Oracle

Exam Questions 1z0-829

Java SE 17 Developer

NEW QUESTION 1
Given:

```
public class Test {
    public void sum(int a, int b) {
        System.out.print(" A");
    }
    public void sum(int a, float b) {
        System.out.print(" B");
    }
    public void sum(float a, float b) {
        System.out.print(" C");
    }
    public void sum(double... a) {
        System.out.print(" D");
    }
    public static void main(String[] args) {
        Test t = new Test();
        t.sum(10,15.25);
        t.sum(10, 24);
        t.sum(10.25,10.25);
    }
}
```

What is the result?

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

Answer: C

Explanation:
The answer is C because the code demonstrates the concept of method overloading and type conversion in Java. Method overloading allows different methods to have the same name but different parameters. Type conversion allows values of one data type to be assigned to another data type, either automatically or explicitly. In the code, the class Test has four methods named sum, each with different parameter types: int, float, and double. The main method creates an instance of Test and calls the sum method with different arguments. The compiler will choose the most specific method that matches the arguments, based on the following rules:
? If there is an exact match between the argument types and the parameter types, that method is chosen.
? If there is no exact match, but there is a method with compatible parameter types, that method is chosen. Compatible types are those that can be converted from one to another automatically, such as int to long or float to double.
? If there is more than one method with compatible parameter types, the most specific method is chosen. The most specific method is the one whose parameter types are closest to the argument types in terms of size or precision.
In the code, the following method calls are made:
? test.sum(10, 10.5) -> This matches the sum(int a, float b) method exactly, so it is chosen. The result is 20.5, which is converted to int and printed as 20 (B).
? test.sum(10) -> This does not match any method exactly, but it matches the sum(double a) method with compatible types, as int can be converted to double automatically. The result is 10.0, which is printed as 10 (A).
? test.sum(10.5, 10) -> This does not match any method exactly, but it matches two methods with compatible types: sum(float a, float b) and sum(double a, double b). The latter is more specific, as double is closer to the argument types than float. The result is 20.5, which is printed as 20 (D).
Therefore, the output is B A D. References:
? Oracle Certified Professional: Java SE 17 Developer
? Java SE 17 Developer
? OCP Oracle Certified Professional Java SE 17 Developer Study Guide
? Method Overloading in Java
? Type conversion in Java with Examples
? Java Method Overloading with automatic type conversions

NEW QUESTION 2
Given:

```
public class Main {
    void print(int i){
        System.out.println("hello");
    }
    void print(long j){
        System.out.println("there");
    }

    public static void main(String[] args) {
        new Main().print(0b1101_1010);
    }
}
```

A. Hello
B. Compilation fails
C. A NumberFormatException is thrown
D. there

Answer: B

Explanation:
The code fragment will fail to compile because the parseInt method of the Integer class is a static method, which means that it can be invoked without creating an object of the class. However, the code is trying to invoke the parseInt method on an object of type Integer, which is not allowed. The correct way to invoke the parseInt method is by using the class name, such as Integer.parseInt (s). Therefore, the code fragment will produce a compilation error. References: Integer (Java SE 17 & JDK 17) - Oracle

NEW QUESTION 3
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 4
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 5
Given:

```
public class App {
    public int x = 100;

    public static void main(String[] args) {
        int x = 1000;
        App t = new App();
        t.myMethod(x);
        System.out.println(x);
    }
    public void myMethod(int x) {
        x++;
        System.out.println(x);
        System.out.println(this.x);
    }
}
```

What is the result?
A.
1001
1001
1000
B.
101
101
1000
C.

100
100
1000
D.
1001
100
1000

A.

Answer: D

Explanation:
The code fragment is using the bitwise operators & (AND), | (OR), and ^ (XOR) to perform operations on the binary representations of the integer values. The & operator returns a 1 in each bit position where both operands have a 1, the | operator returns a 1 in each bit position where either operand has a 1, and the ^ operator returns a 1 in each bit position where only one operand has a 1. The binary representations of the integer values are as follows:
? 1000 = 1111101000
? 100 = 1100100
? 101 = 1100101
The code fragment performs the following operations:
? x = x ^ y; // x becomes 1111010101, which is 1001 in decimal
? y = x ^ y; // y becomes 1100100, which is 100 in decimal
? x = x ^ y; // x becomes 1100101, which is 101 in decimal
The code fragment then prints out the values of x, y, and z, which are 1001, 100, and 1000 respectively. Therefore, option D is correct.

