eazy bytes

JAVA 7 & 8 NEW FEATURES

ZERO TO MASTER

JAVA 7 NEW FEATURES

AGENDA

eazy bytes

1. TRY-WITH-RESOURCES STATEMENT
2. SUPPRESSED EXCEPTIONS
3. CATCHING MULTIPLE EXCEPTIONS
4. RETHROWING EXCEPTIONS WITH TYPE CHECKING
5. EASIER EXCEPTION HANDLING FOR REFLECTIONS
6. OBJECTS CLASS & NULL CHECKS
7. CLOSE METHOD INSIDE URLCLASSLOADER
8. ENHANCEMENTS TO FILES & DIRECTORIES

JAVA 7 NEW FEATURES

AGENDA

eazy bytes

9. WATCHSERVICE

10. STRING IN SWITCH STATEMENT

11. BINARY LITERALS

12. TYPE INFERENCE/DIAMOND OPERATOR

13. USING UNDERSCORE IN NUMERIC LITERALS

14. JDBC IMPROVEMENTS

JAVA 8 NEW FEATURES

AGENDA

eazy bytes

1. DEFAULT METHODS IN INTERFACES
2. STATIC METHODS IN INTERFACES
3. OPTIONAL TO DEAL WITH NULLS
4. LAMBDA (Λ) EXPRESSION
5. FUNCTIONAL INTERFACE
6. METHOD REFERENCES
7. CONSTRUCTOR REFERENCES
8. STREAMS API

JAVA 8 NEW FEATURES

AGENDA

eazy bytes

9. NEW DATE AND TIME API(JODA)

10. COMPLETABLEFUTURE

11. MAP ENHANCEMENTS

12. OTHER MISCELLANEOUS UPDATES

JAVA 7 & 8 NEW FEATURES

PREREQUISITES

eazy bytes

KNOWLEDGE IN ATLEAST JAVA 1.6

Previous knowledge or working experience in at least Java 1.6. You should be comfortable with Java basics & syntaxes

