2) String API Changes
- Introducing new methods and improving the function-ality for better string manipulation

3) Files.mismatch (Path, Path)
- Allowing developers to efficiently find the index of the first differing byte into two files.

4) Compact Number formatting
- Providing a more concise and readable representation of large numbers

5) Support for Unicode 11
- Incorporating the latest Unicode standard for improved character handling and representation.

6) Switch Expression
- Providing a more expressive and concise way to handle multiple conditions within a switch statement

* Java 13
1) JEP 355 - Text Blocks
- Simplifying the creation of multiline strings in Java Code.

2) JEP 354 - Switch Expressions Enhancements
- Allowing it to be used as both a statement and an expression, improving code conciseness

3) JEP 353 - Reimplement the Legacy Socket API
- A reimplementation of the legacy Socket API for better maintainability and improvements in performance and security.

4) JEP 350 - Dynamic CDS Archive
- Introduces a dynamic class-data sharing (CDS) archive format, enabling more efficient sharing of class metadata.

5) JEP 351 - ZGC : Uncommit Unused Memory
- Enhancements to the ZGC to uncommit unused memory, reducing the overall memory footprint.

6) FileSystems.newFileSystem() Method
- Part of the 'java.nio.file' package, this method allows creating a new file system, providing flexibility in handling different file systems

7) DOM and SAX Factories with Namespace Support
- In the context of XML processing, introduces factories with namespace support for Document Object Model (DOM) and Simple API for XML (SAX), facilitating better handling of XML documents with namespaces in Java

* Java 14
1.) JEP 305 - Pattern Matching for instanceof
- Introduces enhanced 'instance of' with pattern matching syntax, simplifying type checks and enabling more expressive code.

2) JEP 368 - Text Blocks
- Continues refining text blocks, enhancing the readability of multiline strings in Java code.

3) JEP 358 - Helpful NullPointerExceptions
- Improves the quality of NPE error messages, providing more content to aid in debugging

4) JEP 359 - Records
- Introduces record classes as a preview feature, offering a concise way to declare classes that are primarily data carriers.

5) JEP 361 - Switch Expressions
- Promotes switch expressions from preview to standard, providing a more flexible and concise syntax for switch statements.

6) JEP 343 - Packaging Tool (Incubator)
- Introduces a new packaging tool as an incubator feature of packaging self-contained Java apps.

7) JEP 345 - NUMA - Aware Memory Allocation G1
- Enhances the G1 garbage collector with support for Non-Uniform Memory Access (NUMA), improving performance on certain hardware architectures

8) JEP 349 - JFR Event Streaming
- Enables continuous streaming of Java Flight Recorder (JFR) events, allowing real-time monitoring and analysis of applications

9.) JEP 352 - Non-Volatile Mapped Byte Buffers
- Introduces non-volatile mapped byte buffers, providing more effecint access to persistent memory.

10) JEP 363 - Remove the concurrent mark sweep (CMS) Garbage Collector
- Removes the deprecated concurrent mark sweep (CMS) garbage collector, encouraging migration to newer garbage collection mechanisms.

11.) JEP 367 - Remove the Pack200 Tools and API
- As they are no longer widely used or maintained

12) JEP 370 - Foreign - Memory Access API (Incubator)
- Allowing efficient and safe access to native memory from Java.

* Java 15
1) Sealed classes and Interfaces
- Restricting the set of subclasses and enhancing code maintainability

2) EdDSA Algorithm (JEP 339)
- Adds support for the EdDSA (Edwards - curve Digital Signature Algorithm) to the security algorithms available in the JDK

3) Hidden Classes (JEP 371)
- Providing a more secure and efficient mechanism for dynamically generated classes.

4) Pattern Matching for instanceof (JEP 375)
- Refining this feature for more expressive and concise type-checking.

5) Removed Nashorn Javascript Engine (JEP 372)
- Encouraging the use of alternative JavaScript engines

6) Reimplement the Legacy Datagram Socket API (JEP 373)
- Improves the Datagram Socket API implementation, enhancing performance and maintainability

7.) Records (JEP 384)
- Continues refining record classes as a preview feature, offering a concise way to declare - immutable data-carrying classes

8) Text Blocks become a standard feature (JEP 378)
- Elevates text blocks from a preview feature to a standard feature, providing a cleaner syntax for multiline strings in Java.

* Java 16
1) JEP 338 : Vector API (Incubator)
- Enabling developers to express vector computations that compile into optimal vector instructions.

2) JEP 347 : Enable C++ 14 Languages Features
- Allows the use of C++ 14 language features in the JDK build process, enhancing compatibility with modern C++ code.