NEW QUESTION 6
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 7
Which two code fragments compile?
A)

```
class L6 {
    public static void main(String[] args) {
        var x = new ArrayList<>();
        x.add(10);
        x.add("30");
        System.out.println(x);
    }
}
```

B)

```
class L2 {
    public void m(int x) {
        var x = 10;
    }
}
```

C)

```
class A {}
class B extends A{}
class L4 {
    public static void main(String[] args) {
        var x = new A();
        x = new B();
    }
}
```

D)

```
class L3 {
    public static void main(String[] args) {
        var a = 10;
        a = "30";
    }
}
```

E)

```
class L5 {
    public void m() {
        var strVar = null;
    }
}
```

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

Answer: BE

Explanation:
The two code fragments that compile are B and E. These are the only ones that use the correct syntax for declaring and initializing a var variable. The var keyword is a reserved type name that allows the compiler to infer the type of the variable based on the initializer expression. However, the var variable must have an initializer, and the initializer must not be null or a lambda expression. Therefore, option A is invalid because it does not have an initializer, option C is invalid because it has a null initializer, and option D is invalid because it has a lambda expression as an initializer. Option B is valid because it has a String initializer, and option E is valid because it has an int initializer. https://docs.oracle.com/en/java/javase/17/language/local-variable-type-inference.html

NEW QUESTION 8
Given:

```
public class Test {
   public static void main(String[] args) {
       List<String>  elements =
             Arrays.asList("car", "truck", "car",
                           "bicycle", "car", "truck", "motorcycle");
       Map<String, Long>  outcome =
             elements.stream().collect(Collectors.groupingBy(Function.identity(), Collectors.counting() ) );
       System.out.println(outcome);
   }
}
```

What is the result?

A. Bicycle =7, car=7, motorcycle=7, truck=7)
B. (3:bicycle, 0:car, 0motercycle, 5:truck)
C. (Bicycle, car, motorcycle, truck)

D. Bicycle-1, car=3, motorcycle=1, truck=2)
E. Compilation fails.

Answer: E

Explanation:
The answer is E because the code fragment contains several syntax errors that prevent it from compiling. Some of the errors are:
? The enum declaration is missing a semicolon after the list of constants.
? The enum constants are not capitalized, which violates the Java naming convention for enums.
? The switch expression is missing parentheses around the variable name.
? The case labels are missing colons after the enum constants.
? The default label is missing a break statement, which causes a fall-through to the next case.
? The println statement is missing a closing parenthesis and a semicolon. A possible corrected version of the code fragment is:
enum Vehicle { BICYCLE, CAR, MOTORCYCLE, TRUCK; } public class Test { public static void main(String[] args) { Vehicle v = Vehicle.BICYCLE; switch (v) {
case BICYCLE:
System.out.print(??1??); break; case CAR: System.out.print(??3??); break; case MOTORCYCLE: System.out.print(??1??); break; case TRUCK:
System.out.print(??2??); break; default: System.out.print(??0??); break; } System.out.println(); } }
This would print 1 as the output. References:
? Oracle Certified Professional: Java SE 17 Developer
? Java SE 17 Developer
? OCP Oracle Certified Professional Java SE 17 Developer Study Guide
? Enum Types
? The switch Statement

NEW QUESTION 9
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). References: AbstractExecutorService (Java SE 17 & JDK 17) - Oracle

NEW QUESTION 10
Given:

```
public class Test {
    static interface Animal {
    }

    static class Dog implements Animal {
    }

    private static void play(Animal a) {
        System.out.print("flips");
    }

    private static void play(Dog d) {
        System.out.print("runs");
    }

    public static void main(String[] args) {
        Animal a1 = new Dog();
        Dog a2 = new Dog();
        play(a1);
        play(a2);
    }
}
```

What is the result?

