Java 9 □□□□□

Java 9 □□□□ 2017 □ 9 □ 21 □□□□ Java8 □□ 3 □□□□□□□□Java 9 □□□□□□□□□□□□□□□□ Java □□□□□□□□□□□□□□□

Java □□ □□□□

Java □□□□□□□□ Project Jigsaw□□□□□□□□□□ Java □□□□□□□□□□□□JDK □□□□□ 94 □□□Java □□□□□□□ jlink □□□

□□□□□□□□□□□□□□□□□□□□Java □□□□□□□□□□□□□□□□□□□□□□□□□□□□□ Apache Maven □ Gradle □□□□□□□□□□

Java 9 □□□□□□□□□□artifact□□□□□□□□□□□□□ module-info.class □ □□ □□□□□□□□□ JAR □□□ Java 9 □□□ JMOD □

- □□□□□□□ exports □□□□□□□□□□□□□□□□ public □ protected □□□□□□□□ public □ protected □□□□□□□□□□□
- □□□□□□□□ requires □□□□□□□□□□□□□□□ requires transitive □ □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ □
- □□□□□□□□□□□□□□□□□□ ServiceLocator □□□□□□□□ □□□□ provides with □□□□□□□□□ □□□□□□□□□□□□□□□□

□□1□□□□□□□□□□□□□□□□□□ c om.mycompany.sample □□ Java □ com.mycompany.sample□□□□□□□ c o
com.mycompany.common.DemoService □□□ c om.mycompany.sample.DemoServiceImpl □

□□ 1. □□□□□

```
module com.mycompany.sample {
    exports com.mycompany.sample;
    requires com.mycompany.common;
    provides com.mycompany.common.DemoService with
        com.mycompany.sample.DemoServiceImpl;
}
```

????

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ Java □□□□□CLASSPATH □□□□□□□□□□□□□□□□□□□□□□□ CLA

□ JVM □□JV

Jshell

jshell □ Java 9 □□□□□□□□□□jshell □ Java □□□□□ NodeJS □ Python □□□□-□□-□□□□□ Read-Evaluation-Print Loop □ □
□$2 □□□□□ □□□□□□□□□□□□□□□□□□□□□□□□ □ jshell □ □□□□□□□□□□□□□□□ Java □□□□□jshell □□□□□□□□□□□□

□□□□□□□2□□□□□□□□□□□ add□

□□ 2. □ jshell □□□□□

```
jshell> int add(int x, int y) {

    ...> return x + y;

    ...> }

 | created method add(int,int)
```

????

□□□□□□ jshell□□□3□□□□□□□□□□□

□□ 3. □ jshell □□□□□□□□

```
jshell> add(1, 2)

$19 ==> 3
```

????

□□□Stream □ Optional

□□□□□Java 9 □□ □ List.of()□Set.of()□Map.of() □ M ap.ofEntries()□□□□□□□□□□□□□ □□

□□ 4 . □□□□□□□

```
```

```
List.of();
List.of("Hello", "World");
List.of(1, 2, 3);
Set.of();
Set.of("Hello", "World");
Set.of(1, 2, 3);
Map.of();
Map.of("Hello", 1, "World", 2);
```

- ????

Stream □□□□□□□ ofNullable□dropWhile□takeWhile □□□iterate □□ 1 □ 5 □ □□□□□□□□□□□□□□□□□□□ 1 □□□□□□□□□□ 4 □□□

□□ 5 . Stream □□ dropWhile □□□□

```
@Test
public void testDropWhile() throws Exception {    final long count = Stream.of(1, 2, 3, 4,
5)
        .dropWhile(i -> i % 2 != 0)
        .count();
    assertEquals(4, count);
}
```

- ????

Collectors □□□□□□□□ filtering □ flatMapping□□□□□□String □ □□□□ flatMapping □ String □□□ Integer □ □□□□□□ Integer

□□ 6 . Collectors □ flatMapping □□□□

```
@Test
public void testFlatMapping() throws Exception {    final Set<Integer> result = Stream.of("a",
"ab", "abc")
        .collect(Collectors.flatMapping(v -> v.chars().boxed(),
Collectors.toSet()));
    assertEquals(3, result.size());
}
```

????

Optiona l □□□□ ifPresentOrElse□or □ stream□□□7□Optiona l □□□ 3 □ □□□□□□ 2 □□□□□□ flatMap □□□□□□□□□ 2 □□□

□□ 7 . Optional □ stream □□□□

```
@Test
public void testStream() throws Exception {
    final long count = Stream.of(
        Optional.of(1),
        Optional.empty(),
        Optional.of(2)
    ).flatMap(Optional::stream)
        .count();
    assertEquals(2, count);
}
```

????

