse LookUp Zone

100.0.168.192.in-addr.arpa

IPv4 UTILS

Implementar en la clase IPv4 el método:

IPv4 (constructor) para inicializar los octetos en una SeqC

getHostClassC para devolver el último octeto

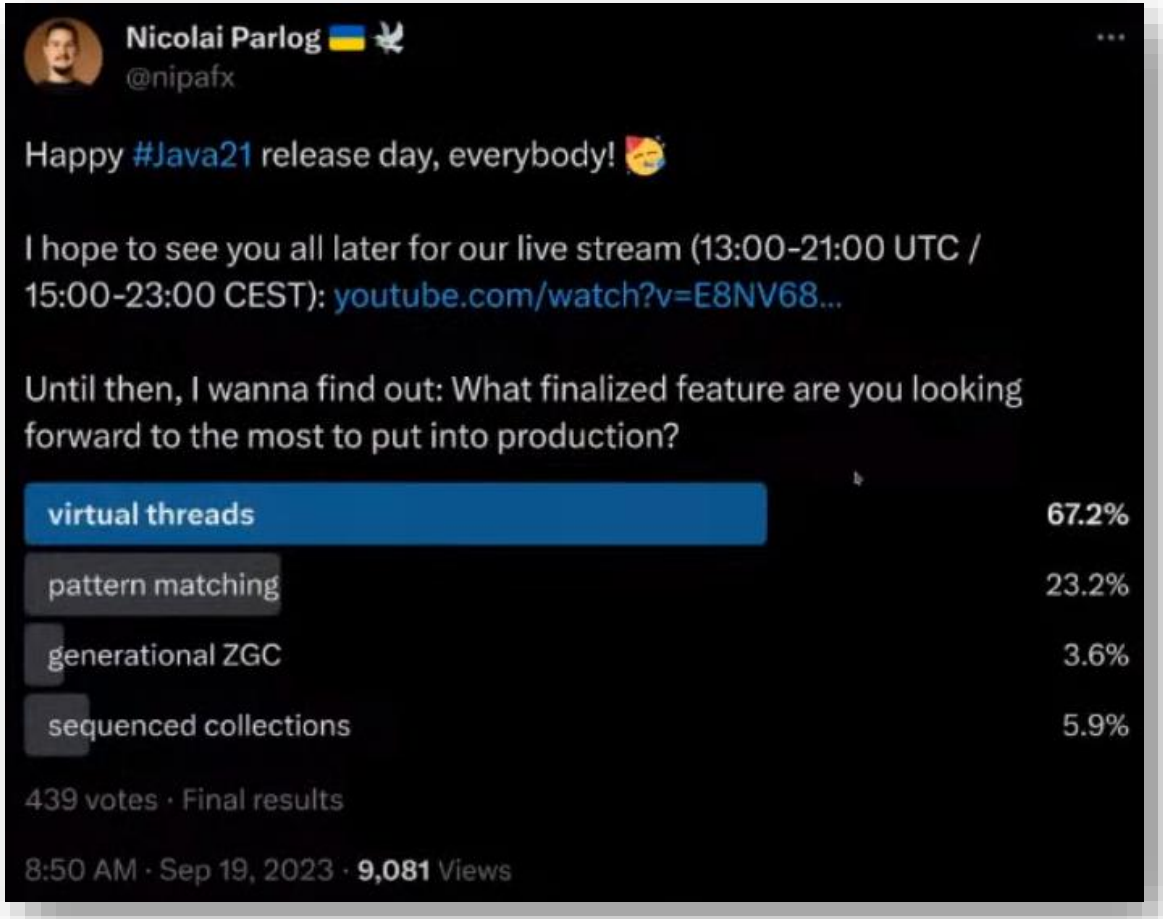
toString para devolver la IP en formato a.b.c.d

reverseLookupZone para devolver la zona DNS inversa

Virtual Threads

Universidad Tecnológica del Perú

- JEP 444: https://openjdk.org/jeps/444
- Es la característica más esperada por muchos desarrolladores
- Reduce dramáticamente el esfuerzo de escribir, mantener y observar aplicaciones concurrentes de alto rendimiento
- Optimiza la API lang.Thread (sin reemplazarla) mejorando su adopción



Link: https://www.youtube.com/watch?v=E8NV68ihJyY

Virtual Threads

Universidad Tecnológica del Perú

Nuevo estilo de programación para Threads

```
private static void run10kThreads() {
    BlockingDeque<String> lista = new LinkedBlockingDeque<>();

    // Exec 10K virtual threads
    try (var executor = Executors.newVirtualThreadPerTaskExecutor()){
        IntStream.range(0,10_000).forEach(i -> {
            var future = executor.submit(() -> {
                return "task " + String.valueOf(i);
            });
            try {
                lista.add(future.get());
            } catch (ExecutionException | InterruptedException e) {
                throw new RuntimeException(e);
            }
        }
        );
    }
    System.out.println(lista.size());
}
```