A. flipsflips
B. Compilation fails
C. flipsruns
D. runsflips
E. runsruns

Answer: B

Explanation:
The code fragment will fail to compile because the play method in the Dog class is declared as private, which means that it cannot be accessed from outside the class. The main method is trying to call the play method on a Dog object, which is not allowed. Therefore, the code fragment will produce a compilation error.

NEW QUESTION 10
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 13
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 15
Given:

```
class A {public void mA() {System.out.println("mA");}}
class B extends A {public void mB() {System.out.println("mB");}}
class C extends B {public void mC() {System.out.println("mC");}}

public class App {
    public static void main(String[] args) {
        A bobj = new B();
        A cobj = new C();
        if (cobj instanceof B v) {
            v.mB();
            if (v instanceof C v1) { v1.mC(); }
        } else {
            cobj.mA();
        }
    }
}
```

What is the result?

A. Mb MC
B. Mb
C. Mb
D. MA
E. mA

Answer: E

Explanation:
The code snippet is an example of Java SE 17 code. The code is checking if the object is an instance of class C and if it is, it will print ??mC??. If it is not an instance of class C, it will print ??mA??. In this case, the object is not an instance of class C, so the output
will be ??mA??. References: Pattern Matching for instanceof - Oracle Help Center

NEW QUESTION 17
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. Interace story extends STnt {} Interface Art extends SInt {}
B. Public interface story extends sInd {} Public interface Art extends SInt {}
C. Sealed interface Storty 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 sInd 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 20
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 25
Given:

```
class StockException extends Exception {
    public StockException(String s) { super(s); }
}
class OutofStockException extends StockException {
    public OutofStockException(String s) { super(s); }
}
```

and the code fragment:

```
public class Test {
    public static void main(String[] args) throws OutofStockException {
        m();
    }
    public static void m() throws OutofStockException {
        try {
            throw new StockException("Raised.");
        } catch (Exception e) {
            throw new OutofStockException(e.getMessage());
        }
    }
}
```

Which statement is true?

A. The program throws StockException.
B. The program fails to compile.
C. The program throws outofStockException.
D. The program throws ClassCastException

Answer: B

Explanation:
The answer is B because the code fragment contains a syntax error that prevents it from compiling. The code fragment tries to catch a StockException in line 10, but the catch block does not have a parameter of type StockException. The catch block should have a parameter of type StockException, such as:
catch (StockException e) { // handle the exception }
This is required by the Java syntax for the catch clause, which must have a parameter that is a subclass of Throwable. Without a parameter, the catch block is invalid and causes a compilation error.
Option A is incorrect because the program does not throw a StockException, as it does not compile.
Option C is incorrect because the program does not throw an OutofStockException, as it does not compile.
Option D is incorrect because the program does not throw a ClassCastException, as it does not compile. References:
? Oracle Certified Professional: Java SE 17 Developer
? Java SE 17 Developer
? OCP Oracle Certified Professional Java SE 17 Developer Study Guide
? The try-with-resources Statement (The Java™ Tutorials > Essential Classes > Exceptions)
? The catch Blocks (The Java™ Tutorials > Essential Classes > Exceptions)

NEW QUESTION 28
Given the code fragment:

```
record Product(int pNumber, String pName) {
    int regNo = 100;
    public int getRegNumber() {
        return regNo;
    }
}

public class App {
    public static void main(String[] args) {
        Product p1 = new Product (1111, "Ink Bottle");
    }
}
```

Which action enables the code to compile?

A. Replace record with void.
B. Remove the regNO initialization statement.
C. Make the regNo variable static.
D. Replace thye regNo variable static
E. Make the regNo variable public

Answer: E

Explanation:
The code will compile if the regNo variable is made public. This is because the regNo variable is being accessed in the main method of the App class, which is outside the scope of the Product class. Making the regNo variable public will allow it to be accessed from outside the class. References: https://education.oracle.com/products/trackp_OCPJSE17, https://mylearn.oracle.com/ou/learning-path/java-se-17-developer/99487, https://docs.oracle.com/javase/tutorial/java/javaOO/accesscontrol.html

NEW QUESTION 32
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 37
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: References: AtomicInteger (Java SE 17 & JDK 17) - Oracle

NEW QUESTION 38
.....

Thank You for Trying Our Product

We offer two products:

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

2nd - Questons 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