

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:

```
class Product {
    String name; double price;
    Product(String s, double d) {
        this.name = s;
        this.price = d;
    }
}

class ElectricProduct extends Product {
    ElectricProduct(String name, double price) {
        super(name, price);
    }
}
```

and the code fragment:

```
List<Product> p = List.of(
    new ElectricProduct("CellPhone",100),
    new ElectricProduct("ToyCar",90),
    new ElectricProduct("Motor",200),
    new ElectricProduct("Fan",300)
);

DoubleSummaryStatistics sts = p.stream().filter(a -> a instanceof ElectricProduct)
                                   .collect(Collectors.summarizingDouble(a ->
a.price));
String s1 = p.stream().filter(a -> a instanceof Product)
               .collect(Collectors.mapping(p2 -> p2.name, Collectors.joining(",")));
System.out.println(sts.getMax());
System.out.println(s1);
```

- A. 300.00CellPhone,ToyCar,Motor,Fan
- B. 100.00CellPhone,ToyCar,Motor,Fan
- C. 100.00 CellPhone,ToyCar
- D. 300.00CellPhone.ToyCar

Answer: A

Explanation:

The code fragment is using the Stream API to perform a reduction operation on a list of ElectricProduct objects. The reduction operation consists of three parts: an identity value, an accumulator function, and a combiner function. The identity value is the initial value of the result, which is 0.0 in this case. The accumulator function is a BiFunction that takes two arguments: the current result and the current element of the stream, and returns a new result. In this case, the accumulator function is (a,b) -> a + b.getPrice (), which means that it adds the price of each element to the current result. The combiner function is a BinaryOperator that takes two partial results and combines them into one. In this case, the combiner function is (a,b) -> a + b, which means that it adds the two partial results together. The code fragment then applies a filter operation on the stream, which returns a new stream that contains only the elements that match the given predicate. The predicate is p -

> p.getPrice () > 10, which means that it selects only the elements that have a price greater than 10. The code fragment then applies a map operation on the filtered stream, which returns a new stream that contains the results of applying the given function to each element. The function is p -> p.getName (), which means that it returns the name of each element.

The code fragment then calls the collect method on the mapped stream, which performs a mutable reduction operation on the elements of the stream using a Collector. The Collector is Collectors.joining (??,??), which means that it concatenates the elements of the stream into a single String, separated by commas.

The code fragment then prints out the result of the reduction operation and the result of the collect operation, separated by a new line. The result of the reduction operation is 300.00, which is the sum of the prices of all ElectricProduct objects that have a price greater than 10. The result of the collect operation is CellPhone,ToyCar,Motor,Fan, which is the concatenation of the names of all ElectricProduct objects that have a price greater than 10. Therefore, the output of the code fragment is: 300.00 CellPhone,ToyCar,Motor,Fan

References: Stream (Java SE 17 & JDK 17) - Oracle, Collectors (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

Which statement is true about modules?

- A. Automatic and unnamed modules are on the module path.
- B. Only unnamed modules are on the module path.
- C. Automatic and named modules are on the module path.
- D. Only named modules are on the module path.
- E. Only automatic modules are on the module path.

Answer: C

Explanation:

A module path is a sequence of directories that contain modules or JAR files. A named module is a module that has a name and a module descriptor (module-info.class) that declares its dependencies and exports. An automatic module is a module that does not have a module descriptor, but is derived from the name and contents of a JAR file. Both named and automatic modules can be placed on the module path, and they can be resolved by the Java runtime. An unnamed module is a special module that contains all the classes that are not in any other module, such as those on the class path. An unnamed module is not on the module path, but it can read all other modules.

NEW QUESTION 6

Given:


```
public enum Desig {
    CEO('A'), CMO('B'), CTO('C'), CFO('D');
    char c;
    private Desig(char c) {
        this.c = c;
    }
}
```

and the code fragment:

```
Arrays.stream(Desig.values()).dropWhile(s -> s.equals(Desig.CMO));
switch (Desig.valueOf("CMO")) {
    case CEO -> System.out.println("Executive");
    case CMO -> System.out.println("Marketing");
    case CFO -> System.out.println("Finance");
    case CTO -> System.out.println("Technical");
    default -> System.out.println("UnDefined");
}
```

What is the result

- A. Marketing Finance Technical
- B. Marketing Undefined
- C. UnDefined
- D. Marketing

Answer: C

Explanation:

The code fragment is using the switch statement with the new Java 17 syntax. The switch statement checks the value of the variable `desig` and executes the corresponding case statement. In this case, the value of `desig` is `CTO`, which does not match any of the case labels. Therefore, the default case statement is executed, which prints `UnDefined`. The other case statements are not executed, because there is no fall through in the new syntax. Therefore, the output of the code fragment is: `UnDefined`

NEW QUESTION 7

Given the code fragment:

```
abstract sealed interface SInt permits Story, Art {
    default String getTitle() { return "Book Title" ; }
}
```

```
abstract sealed interface SInt permits Story, Art { default String getTitle() { return "Book Title" ; }
}
```

Which set of class definitions compiles?