□□ API

Java 9 □□□ ProcessHandle □□□□□□□□□□□□□□□□□□□□□□□□□□□□ P rocessBuilder □□□□□□□□□□□□ Process.toHa

ProcessHandle.Info □ □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ProcessHandle□□□onExit() API □□□□□□C ompletableFuture

□□ 8 . □□API □□

```
final ProcessBuilder processBuilder = new ProcessBuilder("top")
    .inheritIO();final ProcessHandle processHandle = processBuilder.start().toHandle();
processHandle.onExit().whenCompleteAsync((handle, throwable) -> {    if (throwable == null)
{
        System.out.println(handle.pid());
    } else {
        throwable.printStackTrace();
    }
});
```

- ????

□□□□ API □ □□

Java 9 □□□ JDK □□□□□□□□□□□□□□□ System.LoggerFinder □□□ JDK □ □□□□□□□□□JVM □□□□□□□□□□□□ LoggerFinde
java.util.logging □□□□□ LoggerFinder □ getLogger()□□□□□□□□□□□□□□□ System.Logger □□□□□□□□□□□ System.Lo

□□□□ JDK □□□□□□SLF4J□□□ API□□□□□□

□□ 9.□□□□□ API

```
public class Main {
    private static final System.Logger LOGGER = System.getLogger("Main");    public static void
main(final String[] args) {
        LOGGER.log(Level.INFO, "Run!");
    }
}
```

- ????

□□□□ □ Reactive Streams □

□□□□□□□□□□□□□□□□□□ □ Java □□□□□□□□□□ □ RxJava □ R eactor□□□□□□□□□□□□□□□□□□□□□□ non-blocking backpressu

Flow [illegible] Flow.Publisher[illegible]Flow.Subscriber[illegible]Flow.Subscription [illegible] F low.Processor [illegible] 4 [illegible]Java 9 [illegible] Subr
Flow.Publisher [illegible]RxJava 2 [illegible] Reactor [illegible] [illegible] Flow [illegible]

[illegible] J ava [illegible] java.lang.invoke.VarHanc
VarHandle [illegible] [illegible]VarHandle [illegible] [illegible] byte[][illegible] [illegible] Byte

[illegible]10[illegible] HandleTarget [illegible] count [illegible]

[illegible] 10. [illegible]

```
public class HandleTarget {
    public int count = 1;
}
public class VarHandleTest {
    private HandleTarget handleTarget = new HandleTarget();
    private VarHandle varHandle;
    @Before
    public void setUp() throws Exception {
        this.handleTarget = new HandleTarget();
        this.varHandle = MethodHandles
            .lookup()
            .findVarHandle(HandleTarget.class, "count", int.class);
    }
    @Test
    public void testGet() throws Exception {        assertEquals(1,
this.varHandle.get(this.handleTarget));        assertEquals(1,
this.varHandle.getVolatile(this.handleTarget));        assertEquals(1,
this.varHandle.getOpaque(this.handleTarget));        assertEquals(1,
this.varHandle.getAcquire(this.handleTarget));
    }}
```

????

□□□□□□□Method Handle□

□ java.lang.invoke.MethodHandles □□□□□□□□□□□□□□□□□□□□

- arrayConstructor□□□□□□□□□
- arrayLength□□□□□□□□□□□□
- varHandleInvoker □ varHandleExactInvoker□□□ VarHandle □□□□□□□□
- zero□□□□□□□□□□
- empty□□ □ MethodType □□□□□□□□□
- loop□countedLoop□iteratedLoop□whileLoop □ doWhileLoop□□□□□□□□□□□ for □□□while □□ □ do-while □□□
- tryFinally□□□□□□□□□□□ try-finally □□□□

□□□□□□11□□□□ iteratedLoop □□□□□□ S tring □□□□□□□□□□□□□□□□□□□□□□

□□ 11. □□□□□□□□□□

```
public class IteratedLoopTest {
    static int body(final int sum, final String value) {
        return sum + value.length();
    }
    @Test
    public void testIteratedLoop() throws Throwable {        final MethodHandle iterator =
MethodHandles.constant(
            Iterator.class,
            List.of("a", "bc", "def").iterator());        final MethodHandle init =
MethodHandles.zero(int.class);
        final MethodHandle body = MethodHandles
            .lookup()
            .findStatic(
                IteratedLoopTest.class,
                "body",
                MethodType.methodType(
                    int.class,
                    int.class,
                    String.class));        final MethodHandle iteratedLoop =
MethodHandles
            .iteratedLoop(iterator, init, body);        assertEquals(6,
iteratedLoop.invoke());
    }
}
```

????

