Oracle

1Z0-829 Exam

Java SE 17 Developer

Thank you for Downloading 1Z0-829 exam PDF Demo

You can Buy Latest 1Z0-829 full version download

https://www.certkillers.net/Exam/1Z0-829

Version: 4.1

Question: 1

Given the code fragments:

```
class Test {
    volatile int x = 1;
    AtomicInteger xObj = new AtomicInteger(1);
}
```

and

```
public static void main(String[] args) {
    Test t = new Test();
    Runnable r1 = () -> {
        Thread trd = Thread.currentThread();
        while (t.x < 3 ) {
            System.out.print(trd.getName()+" : "+t.x+" : ");
            t.x++;
       }
    };
    Runnable r2 = () -> {
        Thread trd = Thread.currentThread();
        while (t.xObj.get() < 3) {
            System.out.print(trd.getName()+" : "+t.xObj.get()+" : ");
            t.xObj.getAndIncrement();
        }
    };
    Thread t1 = new Thread(r1,"t1");
    Thread t2 = new Thread(r2,"t2");
    t1.start();
    t2.start();
}
```

Which is true?

A. The program prints t1 : 1: t2 : 1: t1 : t2 : 2 : in random order.
B. The program prints t1 : 1 : t2: 1 : t1 : 2 : t2: 2:
C. The program prints t1 : 1: t2 : 1: t1 : 1 : t2 : 1 : indefinitely
D. The program prints an exception

Answer: B

Explanation:

The code creates two threads, t1 and t2, and starts them. The threads will print their names and the value of the Atomic Integer object, x, which is initially set to 1. The threads will then increment the value of x and print their names and the new value of x. Since the threads are started at the same time, the output will be in random order. However, the final output will always be t1 : 1 : t2: 1 : t1 : 2 : t2: 2: Reference: AtomicInteger (Java SE 17 & JDK 17) - Oracle

Question: 2

Which statement is true?

A. IllegalStateException is thrown if a thread in waiting state is moved back to runnable.
B. thread in waiting state consumes CPU cycles.
C. A thread in waiting state must handle InterrupedException.
D. After the timed wait expires, the waited thread moves to the terminated state.

Answer: C

Explanation:

A thread in waiting state is waiting for another thread to perform a particular action, such as calling notify() or notifyAll() on a shared object, or terminating a joined thread. A thread in waiting state can be interrupted by another thread, which will cause the waiting thread to throw an InterruptedException and return to the runnable state. Therefore, a thread in waiting state must handle InterruptedException, either by catching it or declaring it in the throws clause. Reference: Thread.State (Java SE 17 & JDK 17), [Thread (Java SE 17 & JDK 17)]

Question: 3

Given the code fragment:

```
ExecutorService executorService = Executors.newSingleThreadExecutor();
Set<Callable<String>> workers = new HashSet<Callable<String>>();
workers.add(new Callable<String>() {
    public String call() throws Exception {
        return "1";
    }
});
workers.add(new Callable<String>() {
    public String call() throws Exception {
        return "2";
    }
});
workers.add(new Callable<String>() {
    public String call() throws Exception {
        return "3";
    }
});
```

Which code fragment invokes all callable objects in the workers set?

A)

```
List<Future<String>> futures = executorService.invokeAny(workers);
for(Future<String> future : futures){
    System.out.println(future.get());
}
```

B)

```
executorService.submit(cThreads);
```

C)

```
List<Future<String>> futures = executorService.invokeAll(workers);
for(Future<String> future : futures){
    System.out.println(future.get());
}
```

D)

```
for (int i=0; i<3;i++) {
    String result = executorService.invokeAny(cThreads);
    System.out.println(result);
}
```

A. Option A
B. Option B
C. Option C
D. Option D

Answer: C

Explanation:

The code fragment in Option C invokes all callable objects in the workers set by using the ExecutorService's invokeAll() method. This method takes a collection of Callable objects and returns a list of Future objects representing the results of the tasks. The other options are incorrect because they either use the wrong method (invokeAny() or submit()) or have syntax errors (missing parentheses or semicolons). Reference: AbstractExecutorService (Java SE 17 & JDK 17) - Oracle

Question: 4

Given the code fragment:

```
String a = "Hello! Java";
System.out.print(a.indexOf("Java"));
a.replace("Hello!", "Welcome!");
System.out.print(a.indexOf("Java"));
StringBuilder b = new StringBuilder(a);
System.out.print(b.indexOf("Java"));
```

What is the result?

A. 81111
B. 8109
C. 777
D. 71010
E. 888
F. 7107

Answer: C

Explanation:

The code fragment is creating a string variable "a" with the value "Hello! Java". Then, it is printing the index of "Java" in "a". Next, it is replacing "Hello!" with "Welcome!" in "a". Then, it is printing the index of "Java" in "a". Finally, it is creating a new StringBuilder object "b" with the value of "a" and printing the index of "Java" in "b". The output will be 8109 because the index of "Java" in "a" is 8, the index of "Java" in "a" after replacing "Hello!" with "Welcome!" is 10, and the index of "Java" in "b" is 9. Reference: Oracle Java SE 17 Developer source and documents: [String (Java SE 17 & JDK 17)], [StringBuilder (Java SE 17 & JDK 17)]

Question: 5

Given the code fragment:

```
String myStr = "Hello Java 17";
String myTextBlk1 = """
                    Hello Java 17""";
String myTextBlk2 = """
                    Hello Java 17
                    """;
System.out.print(myStr.equals(myTextBlk1)+":");
System.out.print(myStr.equals(myTextBlk2)+":");
System.out.print(myTextBlk1.equals(myTextBlk2)+":");
System.out.println(myTextBlk1.intern() == myTextBlk2.intern());
```

A. True:false:true:true
B. True:true:false:false
C. True:false:true:false
D. True:false:false:false

Answer: D

Explanation:

The code fragment compares four pairs of strings using the equals() and intern() methods. The equals() method compares the content of two strings, while the intern() method returns a canonical representation of a string, which means that it returns a reference to an existing string with the same content in the string pool. The string pool is a memory area where strings are stored and reused to save space and improve performance. The results of the comparisons are as follows:
s1.equals(s2): This returns true because both s1 and s2 have the same content, “Hello Java 17”.
s1 == s2: This returns false because s1 and s2 are different objects with different references, even though they have the same content. The == operator compares the references of two objects, not their content.
s1.intern() == s2.intern(): This returns true because both s1.intern() and s2.intern() return a reference to the same string object in the string pool, which has the content “Hello Java 17”. The intern() method ensures that there is only one copy of each distinct string value in the string pool.
“Hello Java 17” == s2: This returns false because “Hello Java 17” is a string literal, which is automatically interned and stored in the string pool, while s2 is a string object created with the new operator, which is not interned by default and stored in the heap. Therefore, they have different references and are not equal using the == operator.
Reference: String (Java SE 17 & JDK 17) - Oracle

Thank You for trying 1Z0-829 PDF Demo

To try our 1Z0-829 full version download visit link below

https://www.certkillers.net/Exam/1Z0-829

Start Your 1Z0-829 Preparation

[Limited Time Offer] Use Coupon "CKNET" for Further discount on your purchase. Test your 1Z0-829 preparation with actual exam questions.