3) JEP 357 : Migrate from Mercurial to Git
- Initiates the migration of the JDK source code repository from Mercurial to Git for improved collaboration and development workflows

4) JEP 369 : Migrate to Github
- Moves the OpenJDK Community's source code repositories to GitHub, facilitating a more accessible and Collaborative development environment.

5) JEP 376 : ZGC : Concurrent Thread- Stack Processing
- Enabling concurrent processing of thread stacks, reducing pause times.

6) JEP 380 : Unix - Domain Socket Channels
- Allowing communication between processes on the same host using unix domain sockets.

7) JEP 386 : Alpine Linux Port
- Adds support for the alpine linux distribution as a platform for running Java apps.

8.) JEP 387 : Elastic Metaspace
- Improves the metaspace memory allocation by making it elastic, allowing it to adapt more dynamically to application requirements.

9) JEP 388 : Windows/AArch64 Port.
- Introduces support for the windows/aarch64 platform Combination, expanding the range of supported architectures.

10) JEP 389 : Foreign Linker API
- Providing a Programmatic interface to native code.

11) JEP 390 : Warnings for Value-Based Classes
- Enhances the compiler to generate warnings for potentially unsafe use of value-based classes.

12) JEP 392 : Packaging Tool
- Streamlining the process of creating native packages.

13) JEP 393 : Foreign - Memory Access API
- Advances the foreign - memory access API to its third incubator stage, offering efficient and safe access to native memory.

14) JEP 394 : Pattern - Matching for Instance of
- Providing more expressive and concise type-checking.

15) JEP 395 : Records
- Finalizes record classes, offering a compact syntax for declaring immutable data - carrying classes.

16) JEP 396 : Strongly Encapsulate JDK Internals by defaults
- Strengthens encapsulation by default, restricting access to internal APIs to improve security and maintainability.

17) JEP 397 : Sealed Classes
- Advances sealed classes to their Second preview, refining this feature for enhanced code maintainability and Security.

* Java 17

1) JEP-306: Restore Always - Strict Floating-Point Semantics
- Revert changes made in Java 1.2, restoring always-strict floating-point semantics for mathematical operations.

2) JEP-356 : Enhanced Pseudo-Random Number Generators
- Introduced new interfaces and implementations for enhanced pseudo-random number generators, providing improved flexibility and performance

3) JEP-382 : New macOs Rendering Pipeline
- Introduces a new rendering pipeline for macOS, enhancing graphics performances and compatibility.

4) JEP-391 : macOS/AArch64 Port
- Extends Java support to macOS on the AArch64 architecture, expanding the range of supported platforms

5) JEP-398 : Deprecate the Applet API for Removal
- Marks the applet API as deprecated, paving the way for its eventual removal in future Java release

6) JEP-403 : Strongly Encapsulate JDK Internals
- Strengthens encapsulation by default, restricting access to internal JDK APIs for improved security and maintainability.

7) JEP-406 : Pattern Matching for Switch
- Enhances the 'switch' statement with pattern matching syntax, providing more expressive and concise code.

8) JEP-407 : Remove RMI Activation
- Removes the RMI Activation mechanism, simplifying the RMI (Remote Method Invocation) systems.

9) JEP-409 : Sealed Classes
- Finalizes sealed classes, provingding a more controlled and secure way to define class hierarchies

10) JEP-410 : Remove the Experimental AOT and JIT Complier
- Removes the experimental Ahead-of-Time (AOT) and Just-In-Time (JIT) compiler known as Graal.

11) JEP-411 : Deprecate the Security Manager for Removal
- Marks the security manager as deprecated, signaling its eventual removal in future Java versions

12) JEP-412 : Foreign Function & Memory API
- Enabling interaction with native code more easily

13) JEP-414 : Vector API
- Advances the vector API to its second incubator stage, providing a mechanism for expressing vector computations

14) JEP-415 : Content - Specific Deserialization filters
- Enhances deserialization filtering with the ability to define filters based on the context. Improving security in handling object deserialization

* Java 18

1) JEP-400 : UTF-8 by Default
- Switches the default character encoding for Java Source files to UTF-8, enhancing internationaliz-ation support.

2) JEP-408 : Simple Web Server
- Introduces a lightweight, simple HTTP web server as part of the JDK, facilating easy development and testing.

3) JEP-413 : Code Snippets in Java API Documentation
- Enhances Java API documentation by allowing code snippets within the documentation for better illustration and understanding.

4) JEP-416 : Reimplement Core Reflection with Method Handles
- Improves core reflection by reimplementing it using method handles, providing better performance and maintainability.