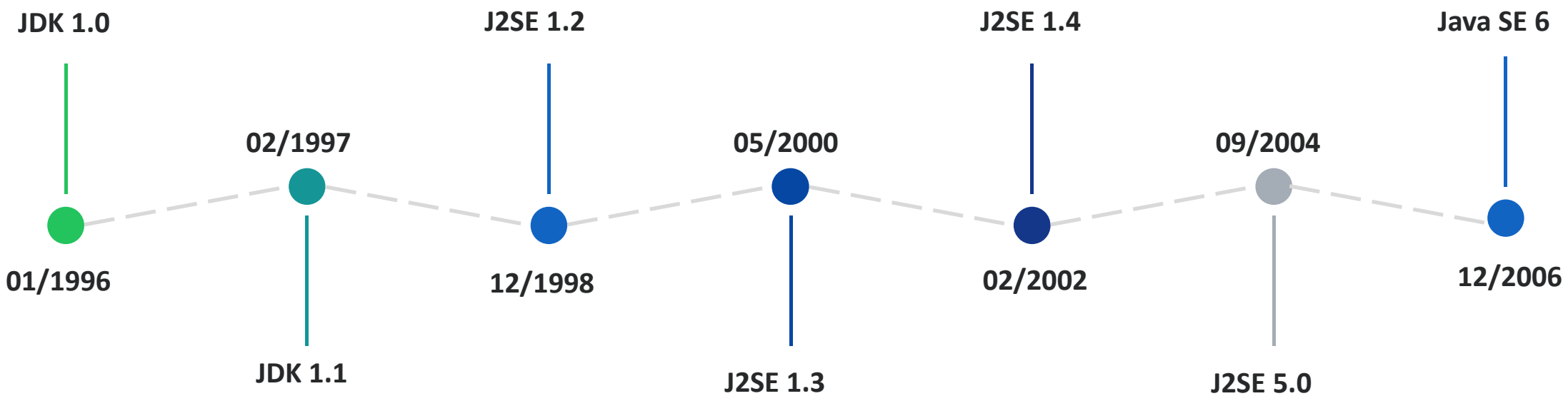
INTEREST TO LEARN & UPSKILL

Interest to learn and explore latest features of Java

JAVA VERSION HISTORY

MILESTONE DATES

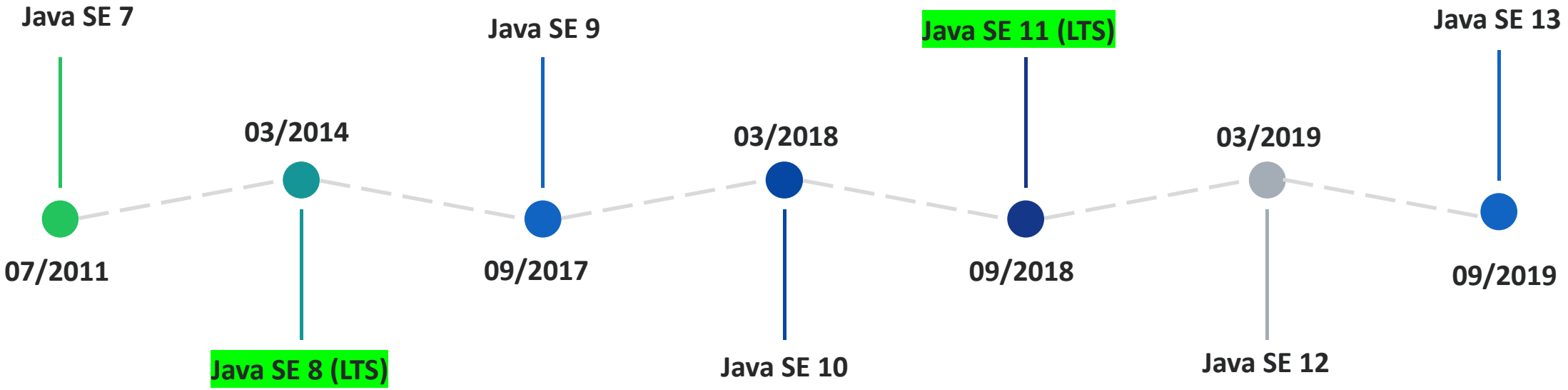
eazy bytes



JAVA VERSION HISTORY

MILESTONE DATES

eazy bytes

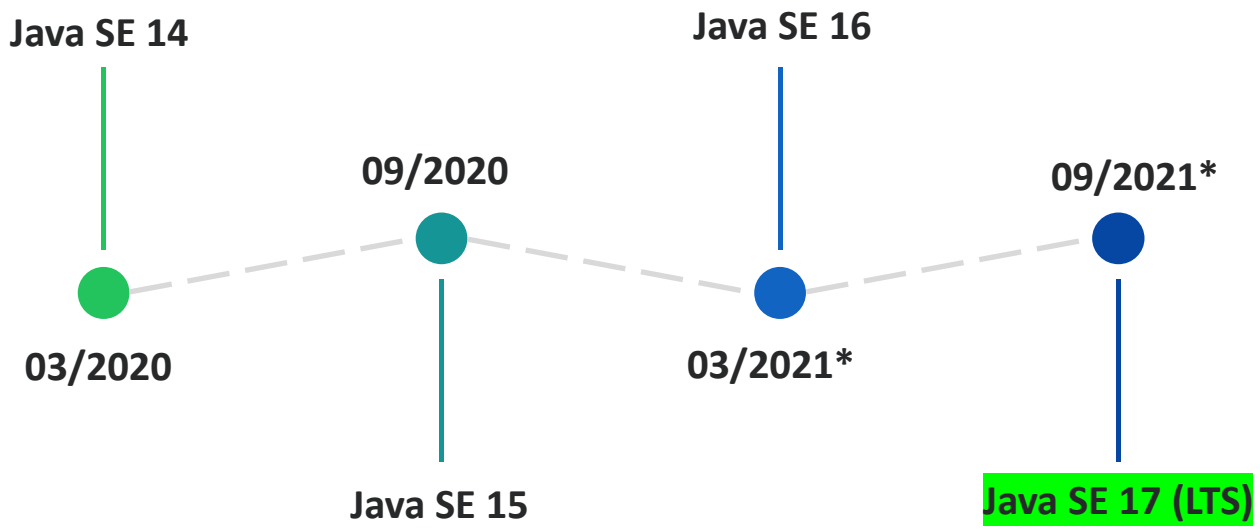


LTS – Long Term Support Release

JAVA VERSION HISTORY

MILESTONE DATES

eazy bytes



- ✓ *LTS – Long Term Support Release*
- ✓ ** - Proposed Dates*

JAVA VERSION HISTORY

MILESTONE DATES

eazy bytes

- ✓ Previously on an average Java use to release an version every 2 years. In some cases it even 5 years. But Java team realized that this approach will no longer sustainable because it will not evolve Java language fast enough to compete with other languages.

- ✓ So from Java 9 version, a new version is being released every 6 months.

- ✓ But since it could be a problem for some conservative organizations to update their Java every 6 months, so Java team decided every 3 years there will be a long-term support (LTS) release that will be supported for the subsequent 3-5 years. But minor versions will continue to release every 6 months to roll out new changes/features as per the demand.

- ✓ But when ever a non-LTS or LTS version release, the previous non-LTS versions will be archived and no support will be provided for them.

- ✓ For example, with Java 10 release since Java 9 is a non-LTS version it will be archived and support will be stopped. But all the new features from Java 9 will be present in Java 10 as well.

DIFFERENT JDK VENDORS IN JAVA

ORACLE JDK, OPEN JDK, ADOPTOPENJDK

eazy bytes

Open JDK

In terms of Java source code, there is only one, living at the OpenJDK project site. These builds are free and unbranded, but Oracle won't release updates for older versions, say Java 15, as soon as Java 16 comes out.

https://openjdk.java.net/.

Oracle JDK

OracleJDK, which is a branded, commercial build starting with the license change in 2019. Which means it can be used for free during development, but you need to pay Oracle if using it in production.

https://www.oracle.com/java/technologies/javase-downloads.html

AdoptOpenJDK

In 2017, a group of vendors (Amazon, Microsoft, Pivotal, Redhat and others) started a community, called AdoptOpenJDK.

They provide free, rock-solid OpenJDK builds with longer availibility/updates

https://adoptopenjdk.net/

JAVA 7 NEW FEATURES

THE TRY-WITH-RESOURCES STATEMENT

eazy bytes

- To make Developer life easy and improve the quality of the code, a new feature called *'TRY-WITH-RESOURCES statements'* are introduced.