□□□□□□ CompletableFuture □□□□□□□□□□□completeAsync □□□□□□□□□□□□□□□□□ CompletableFuture□orTimeout □
□□
TimeoutException □□□□□ CompletableFuture□completeOnTimeout □ o rTimeout □□□□□□□□□□□□□□□□□□□□□ Com
Thread.onSpinWai t □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Nashorn

Nashorn □ Java 8 □□□□□□ JavaScript □□□Java 9 □□ Nashorn □□□□□□□ ECMAScript 6 □□□□□□□□□□□□□□□□□□□□□□□□□□□□□
Syntax Tree□AST □ □□□□□□ ECMAScript □□□□□□□□

I/O □□□□

□ java.io.InputStream □□□□□□□□□□□□□□ InputStream □□□□□□□

- readAllBytes□□□ InputStream □□□□□□□□□
- readNBytes□ □ InputStream □□□□□□□□□□□□□□□
- transferTo□□□ InputStream □□□□□□□□□□□□□ OutputStream □ □

□□□□□12□□□□□□□□□□□□

□□ 12. InputStream □□□□□□□□□

```
public class TestInputStream {
    private InputStream inputStream;
    private static final String CONTENT = "Hello World";
    @Before
    public void setUp() throws Exception {
        this.inputStream =
TestInputStream.class.getResourceAsStream("/input.txt");
    }
    @Test
    public void testReadAllBytes() throws Exception {        final String content = new
String(this.inputStream.readAllBytes());
        assertEquals(CONTENT, content);
    }
    @Test
    public void testReadNBytes() throws Exception {        final byte[] data = new
byte[5];
        this.inputStream.readNBytes(data, 0, 5);        assertEquals("Hello", new
String(data));
    }
    @Test
    public void testTransferTo() throws Exception {        final ByteArrayOutputStream
outputStream = new ByteArrayOutputStream();
this.inputStream.transferTo(outputStream);        assertEquals(CONTENT,
outputStream.toString());
    }
}
```

????

ObjectInputFilter □□□ ObjectInputStream □ □□□□□□□□□□□□□□□□□□ ObjectInputStream □□□ setObjec

Java 9 □□□ 4 □ SHA- 3 □□□□SHA3-224□SHA3-256□SHA3-384 □ S HA3-512□□□□□□□□ java.security.SecureRa
SHA-3 □□□□□□□

□□□□□□□□

□□ 13. SHA-3 □□□□□□□

```
import org.apache.commons.codec.binary.Hex;
public class SHA3 {    public static void main(final String[] args) throws
NoSuchAlgorithmException {        final MessageDigest instance = MessageDigest.getInstance("SHA3-
224");
        final byte[] digest = instance.digest("".getBytes());
System.out.println(Hex.encodeHexString(digest));
    }
}
```

????

[illegible] java.awt.Desktop [illegible] addAppEventListener [illegible]

[illegible]

[illegible] JVM [illegible]

Java 9 [illegible] JVM [illegible]-Xlog [illegible] JVM [illegible]Java 9

[illegible] Java [illegible]Java 9 [illegible] try-with-resources [illegible] e ffectively-final [illegible] java.lang.StackWalke

[illegible]

8.0[illegible]ResourceBundle [illegible] ISO-8859-1 [illegible] UTF-8[illegible] native2ascii [illegible]@Deprecat

[illegible] 14 [illegible]buildMessage [illegible] SayHi [illegible] sayHi [illegible]

[illegible] 14. [illegible]

```
public interface SayHi {
    private String buildMessage() {
        return "Hello";
    }
    void sayHi(final String message);
    default void sayHi() {
        sayHi(buildMessage());
    }
}
```

????

□□ Java □□□□□□□□□□□Java 9 □□□□□□□□□□□□□□□□ Java □□□□□□□□□□□□□□□□ Java 9 □□□□□□□□□□□□□□□□□□ Java 9 □□□□

□□□

□□□https://developer.ibm.com/zh/articles/the-new-features-of-java-9/

Revision #3
Created Fri, Oct 16, 2020 2:27 AM by □□□
Updated Fri, Oct 16, 2020 2:31 AM by □□□