Sección donde se ejecuta el código de cada thread

Nota: En el ejemplo el programa obtiene un objeto ExecutorService que a su vez creará 10 mil virtual threads

Los resultados serán guardados en una lista *thread safe*

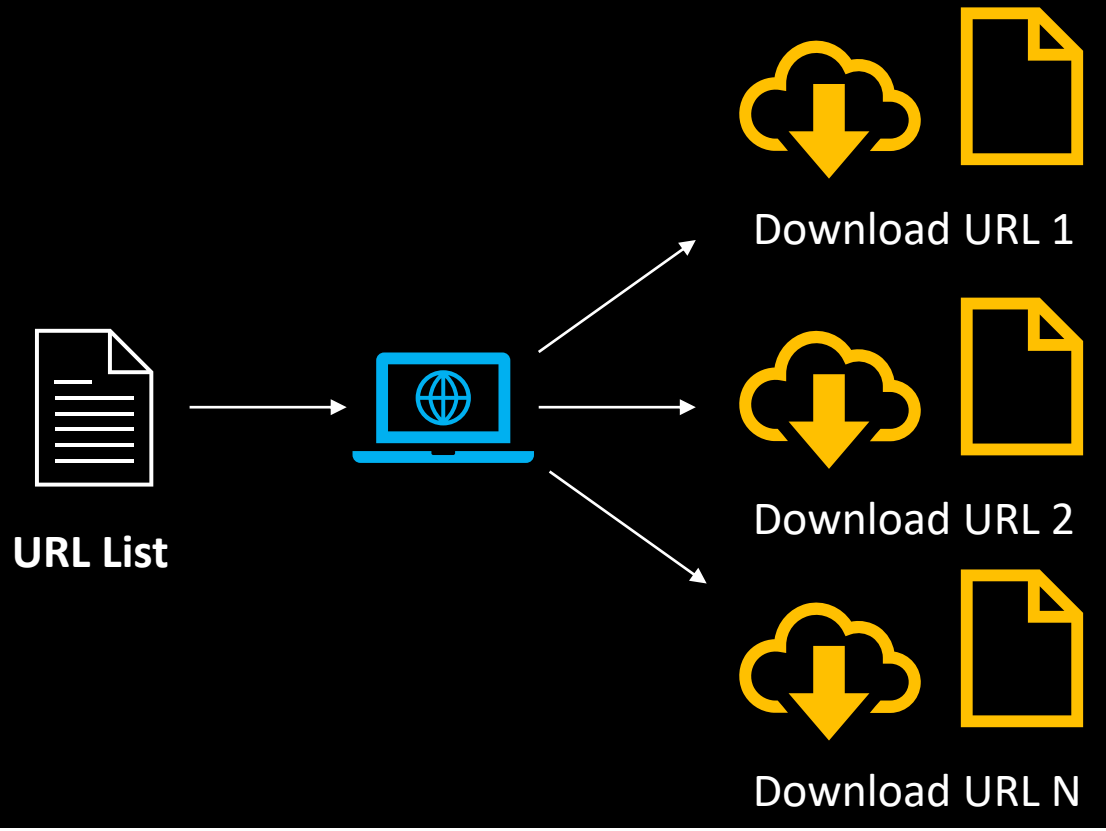
Java 21 en Acción

Universidad Tecnológica del Perú

MULTI DOWNLOAD

Implementar un programa que permita descargar el contenido HTML de múltiples páginas HTTPS proporcionados por el usuario

Nota: Guardar el contenido en una lista *thread safe*



Referencias

JDK 21: The new features in Java 21
https://www.infoworld.com/article/3689880/jdk-21-the-new-features-in-java-21.html

The Arrival of Java 21
https://inside.java/2023/09/19/the-arrival-of-java-21/

JEP 433: Pattern Matching for switch
https://openjdk.org/jeps/433

JEP 440: Record Patterns
https://openjdk.org/jeps/440

JEP 431: Sequenced Collections
https://openjdk.org/jeps/431

JEP 444: Virtual Threads
https://openjdk.org/jeps/444

JEP 439: Generational ZGC
https://openjdk.org/jeps/439

JEP 452: Key Encapsulation Mechanism API
https://openjdk.org/jeps/452

GRACIAS

fidiaz@utp.edu.pe