- Using this we don't have to explicitly close the resource. We just have to initialize the resources details inside the () immediately next to try keyword. In its simplest form, the try-with-resources statement can be written as,

```
try (BufferedReader br = ...) {
      //work with br
}
```

- When try block execution completes the br.close() method will be automatically called by JVM. For this a new interface is created in Java 7 which is *'java.lang.AutoCloseable'.*

JAVA 7 NEW FEATURES

THE TRY-WITH-RESOURCES STATEMENT

eazy bytes

- Before *'java.lang.AutoCloseable'* introduced in Java 7, we have an interface *'java.io.Closeable'* which restricts the type of exception thrown to only IOException.
- But the new Interface(AutoCloseable) can be used in more general contexts, where the exception thrown during closing is not necessarily an IOException.
- Since Closeable meets the requirements for AutoCloseable, it is implementing AutoCloseable when the latter was introduced.
- A try-with-resources statement can itself have catch and finally clauses for other requirements inside the application.
- You can specify multiple resources as well inside the try-with-resources statement.

JAVA 7 NEW FEATURES

THE TRY-WITH-RESOURCES STATEMENT

eazy bytes

- Sample implementation of try-with-resources statement,

```
public static void withJava7() throws FileNotFoundException, IOException {
    try(BufferedReader br = new BufferedReader(new FileReader("C:/eazybytes.txt"));) {
        String sCurrentLine;
        while ((sCurrentLine = br.readLine()) != null) {
            System.out.println(sCurrentLine);
        }
    }
}
```

- All resource reference variables should be final or effective final. So we can't perform reassignment with in the try block.

- Till Java 1.6, try block should be followed by either catch or finally block but from Java 7 we can have only try with resource block with out catch & finally blocks.

JAVA 7 NEW FEATURES

SUPPRESSED EXCEPTIONS

eazy bytes

- Suppressed exceptions are the exceptions thrown in the code but were ignored somehow. One of the classic example is in the scenario's 'try-catch-finally' block execution, where we received an exception in try block and again there is one more exception thrown due to which the super exception from try block will be ignored.

- To support suppressed exceptions better handling, a new constructor and two new methods were added to the Throwable class (parent of Exception and Error classes) in JDK 7.

  - ✓ *Throwable.addSupressed(aThrowable);*
  - ✓ *Throwable.getSupressed(); // Returns Throwable[]*

JAVA 7 NEW FEATURES

CATCHING MULTIPLE EXCEPTIONS

eazy bytes

- Before Java 7, suppose if you have the same action need to be performed in the case of different RuntimeException exceptions like NullPointerException & ArrayIndexOutOfBoundsException we will forced to write multiple catch blocks like below,

```
public static void beforeJava7() {
    int arr[] = { 1, 2, 3 };
    try {
        for (int i = 0; i < arr.length + 1; i++) {
            System.out.println(arr[i]);
        }
    } catch (NullPointerException nex) {
        LOGGER.log(Level.SEVERE, nex.toString());
    } catch (ArrayIndexOutOfBoundsException aioex) {
        LOGGER.log(Level.SEVERE, aioex.toString());
    }
}
```

JAVA 7 NEW FEATURES

CATCHING MULTIPLE EXCEPTIONS

eazy bytes

- To reduce the code duplication in the scenarios where we have a common action/business logic need to be performed for different run time exceptions we can use a single catch block with multiple exceptions separated by a | operator as shown below,

```
public static void withJava7() {
    int arr[] = { 1, 2, 3 };
    try {
        for (int i = 0; i < arr.length + 1; i++) {
            System.out.println(arr[i]);
        }
    } catch (NullPointerException | ArrayIndexOutOfBoundsException ex) {
        LOGGER.log(Level.SEVERE, ex.toString());
    }
}
```

- When you catch multiple exceptions, the exception variable is implicitly final. So, you cannot assign a new value to ex in the body of the catch clause

JAVA 7 NEW FEATURES

RETHROWING EXCEPTIONS WITH MORE INCLUSIVE TYPE CHECKING

eazy bytes

- The Java SE 7 compiler performs more precise analysis of rethrown exceptions than earlier releases of Java SE. This enables you to specify more specific exception types in the throws clause of a method declaration.

- Consider the below example,

```
static class FirstException extends Exception { }

static class SecondException extends Exception { }

/**
 * Sample implementation before Java 7
 *
 * @throws Exception
 *
 */
public static void beforeJava7(String exceptionName) throws Exception {
    try {
        if (exceptionName.equals("First")) {
            throw new FirstException();
        } else {
            throw new SecondException();
        }
    } catch (Exception e) {
        throw e;
    }
}
```

JAVA 7 NEW FEATURES

RETHROWING EXCEPTIONS WITH MORE INCLUSIVE TYPE CHECKING

eazy bytes

- This examples' try block could throw either FirstException or SecondException. Suppose you want to specify these exception types in the throws clause of the beforeJava7 method declaration. In releases prior to Java SE 7, you cannot do so. Because the exception parameter of the catch clause, e, is type Exception, and the catch block rethrows the exception parameter e, you can only specify the exception type Exception in the throws clause of the beforeJava7 method declaration.

- However, in Java SE 7, you can specify the exception types FirstException and SecondException in the throws clause in the rethrowException method declaration. The Java SE 7 compiler can determine that the exception thrown by the statement throw e must have come from the try block, and the only exceptions thrown by the try block can be FirstException and SecondException. Even though the exception parameter of the catch clause, e, is type Exception, the compiler can determine that it is an instance of either FirstException or SecondException.

