

Oracle

Exam Questions 1z0-829

Java SE 17 Developer



NEW QUESTION 1

Which statement is true?

- A. The tryLock () method returns a boolean indicator immediately regardless if it has or has not managed to acquire the lock.
- B. The tryLock () method returns a boolean indicator immediately if it has managed to acquire the lock, otherwise it waits for the lock acquisition.
- C. The lock () method returns a boolean indicator immediately if it has managed to acquire the lock, otherwise it waits for the lock acquisition.
- D. The Lock () method returns a boolean indicator immediately regardless if it has or has not managed to acquire the lock

Answer: A

Explanation:

The tryLock () method of the Lock interface is a non-blocking attempt to acquire a lock. It returns true if the lock is available and acquired by the current thread, and false otherwise. It does not wait for the lock to be released by another thread. This is different from the lock () method, which blocks the current thread until the lock is acquired, and does not return any value. References: [https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/concurrent/locks/Lock.html#tryLock\(\)](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/concurrent/locks/Lock.html#tryLock()), 3, 4, 5

NEW QUESTION 2

Given the code fragment:

```
List lst = new ArrayList();
lst.add("e1");
lst.add("e3");
lst.add("e2");

int x1 = Collections.binarySearch(lst, "e3");
System.out.println(x1);
Collections.sort(lst);
int x2 = Collections.binarySearch(lst, "e3");
System.out.println(x2);

Collections.reverse(lst);
int x3 = Collections.binarySearch(lst, "e3");
System.out.println(x3);
```

What is the result?

- A. 2
- B. -2
- C. 22E.111F.12-4

Answer: B

Explanation:

The code fragment uses the Collections.binarySearch method to search for the string "e3" in the list. The first search returns the index of the element, which is 2. The second search returns the index of the element, which is 0. The third search returns the index of the element, which is -4. The final result is 2. References: Collections (Java SE 17 & JDK 17) - Oracle

NEW QUESTION 3

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: C

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.

References: String (Java SE 17 & JDK 17) - Oracle

NEW QUESTION 4

Given the product class:

```
import java.io.*;
public class Product implements Serializable {
    private static float averagePrice = 2.99f;
    private String description;
    private transient float price;
    public Product(String description, float price) {
        this.description = description;
        this.price = price;
    }
    public void readObject(ObjectInputStream in)
        throws IOException, ClassNotFoundException {
        in.defaultReadObject();
        price = averagePrice;
    }
    public String toString() {
        return description+" "+price+" "+averagePrice;
    }
}
```

And the shop class:

```
import java.io.*;
public class Shop {
    public static void main(String[] args) {
        Product p = new Product("Cookie", 3.99f);
        try {
            try (ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream("p.ser"))) {
                out.writeObject(p);
            }
            try (ObjectInputStream in = new ObjectInputStream(new FileInputStream("p.ser"))) {
                p = (Product)in.readObject();
            }
        } catch (Exception e) { e.printStackTrace(); }
        System.out.println(p);
    }
}
```

What is the result?

- A. Cookie 2.99 2.99
- B. Cookie 3.99 2.99
- C. Cookie 0.0 0.0
- D. An exception is produced at runtime
- E. Compilation fails
- F. Cookie 0.0 2.99

Answer: E

Explanation:

The code fragment will fail to compile because the readObject method in the Product class is missing the @Override annotation. The readObject method is a special method that is used to customize the deserialization process of an object. It must be declared as private, have no return type, and take a single parameter of type ObjectInputStream. It must also be annotated with @Override to indicate that it overrides the default behavior of the ObjectInputStream class. Without the @Override annotation, the compiler will treat the readObject method as a normal method and not as a deserialization hook. Therefore, the code fragment will produce a compilation error. References: Object Serialization - Oracle, [ObjectInputStream (Java SE 17 & JDK 17) - Oracle]

NEW QUESTION 5

Given the code fragment:

```
Stream<String> s1 = Stream.of("A", "B", "C", "B");
Stream<String> s2 = Stream.of("A", "D", "E");
Stream.concat(s1, s2).parallel().distinct().forEach(element -> System.out.print(element));
```

What is the result:

- A. ADEABCB // the order of element is unpredictable
- B. ABCE
- C. ABCDE // the order of elements is unpredictable
- D. ABBCDE // the order of elements is unpredictable

Answer: D

Explanation:

The answer is D because the code fragment uses the Stream API to create two streams, s1 and s2, and then concatenates them using the concat() method. The resulting stream is then processed in parallel using the parallel() method, and the distinct() method is used to remove duplicate elements. Finally, the forEach() method is used to print the elements of the resulting stream to the console. Since the order of elements in a parallel stream is unpredictable, the output could be any of the options given, but option D is the most likely. References:

? Oracle Certified Professional: Java SE 17 Developer

? Java SE 17 Developer

? OCP Oracle Certified Professional Java SE 17 Developer Study Guide

? Parallelizing Streams

NEW QUESTION 6

Given:

```
final class Folder {    // line n1
    // line n2
    public void open(){
        System.out.print("Open ");
    }
}

public class Test {
    public static void main(String[] args) throws Exception {
        try (Folder f = new Folder()) {
            f.open();
        }
    }
}
```

Which two modifications enable the code to print Open Close?