- A. Interface story extends STnt {} Interface Art extends SInt {}
- B. Public interface story extends sInt {} Public interface Art extends SInt {}
- C. Sealed interface Story extends sInt {} Non-sealed class Art implements SInt {}
- D. Non-sealed interface story extends SInt {} Class Art implements SInt {}
- E. Non-sealed interface story extends SInt {} Non-sealed interaface Art extends SInt {}

Answer: C

Explanation:

The answer is C because the code fragment given is an abstract sealed interface `SInt` that permits `Story` and `Art`. The correct answer is option C, which is a sealed interface `Story` that extends `SInt` and a non-sealed class `Art` that implements `SInt`. This is because a sealed interface can only be extended by the classes or interfaces that it permits, and a non-sealed class can implement a sealed interface.

Option A is incorrect because interface is misspelled as `interace`, and `Story` and `Art` should be capitalized as they are the names of the permitted classes or interfaces.

Option B is incorrect because `public` is misspelled as `public`, and `sInt` should be `SInt` as it is the name of the sealed interface.

Option D is incorrect because a non-sealed interface cannot extend a sealed interface, as it would violate the restriction of permitted subtypes.

Option E is incorrect because both `Story` and `Art` cannot be non-sealed interfaces, as they would also violate the restriction of permitted subtypes.

References:

? Oracle Certified Professional: Java SE 17 Developer

? Java SE 17 Developer

? OCP Oracle Certified Professional Java SE 17 Developer Study Guide
? Sealed Classes and Interfaces in Java 15 | Baeldung
? Sealed Class in Java - Javatpoint

NEW QUESTION 8

Given:

```
public class App{
    String name;
    public App(String name){
        this.name = name;
    }
    public static void main(String args[]) {
        App t1= new App("t1");
        App t2= new App("t2");
        t1 = t2;
        t1 = null;
        System.out.println("GC");
    }
}
```

Which statement is true while the program prints GC?

- A. Only the object referenced by t2 is eligible for garbage collection.
- B. Both the objects previously referenced by t1 are eligible for garbage collection.
- C. None of the objects are eligible for garbage collection.
- D. Only one of the objects previously referenced by t1 is eligible for garbage collection.

Answer: B

NEW QUESTION 9

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 10

Given the code fragment:

```
Duration duration = Duration.ofMillis(5000);  
System.out.print(duration);  
duration = Duration.ofSeconds(60);  
System.out.print(duration);  
Period period = Period.ofDays(6);  
System.out.print(period);
```

What is the result?

- A. \$SIM6D
- B. PT5000PT60MP6D
- C. PT5SPTIMP6D
- D. 5000\$60M6D

Answer: B

Explanation:

The code fragment is creating a Duration object with a value of 5000 milliseconds, then printing it. Then, it is creating another Duration object with a value of 60 seconds, then printing it. Finally, it is creating a Period object with a value of 6 days, then printing it. The output will be ??PT5000PT60MP6D??. References: <https://docs.oracle.com/javase/8/docs/api/java/time/Duration.html>, <https://docs.oracle.com/javase/8/docs/api/java/time/Period.html>

NEW QUESTION 10

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 15

Given the course table:

COURSE_ID	COURSE_NAME	COURSE_FEE	COURSE_LEVEL
1021	Java Programmer	400.00	1
1022	Java Architect	600.00	2
1023	Java Master	600.00	2

Given the code fragment:

```
try (Connection con = DriverManager.getConnection(connectionString)) {
    Statement statement = con.createStatement(TYPE_SCROLL_INSENSITIVE,ResultSet.CONCUR_UPDATABLE);
    String qry = "UPDATE course SET course_fee = ? where COURSE_LEVEL = ?";
    PreparedStatement prStmt = con.prepareStatement(qry, TYPE_SCROLL_INSENSITIVE);
    prStmt.setDouble(1,600.00);
    prStmt.setInt(2,2);
    System.out.println(prStmt.executeUpdate());
}
catch(SQLException sqlException) {
    System.out.println(sqlException);
}
```

- A. 2
- B. false
- C. true
- D. 1

Answer: C

Explanation:

The code fragment will execute the update statement and set the course fee of the course with ID 1021 to 5000. The executeUpdate method returns an int value that indicates the number of rows affected by the SQL statement. In this case, only one row will be updated, so the result variable will be 1. The if statement will check if the result is greater than 0, which is true, and print ??Updated successfully??. Therefore, the output of the code fragment is true. References:

https://education.oracle.com/products/trackp_OCPJSE17, <https://mylearn.oracle.com/ou/learning-path/java-se-17-developer/99487>,

[https://docs.oracle.com/en/java/javase/17/docs/api/java.sql/java/sql/Statement.html#executeUpdate\(java.lang.String\)](https://docs.oracle.com/en/java/javase/17/docs/api/java.sql/java/sql/Statement.html#executeUpdate(java.lang.String))

NEW QUESTION 18

Given the code fragment:

```
class Book {
    String author;
    String title;
    Book(String authorName, String title) {
        this.author = authorName;
        this.title = title;
    }
}

class SortBook {
    public static void main(String[] args) {
        List books = List.of(new Book("A1","T1"), new Book("A2", "T2"), new Book("A1","T2")); // Line n1
        books.sort((Book a, Book b) -> a.title.compareTo(b.title)); // Line n2
        System.out.println(books);
    }
}
```

Which action sorts the book list?