JAVA 7 NEW FEATURES

RETHROWING EXCEPTIONS WITH MORE INCLUSIVE TYPE CHECKING

eazy bytes

- Code snipper with Java 7 new feature,

```
static class FirstException extends Exception { }

static class SecondException extends Exception { }

/**
 * Sample implementation from Java 7
 *
 * @throws FirstException,SecondException
 *
 */
public static void withJava7(String exceptionName) throws FirstException, SecondException {
    try {
        if (exceptionName.equals("First")) {
            throw new FirstException();
        } else {
            throw new SecondException();
        }
    } catch (Exception e) {
        throw e;
    }
}
```

JAVA 7 NEW FEATURES

EASIER EXCEPTION HANDLING FOR REFLECTIVE METHODS

eazy bytes

- Before Java 7 when we try to invoke methods inside a class using reflections, the developer will be forced to handle multiple exceptions related to reflections as shown below,

```
public static void beforeJava7() {
    try {
        Class.forName("com.eazybytes.java7.CatchingMultipleExceptions").getMethod("withJava7").invoke(null,
                new Object[] {});
    } catch (IllegalAccessException | InvocationTargetException | NoSuchMethodException
            | ClassNotFoundException nex) {
        LOGGER.log(Level.SEVERE, nex.toString());
    }
}
```

- This is a very annoying and tedious job for the developers to handle all these exceptions. Ideally all these exceptions should have been grouped together.

JAVA 7 NEW FEATURES

EASIER EXCEPTION HANDLING FOR REFLECTIVE METHODS

eazy bytes

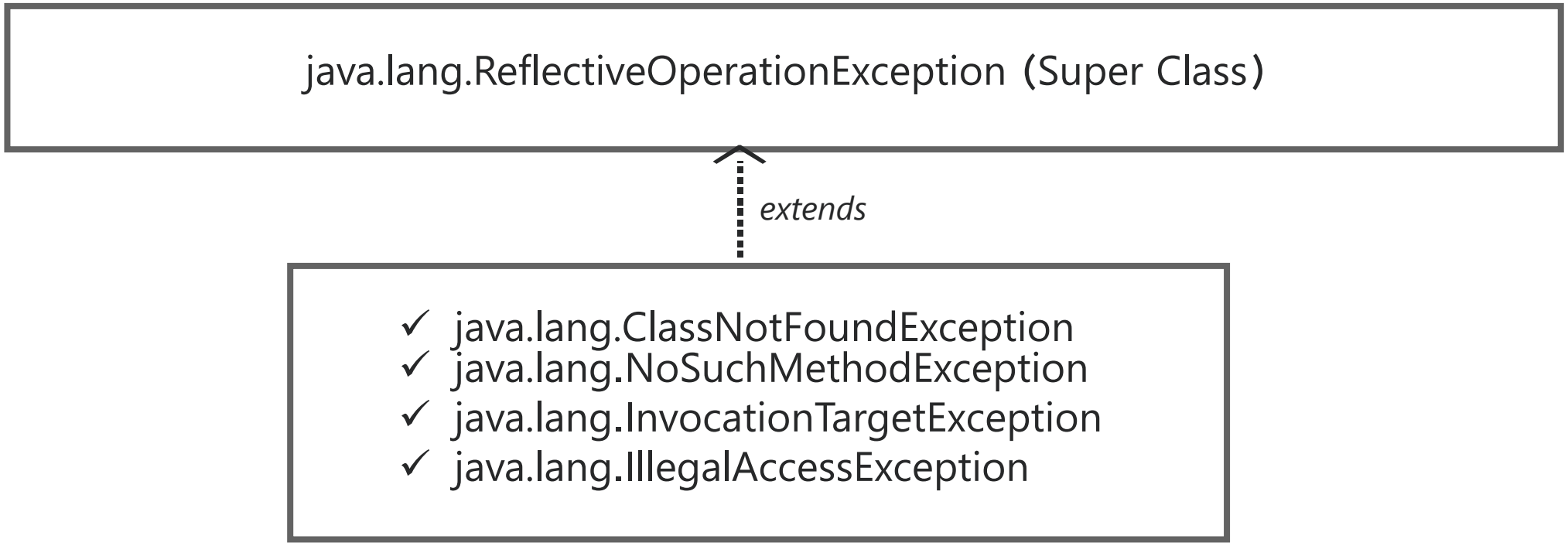
- From Java 7, all the exceptions related to reflections are grouped by creating a common super class called java.lang.ReflectiveOperationException

```
public static void withJava7() {
    try {
        Class.forName("com.eazybytes.java7.CatchingMultipleExceptions").getMethod("withJava7").invoke(null,
                new Object[] {});
    } catch (ReflectiveOperationException nex) {
        LOGGER.log(Level.SEVERE, nex.toString());
    }
}
```

JAVA 7 NEW FEATURES

EASIER EXCEPTION HANDLING FOR REFLECTIVE METHODS

eazy bytes



JAVA 7 NEW FEATURES

OBJECTS CLASS & NULL CHECKS

eazy bytes

- Java 7 introduced a new class "java.util.Objects" consists of utility static methods for operating on objects. This methods include null checks, computing hash code, comparing two objects, returning a string for an object.

- Importantly the Objects class has the below 2 methods to null-check the object. Both of them will throw NullPointerException if the given object is null and the custom message also can be passed to display on the NullPointerException details.