A)

At line n2, insert:

```
final void close() {
    System.out.print("Close ");
}
```

B)


```
Replace line n1 with:  
class Folder extends Closeable {
```

C)

```
Replace line n1 with:  
class Folder extends Exception {
```

D)

```
Replace line n1 with:  
class Folder implements AutoCloseable {
```

E)

```
At line n2, insert:  
public void close() throws IOException {  
    System.out.print("Close ");  
}
```

- A. Option A
- B. Option B
- C. Option C
- D. Option D
- E. Option E

Answer: BE

Explanation:

The code given is a try-with-resources statement that declares a resource of type AutoCloseable. The resource is an anonymous class that implements the AutoCloseable interface and overrides the close() method. The code also has a print() method that prints the value of the variable s. The code is supposed to print ??Open Close??, but it does not compile because of two errors.

The first error is at line n1, where the anonymous class is missing a semicolon at the end of its declaration. This causes a syntax error and prevents the code from compiling. To fix this error, option B adds a semicolon after the closing curly brace of the anonymous class.

The second error is at line n2, where the print() method is called without an object reference. This causes a compilation error because the print() method is not static and cannot be invoked without an object. To fix this error, option E adds an object reference to the print() method by using the variable t.

Therefore, options B and E are correct and enable the code to print ??Open Close??.

NEW QUESTION 7

Given the code fragment:

```
String s = "10_00";  
Integer s2 = 10_00;  
// Line n1  
System.out.println(res);
```

Which two statements at Line n1 independently enable you to print 1250?

- A. Integer res = 250 + integer.parseInt (s)
- B. Integer res = 250 + s;
- C. Integer res = 250 + integer (s2):
- D. Integer res= 250 + s2;
- E. Integer res = 250 + integer . valueOf (s);
- F. Integer res = 250; Res = + s2;

Answer: AE

Explanation:

The code fragment is creating a string variable `s` with the value `"10_00"` and an integer variable `s2` with the value 10. The string `s` is using an underscore as a separator for readability, which is allowed in Java SE 17. The question is asking for two statements that can add 250 to the numeric value of `s` and assign it to an integer variable `res`. The correct answers are A and E because they use the methods `parseInt` and `valueOf` of the `Integer` class to convert the string `s` to an integer. Both methods interpret the string as a signed decimal integer and return the equivalent `int` or `Integer` value. The other options are incorrect because they either use invalid syntax, such as B and C, or they do not convert the string `s` to an integer, such as D and F. References: Binary Literals (The Java™ Tutorials > Learning the Java Language > Numbers and Strings), Integer (Java SE 17 & JDK 17), Integer (Java SE 17 & JDK 17)

NEW QUESTION 8

Given the code fragment:

```
int a = 2;  
int b = ~a;  
int c = a^b;  
boolean d = a < b & a > c++;  
System.out.println(d + " " + c);  
boolean e = a > b && a > c++;  
System.out.println(e + " " + c);
```

What is the result?

- A. false 1false 2
- B. true 1false 2
- C. false 1ture 2
- D. falase 0true 1

Answer: B

Explanation:

The code fragment is comparing the values of `a`, `b`, and `c` using the `<` and `>` operators. The first comparison, `d`, is checking if `a` is less than `b` and greater than `c`. Since `a` is equal to 2, `b` is equal to -2, and `c` is equal to -4, this comparison will evaluate to true. The second comparison, `e`, is checking if `a` is greater than `b` and `a` is greater than `c`. Since `a` is equal to 2, `b` is equal to -2, and `c` is equal to -4, this comparison will evaluate to false. Therefore, the result will be true 1 false 2. References: Operators (The Java™ Tutorials > Learning the Java Language - Oracle)

NEW QUESTION 9

Given:

```
package com.transport.vehicle.cars;

public interface Car {
    int getSpeed();
}

and

package com.transport.vehicle.cars.impl;

import com.transport.vehicle.cars.Car;

public class CarImpl implements Car {
    private int speed;

    public CarImpl() {
        this(10);
    }

    public CarImpl (int speed) {
        this.speed = speed;
    }

    @Override
    public int getSpeed() {
        return speed;
    }
}
```

Which two should the module-info file include for it to represent the service provider interface?