5) JEP-417 : Vector API
- Continues the incubation of the Vector API, offering a mechanism for expressing vector Computations.

6) JEP-418 : Internet-Address Resolution SPI
- Introduces a Service Provider Interface (SPI) for internet address resolution, providing a more extensible approach

7) JEP-419 : Foreign Function & Memory API
- Allowing more seamless integration with native code.

8) JEP-420 : Pattern Matching for Switch
- Refining this feature for more expressive code.

9) JEP-421 : Deprecate finalization for Removal
- Marks the Object.finalize() method as deprecated, signaling its eventual removal, and encourages the use of alternative cleanup mechanisms

* Java 19

1) JEP-405 : Record Patterns
- Improves the way of deconstructing or extracting the record values

2) JEP 422 : Linux/RISC-V Port
- The port of the JDK will integrate into the JDK mainline repository.

3) JEP 424 : Foreign Function & Memory API
- It resides in the java.lang.foreign package of the java.base module.

4) JEP 425 : Virtual Threads
- A lightweight implementation of threads provided by the JDK instead of the OS

5) JEP 426 : Vector API
- JEP improves the vector API performance and other enhancements in response to feedback.

6) JEP 427 : Pattern Matching for Switch
- Guarded patterns are replaced with when clauses in switch blocks.

7) JEP 428 : Structured Concurrency
- To simplify multithreaded programming
- Unstructured Concurrency API

* Java 20 :
1) Scoped Values - (JEP - 429)
- They allow storing a value for a limited time in such a way that only the thread that wrote the value can read it.

2) Record Patterns - (JEP - 432)
- A record pattern can be used with instance of or switch to access the fields of a record without casting and calling accessor methods

3) Inference of Type Arguments of Generic Record Pattern
- The compiler can infer the type so that we can omit it from the instance of checks.

) Record Patterns in for loop
- We can specify a record pattern in the for loop and then access x and y directly (just like with instance of and switch).

5) Removal of Support for Named Record Patterns
- In named record pattern, there are two ways to access the fields of the record - either via x & y variables
- This variant was decided to be superflous and removed

6) Pattern Matching for switch - (JEP - 433)
- We can also combine the switch statement with record patterns to access the record fields directly

7) Matching Exception for Exhausting Switch
- An exhaustive switch (i.e., a switch that includes all possible values) throws a Match Exception (rather than an Incompatible Class Change Error) if it is determined at runtime that no switch label matches

8) Deprecations and Deletions
- Some methods were marked as "deprecated" or completely disabled.
- java.net.URL constructors are deprecated

9.) Structured Concurrency - (JEP 437)
- When In the second incubator phase, Structured Task Scope is extended to automatically inherit "Scoped values" to all child threads

10.) Virtual Threads - (JEP 436)
- New Methods were introduced in Thread - join (Duration), sleep(Duration), & threadId()
- New Methods in Future - resultNow(), exceptionNow() and state().
- ExecutorService extends the Autocloseable interface.

* Java 21

1) String Templates - (JEP-430)
   - String templates for concise string formatting

2) Sequenced Collections - (JEP-431)
   - Enhances collections with efficient, ordered element traversal.

3) Generational ZGC - (JEP-439)
   - Introduces Generational Z Garbage Collector for improved performance.

4) Record Patterns - (JEP-440)
   - Enables pattern matching for records, enhancing code expressiveness

5) Pattern Matching for switch - (JEP-441)
   - Extends pattern matching to switch statements for more robust code.

6) Foreign Function & Memory API - (JEP-442)
   - Advances the foreign function and Memory API for improved native integration

7) Unnamed Patterns and Variables - (JEP-443)
   - Introduces unnamed patterns and variables for more flexible pattern matching.

8) Virtual Threads - (JEP-444)
   - Introduces lightweight virtual threads for efficient Concurrent programming.

9) Unnamed Classes and Instance Main Methods (JEP-445)
   - Introduces unnamed classes and instance main methods for enhanced code organization

10) Scoped Values - (JEP-446)
   - It is for better resource management

11) Vector API - (JEP-448)
   - Advances the vector API for high-performance parallel computing.

12) Deprecate the Windows 32-bit x86 Port for Removal - (JEP-449)
   - Marks the deprecation of the windows 32-bit x86 port.

13) Prepare to Disallow the dynamic loading of agents - (JEP-451)
   - It is for security improvements

14) Key Encapsulation Mechanism API - (JEP-452)
   - It is for cryptographic operations

15) Structured Concurrency - (JEP-453)
   - It is for improved control over concurrent t