  - ✓ requireNonNull(T obj)
  - ✓ requireNonNull(T obj, String message)

- It may look for the developers why I should use these methods as if the object is null anyways the NullPointerException will be thrown by the code. But these methods provides controlled behavior, easier debugging.

JAVA 7 NEW FEATURES

CLOSE METHOD INSIDE URLCLASSLOADER

eazy bytes

- java.net.URLClassLoader is used to load classes and resources from a search path of URLs referring to both JAR files and directories.

```
URL[] urls = {new URL("file:junit-4.11.jar"),
              new URL("file:commons-io-2.8.0.jar")};
URLClassLoader loader = new URLClassLoader(urls);
Class<?> junitClass = loader.loadClass("org.junit.runner.JUnitCore");
```

- Before Java 7, code such as this could lead to resource leaks. Java 7 simply adds a close method to close the URLClassLoader by implementing AutoCloseable. So from Java 7 you can use a try-with-resources statement as shown below,

```
try (URLClassLoader loader = new URLClassLoader(urls)) {
        Class<?> junitClass = loader.loadClass("org.junit.runner.JUnitCore");
}
```

JAVA 7 NEW FEATURES

@SAFEVARARGS ANNOTATION

eazy bytes

- Java 5 introduced the concept of varargs, or a method parameter of variable length, as well as parameterized types. @SafeVarargs annotation is introduced in JDK 7 which is used to suppress the unsafe operation warnings at the compile time.

- @SafeVarargs annotation is used to indicate that methods will not cause heap pollution. These methods are considered to be safe.

- @SafeVarargs can only be applied on

    - ✓ Final methods
    - ✓ Static methods
    - ✓ Constructors

- From Java 9, it can also be used with private instance methods.

JAVA 7 NEW FEATURES

ENHANCEMENTS RELATED TO FILES & DIRECTORIES

eazy bytes

- Java 7 brought series of enhancements which will help developers while doing operations related to files like reading, copying & deleting.

- Java team refreshed the NIO(new I/O) package in Java 7 which originally introduced in Java 1.4

- Below are the newly created interfaces and classes created in Java 7,

  - ✓ *java.nio.file.Path*
  - ✓ *java.nio.file.Paths*
  - ✓ *java.nio.file.Files*

- These new interfaces and classes from 'java.nio.file.*' package can be used to overcome the limitations of the "java.io.File" class.

JAVA 7 NEW FEATURES

ENHANCEMENTS RELATED TO FILES & DIRECTORIES

eazy bytes

java.nio.file.Path & java.nio.file.Paths

- Java.nio.file.Path interface is introduced in Java 7 to replace the java.io.File class
- A Path is a sequence of directory names, optionally followed by a file name and it is the main entry point to all operations involving file system paths. It allows us to create and manipulate paths to files and directories.
- The utility class, java.nio.file.Paths (in plural form) is the formal way of creating Path objects. It has two static methods for creating a Path from a path string & URI.
- The static Paths.get method receives one or more strings, which it joins with the path separator of the default file system (/ for a Unix-like file system, \ for Windows).

JAVA 7 NEW FEATURES

ENHANCEMENTS RELATED TO FILES & DIRECTORIES

eazy bytes

Important methods inside Path & Paths

Path.getFileSystem()
Path.isAbsolute()
Path.getRoot()
Path.getFileName()
Path.getParent()
Path.startsWith()
Path.endsWith()
Path.normalize()
Path.resolve()
Path.relativize()
Path.toUri()
Path.toAbsolutePath()
Path.toFile()
Paths.get()

Important methods inside Files

Files.readAllBytes()
Files.readAllLines()
Files.write()
Files.newInputStream()
Files.newOutputStream()
Files.newBufferedReader()
Files.newBufferedWriter()
Files.copy()
Files.createDirectory()
Files.createFile()
Files.createTempFile()
Files.createTempDirectory()
Files.move()
Files.delete()
Files.deleteIfExists()

JAVA 7 NEW FEATURES

WATCHSERVICE

eazy bytes

- Java 7 introduces a new interface called 'java.nio.file.WatchService' that watches registered objects for changes and events. For example a file manager may use a watch service to monitor a directory for changes so that it can update its display of the list of files when files are created or deleted.

- It is very common for many application to track on files such as configuration so that we can refresh the value inside the memory on every change. We used to handle using a Thread that periodically poll for file change before Java 7. But this made easy using 'WatchService'.

- A Watchable object is registered with a watch service by invoking its register method, returning a WatchKey to represent the registration. When an event for an object is detected the key is signaled, and if not currently signaled, it is queued to the watch service so that it can be retrieved by consumers that invoke the poll or take methods to retrieve keys and process events. Once the events have been processed the consumer invokes the key's reset method to reset the key which allows the key to be signaled and re-queued with further events.

JAVA 7 NEW FEATURES

WATCHSERVICE

eazy bytes

Important methods & classes related to WatchService

WatchService.poll()
WatchService.take()
WatchService.close()

Watchable.register()

WatchKey.isValid()
WatchKey.pollEvents()
WatchKey.reset()
WatchKey.cancel()
WatchKey.watchable()

StandardWatchEventKinds. ENTRY_CREATE/
ENTRY_DELETE/ENTRY_MODIFY/OVERFLOW

JAVA 7 NEW FEATURES

BINARY LITERALS

eazy bytes

- A binary literal is a number that is represented in 0s and 1s (binary digits). Java allows you to express integral types (byte, short, int, and long) in a binary number system especially when we are dealing with the bit-wise operators.