- A. Requires cm.transport.vehicle,cars:
- B. Provides com.transport.vehicle.cars.Car with com.transport.vehicle.car
- C. impt, CatImpl;
- D. Requires cm.transport.vehicle,cars:
- E. Provides com.transport.vehicle.cars.Car impl,CarImp1 to com.transport.vehicle.car
- F. Cars
- G. exports com.transport.vehicle.cars.Car;
- H. Exports com.transport.vehicle.cars;
- I. Exports com.transport.vehicle;

Answer: BE

Explanation:

The answer is B and E because the module-info file should include a provides directive and an exports directive to represent the service provider interface. The provides directive declares that the module provides an implementation

of a service interface, which is `com.transport.vehicle.cars.Car` in this case. The `with` clause specifies the fully qualified name of the service provider class, which is `com.transport.vehicle.cars.impl.CarImpl` in this case. The `exports` directive declares that the module exports a package, which is `com.transport.vehicle.cars` in this case, to make it available to other modules. The package contains the service interface that other modules can use.

Option A is incorrect because `requires` is not the correct keyword to declare a service provider interface. `Requires` declares that the module depends on another module, which is not the case here.

Option C is incorrect because it has a typo in the module name. It should be `com.transport.vehicle.cars`, not `cm.transport.vehicle.cars`.

Option D is incorrect because it has a typo in the keyword `provides`. It should be `provides`, not `Provides`. It also has a typo in the service interface name. It should be `com.transport.vehicle.cars.Car`, not `com.transport.vehicle.cars.Car impl`. It also has an unnecessary `to` clause, which is used to limit the accessibility of an exported package to specific modules.

Option F is incorrect because it exports the wrong package. It should export `com.transport.vehicle.cars`, not `com.transport.vehicle.cars.impl`. The `impl` package contains the service provider class, which should not be exposed to other modules.

Option G is incorrect because it exports the wrong package. It should export `com.transport.vehicle.cars`, not `com.transport.vehicle`. The `vehicle` package does not contain the service interface or the service provider class. References:

? Oracle Certified Professional: Java SE 17 Developer

? Java SE 17 Developer

? OCP Oracle Certified Professional Java SE 17 Developer Study Guide

? Java Modules - Service Interface Module - GeeksforGeeks

? Java Service Provider Interface | Baeldung

NEW QUESTION 10

Daylight Saving Time (DST) is the practice of advancing clocks at the start of spring by one hour and adjusting them backward by one hour in autumn.

Considering that in 2021, DST in Chicago (Illinois) ended on November 7th at 2 AM, and given the fragment:

```
ZoneId zoneID = ZoneId.of("America/Chicago");
ZonedDateTime zdt = ZonedDateTime.of(
    LocalDate.of(2021, 11, 7),
    LocalTime.of(1, 30),
    zoneID
);
ZonedDateTime anHourLater = zdt.plusHours(1);
System.out.println(zdt.getHour() == anHourLater.getHour());
System.out.print(zdt.getOffset().equals(anHourLater.getOffset()));
```

What is the output?

- A. true false
- B. False false
- C. true true
- D. false true

Answer: A

Explanation:

The answer is A because the code fragment uses the `ZoneId` and `ZonedDateTime` classes to create two date-time objects with the same local date-time but different zone offsets. The `ZoneId` class represents a time-zone ID, such as `America/Chicago`, and the `ZonedDateTime` class represents a date-time with a time-zone in the ISO-8601 calendar system. The code fragment creates two `ZonedDateTime` objects with the same local date-time of `2021-11-07T01:30`, but different zone IDs of `America/Chicago` and `UTC`. The code fragment then compares the two objects using the `equals` and `isEqual` methods.

The `equals` method compares the state of two objects for equality. In this case, it compares the local date-time, zone offset, and zone ID of the two `ZonedDateTime` objects. Since the zone offsets and zone IDs are different, the `equals` method returns `false`.

The `isEqual` method compares the instant of two temporal objects for equality. In this case, it compares the instant of the two `ZonedDateTime` objects, which is derived from the local date-time and zone offset. Since DST in Chicago ended on November 7th at 2 AM in 2021, the local date-time of `2021-11-07T01:30` in `America/Chicago` corresponds to the same instant as `2021-11-07T06:30` in `UTC`. Therefore, the `isEqual` method returns `true`.

Hence, the output is `true false`. References:

? Oracle Certified Professional: Java SE 17 Developer

? Java SE 17 Developer

? OCP Oracle Certified Professional Java SE 17 Developer Study Guide

? `ZoneId` (Java Platform SE 8)

? `ZonedDateTime` (Java Platform SE 8)

? Time Zone & Clock Changes in Chicago, Illinois, USA

? Daylight Saving Time Changes 2023 in Chicago, USA

NEW QUESTION 10

.....

Thank You for Trying Our Product

We offer two products:

1st - We have Practice Tests Software with Actual Exam Questions

2nd - Questions and Answers in PDF Format

1z0-829 Practice Exam Features:

- * 1z0-829 Questions and Answers Updated Frequently
- * 1z0-829 Practice Questions Verified by Expert Senior Certified Staff
- * 1z0-829 Most Realistic Questions that Guarantee you a Pass on Your FirstTry
- * 1z0-829 Practice Test Questions in Multiple Choice Formats and Updatesfor 1 Year

100% Actual & Verified — Instant Download, Please Click
[Order The 1z0-829 Practice Test Here](#)