- A. At Line n2, replace books.sort() with books.stream().sort(0.
- B. At line n1, convert books type to mutable ArrayList type.
- C. At Line n1, convert type to mutable array type.
- D. At Line n2, replace compareTo () with compare ()).

Answer: D

Explanation:

The code fragment is trying to sort a list of books using the Collections.sort() method. The correct answer is D, because the compareTo() method is not the correct way to compare two objects in a Comparator. The compare() method is the correct way to compare two objects in a Comparator and return an int value that indicates their order¹. The compareTo() method is used to implement the Comparable interface, which defines the natural order of objects of a class². The other options are incorrect because they either do not change the type of the list, which is already mutable, or they do not use the correct syntax for sorting a stream, which requires a terminal operation such as collect()³. References: Comparator (Java SE 17 & JDK 17), Comparable (Java SE 17 & JDK 17), Stream (Java SE 17 & JDK 17)

NEW QUESTION 19

Given the code fragments:

```
class Car implements Serializable {
    private static long serialVersionUID = 454L;
    String name;
    public Car(String name) { this.name = name; }
}

class LuxuryCar extends Car {           // line n1
    int flag_HHC;
    public LuxuryCar(String name, int flag_HHC) {
        super(name);
        this.flag_HHC = flag_HHC;
    }
    public String toString() {
        return name + " : " + flag_HHC;
    }
}

and:
public static void main(String[] args) {    // line n2
    Car b = new LuxuryCar("Wagon", 200);
    try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("car.ser"));
        ObjectInputStream ois = new ObjectInputStream(new FileInputStream("car.ser"));) {
        oos.writeObject(b);
        System.out.println((Car)(ois.readObject()));           // line n3
    }
}
```

Which action prints Wagon : 200?

- A. At line n1, implement the java.io, Serializable interface.
- B. At line n3, replace readObject () with readLine().
- C. At Line n3, replace Car with LuxurayCar.
- D. At Line n1, implement the java.io.AutoCloseable interface
- E. At line n2, in the main method signature, add throws IOException, ClassCastException.
- F. At line n2, in the main method signature, add throws IOException, ClassNotFoundException.

Answer: F

Explanation:

The code fragment is trying to read an object from a file using the ObjectInputStream class. This class throws an IOException and a ClassNotFoundException. To handle these exceptions, the main method signature should declare that it throws these exceptions. Otherwise, the code will not compile. If the main method throws these exceptions, the code will print Wagon : 200, which is the result of calling the toString method of the LuxuryCar object that was written to the file. References: ObjectInputStream (Java SE 17 & JDK 17) - Oracle, ObjectOutputStream (Java SE 17 & JDK 17) - Oracle

NEW QUESTION 22

Given the code fragment:

```
List<String> specialDays = List.of("NewYear","Valentines","Spring","Labour");
System.out.print(specialDays.stream().allMatch(s ->s.equals("Labour")));
System.out.print(" " + specialDays.stream().anyMatch(s ->s.equals("Labour")));
System.out.print(" " + specialDays.stream().noneMatch(s -> s.equals("Halloween")));
System.out.print(" " +specialDays.stream().findFirst());
```

What is the result?

- A. False true true optional (Newyear)
- B. 0110
- C. True true false NewYear
- D. 010 optional (Newyear)

Answer: A

Explanation:

The code fragment is using the stream methods `allMatch`, `anyMatch`, `noneMatch`, and `findFirst` on a list of strings called `specialDays`. These methods are used to perform matching operations on the elements of a stream, such as checking if all, any, or none of the elements satisfy a given predicate, or finding the first element that matches a predicate¹. The predicate in this case is that the string equals `??Labour??` or `??Halloween??`. The output will be:

? False: because not all of the elements in `specialDays` are equal to `??Labour??` or `??Halloween??`.

? true: because at least one of the elements in `specialDays` is equal to `??Labour??` or `??Halloween??`.

? true: because none of the elements in `specialDays` are equal to both `??Labour??` and `??Halloween??`.

? Optional[NewYear]: because the first element in `specialDays` that matches the predicate is `??NewYear??`, and the `findFirst` method returns an Optional object that may or may not contain a non-null value².

References: Stream (Java SE 17 & JDK 17), Optional (Java SE 17 & JDK 17)

NEW QUESTION 25

.....

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](#)