- To handle the scenarios where we need to express an long/integer/short/byte in binary number system, Java 7 added a new feature called 'Binary Literal'. We just need to add a prefix of either 0b/0B Infront of the binary digit representation, then java will take care of converting into a decimal representation value of it.

```
int num = 0B111;
System.out.println("The number value is = "+ num);  // This will print 7 on the console
```

JAVA 7 NEW FEATURES

STRING IN SWITCH STATEMENT

eazy bytes

- Before Java 7, switch statements allows only Enum and int types.
- From Java 7, Java allows to use String objects in the expression of switch statement.
- String value is case sensitive and null objects are not allowed. In case if you use null value, java will throw an NullPointerException.
- It must be only String object and normal Object is not allowed

```
switch (input)
{
    case ("Monday"):
        System.out.println("Today is Monday");
         break;
    default:
        System.out.println("Today is not a Monday");
}
```

JAVA 7 NEW FEATURES

TYPE INFERENCE/DIAMOND OPERATOR

eazy bytes

- Before Java 7, we have to create an object with Generic type on both side of the expression like below:

```
Map<String, Integer> inputMap = new HashMap<String, Integer>();
```

- From Java 7, Java provides a improved compiler which is smart enough to infer the type of generic instance. It simplifies the use of generics when creating an object.

- It is also called as Diamond operator. Using it we can create the object without mentioning generic type on right side of expression like below:

```
Map<String, Integer> inputMap = new HashMap<>();
```

- Please note that we can't use this diamond operator feature inside the anonymous inner class. But this limitation is resolved in Java 9.

JAVA 7 NEW FEATURES

USING UNDERSCORE IN NUMERIC LITERALS

eazy bytes

- From Java 7, we can use the underscore(_) inside the numeric values to improve readability of the code. However the compiler will remove them internally while processing the numeric values.

- For example if our code contains numbers with many digits, we can use an underscore character to separate digits in groups of three, similar to how we would use a punctuation mark like a comma in mathematics.

```
int num = 1_00_00_000;
```

- But please note that below restrictions while using the underscore inside your numeric values,

  - ✓ It is not allowed at the beginning or end of a number
  - ✓ It is not allowed adjacent to a decimal point
  - ✓ It is not allowed prior to an L or F suffix that we use to indicate long/float numbers
  - ✓ It is not allowed where a string of digits is expected.

```
~~int num = _6, 6_, 6._0, 6_.0,6_54_00_L, 6_54_00_F, 0B_0101~~
```

JAVA 7 NEW FEATURES

JDBC IMPROVEMENTS

eazy bytes

- JDBC 4.1, which is part of Java 7, introduces the following features:

  - ✓ *The ability to use a try-with-resources statement to automatically close resources of type Connection, ResultSet, and Statement*
  - ✓ *RowSet 1.1: The introduction of the RowSetFactory interface and the RowSetProvider class, which enable you to create all types of row sets supported by your JDBC driver.*

- You can use a try-with-resources statement to automatically close java.sql.Connection, java.sql.Statement, and java.sql.ResultSet objects, regardless of whether a SQLException or any other exception has been thrown. A try-with-resources statement consists of a try statement and one or more declared resources (which are separated by semicolons).

- With the help of new RowSetFactory interface and the RowSetProvider class, we can create a RowSet objects of CachedRowSet/ FilteredRowSet / JdbcRowSet / JoinRowSet / WebRowSet. Before Java 7 we used to depend on JdbcRowSetImpl class for the same.

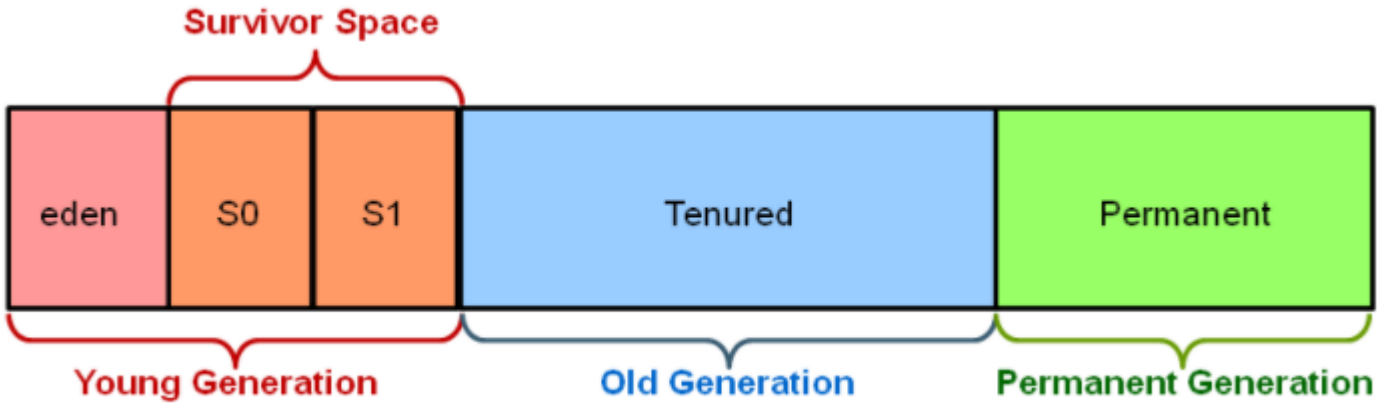
JAVA 7 NEW FEATURES

GARBAGE-FIRST COLLECTOR

eazy bytes

- JDK 7 introduced a new Garbage Collector known as G1 Garbage Collection, which is short form of garbage first. G1 is planned as the long term replacement for the existing Concurrent Mark-Sweep Collector (CMS). Comparing G1 with CMS, there are differences that make G1 a better solution.

- The older garbage collectors (serial, parallel, CMS) all structure the heap into three sections: young generation, old generation, and permanent generation of a fixed memory size. All memory objects end up in one of these three sections.

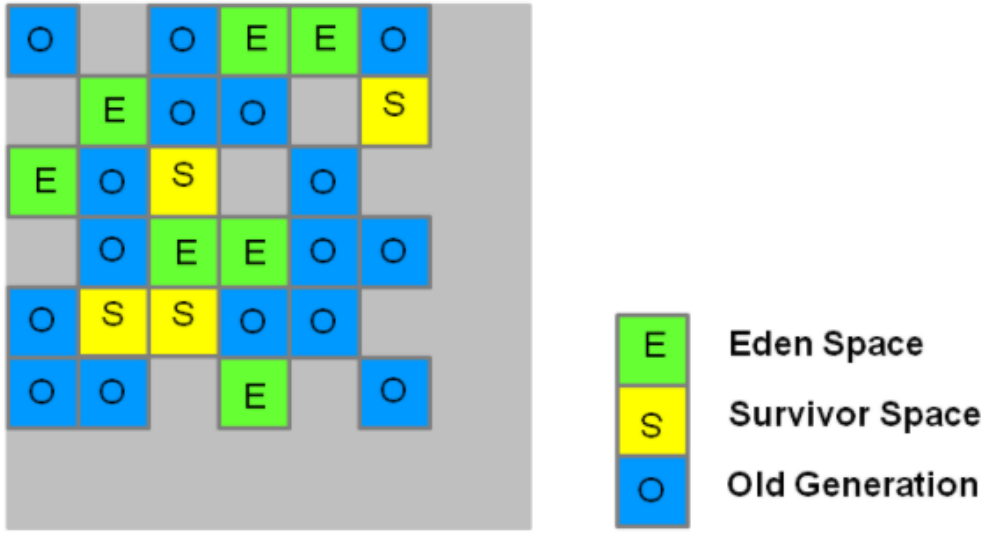


JAVA 7 NEW FEATURES

GARBAGE-FIRST COLLECTOR

eazy bytes

- The G1 collector takes a different approach. The heap is partitioned into a set of equal-sized heap regions, each a contiguous range of virtual memory. Certain region sets are assigned the same roles (eden, survivor, old) as in the older collectors, but there is not a fixed size for them. This provides greater flexibility in memory usage.



JAVA 7 NEW FEATURES

GARBAGE-FIRST COLLECTOR

eazy bytes

- Compared to most other garbage collectors, the G1 has two big advantages:
  - ✓ It can do most of its work concurrently (i.e., without halting application threads), and
  - ✓ Splitting the heap into small regions enables the G1 to select a small group of regions to collect and finish quickly.
  - ✓ One of the good property of this is you can configure this for maximum pause time using flag (-XX:MaxGCPauseMillis=n)

- G1 will be more suitable for users running applications that require large heaps with limited GC latency. This means heap sizes of around 6GB or larger, and stable and predictable pause time below 0.5 seconds.

- From Java 9, the Garbage First (G1) GC as its default garbage collector.

JAVA 7 NEW FEATURES

FORK & JOIN FRAMEWORK

eazy bytes

- With advancements happening in multicore processors and GPUs, there is a popular demand from Java community for parallel computing framework that can leverage all the processor available efficiently.

- To address that Java 7 introduces new Fork & Join framework that supports parallel processing by splitting (forking) big tasks into a multiple small subtasks that can be processed independently by using all the available CPU cores and eventually joining the results from all the subtasks to get the final results.

- Sample pseudo code of Fork & Join framework,

```
Result solve(Problem problem) {

    if (problem is small)
      directly solve problem
    else {
     split problem into independent parts
     fork new subtasks to solve each part
     join all subtasks
     compose result from subresults
   }
}
```

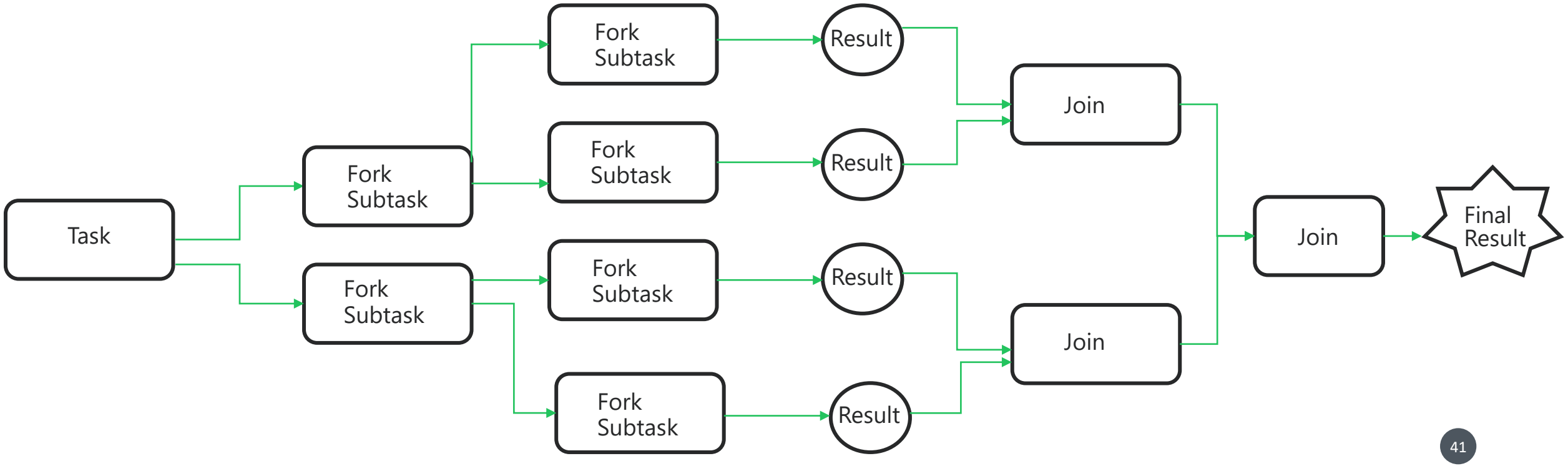
JAVA 7 NEW FEATURES

FORK & JOIN FRAMEWORK

eazy bytes

- Applying a divide and conquer principle, the framework recursively divides the task into smaller subtasks until a given threshold is reached. This is the fork part.

- Then, the subtasks are processed independently and if they return a result, all the results are recursively combined into a single result. This is the join part.



JAVA 7 NEW FEATURES

FORK & JOIN FRAMEWORK

eazy bytes

- Important interfaces and classes inside the Fork & Join framework,

  - ✓ java.util.concurrent.ForkJoinPool - The ForkJoinPool is the heart of the framework. It is an implementation of the ExecutorService that manages worker threads and provides us with tools to get information about the thread pool state and performance.

  - ✓ java.util.concurrent.ForkJoinTask - ForkJoinTask is the base type for tasks executed inside ForkJoinPool. In practice, one of its two subclasses should be extended: the RecursiveAction for void tasks and the RecursiveTask<V> for tasks that return a value. They both have an abstract method compute() in which the task's logic is defined.

- ForkJoin framework follows 'Work Stealing Algorithm' where free threads try to "steal" work from deques of busy threads

- In fact, Java 8's parallel streams underline uses the ForkJoin framework.

JAVA 7 SUMMARY

eazy bytes

01

THE TRY-WITH-RESOURCES STATEMENT

JVM handles closing the resources opened in the code automatically after the try block by calling close().

02

SUPPRESSED EXCEPTIONS

Handles the scenarios where we get multiple exceptions by holding all the subsequent exceptions inside an array.

03

CATCHING MULTIPLE EXCEPTIONS

A single catch block can handle multiple exceptions using a | operator if we have similar action need to be taken.

04

RETHROWING EXCEPTIONS WITH TYPE CHECKING

Enables to specify more specific exception types in the throws clause of a method declaration.

05

EASIER EXCEPTION HANDLING FOR REFLECTIONS

Introduces a single exception that replace multiple exceptions that need to be handles when we are invoking methods using reflections.

JAVA 7 SUMMARY

eazy bytes

06

OBJECTS CLASS & NULL CHECKS

requireNonNull methods inside Objects class can be used to check null values of an Object

07

CLOSE METHOD INSIDE URLCLASSLOADER

Provided a new close method inside the URLClassLoader which will be invoked by JVM to avoid memory leaks.

08

ENHANCEMENTS TO FILES & DIRECTORIES

New interfaces, classes like Path, Paths, Files introduced to make the operations on files and directories easy.

09

WATCHSERVICE

Provides easier approach to track any directory/ files changes instead of using Threads.

10

BINARY LITERALS & STRING IN SWITCH STATEMENT

Binary representation of numbers can be easily transferred /assigned to decimal representation using a prefix 0b/0B. And Strings are allowed inside the Switch statements

JAVA 7 SUMMARY

eazy bytes

11

TYPE INFERENCE/DIAMOND OPERATOR

Simplifies the use of generics when creating an object

12

USING UNDERSCORE IN NUMERIC LITERALS

Improves the readability of the long numbers as it allows '_'
inside the numeric values.

13

JDBC IMPROVEMENTS

JDBC library is optimized to use the try-with-resources features and added new approaches to create RowSet classes.

14

GARBAGE-FIRST COLLECTOR

A new garbage framework is build to handle applications that have large heap sizes.

15

FORK & JOIN FRAMEWORK

A new framework that will process huge tasks parallelly by levering all the CPU cores available.

JAVA 7 NEW FEATURES

RELEASE NOTES & GITHUB LINK

eazy bytes

Apart from the discussed new features, there are many security, non developer focused features are introduced in Java 7. For more details on them, please visit the link

https://www.oracle.com/java/technologies/javase/jdk7-relnotes.html

GitHub

https://github.com/eazybytes/Java-New-features/tree/main/Java7

THANK YOU & CONGRATULATIONS

YOU ARE NOW A MASTER OF JAVA 7 NEW FEATURES

eazy bytes