Exam Questions 1z0-829

Java SE 17 Developer

https://www.2passeasy.com/dumps/1z0-829/

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 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 3
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 4
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 5
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 6
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 7
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: B

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. References: Oracle Java SE 17 Developer source and documents: [String (Java SE 17 & JDK 17)], [StringBuilder (Java SE 17 & JDK 17)]

NEW QUESTION 8
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 9
Which statement is true about migration?

A. Every module is moved to the module path in a top-down migration.
B. Every module is moved to the module path in a bottom-up migration.
C. The required modules migrate before the modules that depend on them in a top-down^migration.
D. Unnamed modules are automatic modules in a top-down migration.

Answer: B

Explanation:
The answer is B because a bottom-up migration is a strategy for modularizing an existing application by moving its dependencies to the module path one by one, starting from the lowest-level libraries and ending with the application itself. This way, each module can declare its dependencies on other modules using the module-info.java file, and benefit from the features of the Java Platform Module System (JPMS), such as reliable configuration, strong encapsulation, and service loading.
Option A is incorrect because a top-down migration is a strategy for modularizing an existing application by moving it to the module path first, along with its dependencies as automatic modules. Automatic modules are non-modular JAR files that are treated as modules with some limitations, such as not having a module descriptor or a fixed name. A top-down migration allows the application to use the module path without requiring all of its dependencies to be modularized first.
Option C is incorrect because a top-down migration does not require any specific order of migrating modules, as long as the application is moved first and its dependencies are moved as automatic modules. A bottom-up migration, on the other hand, requires the required modules to migrate before the modules that depend on them.
Option D is incorrect because unnamed modules are not automatic modules in any migration strategy. Unnamed modules are modules that do not have a name or a module descriptor, such as classes loaded from the class path or dynamically generated classes. Unnamed modules have unrestricted access to all other modules, but they cannot be accessed by named modules, except through reflection with reduced security checks. References:
? Oracle Certified Professional: Java SE 17 Developer
? Java SE 17 Developer
? OCP Oracle Certified Professional Java SE 17 Developer Study Guide
? Migrating to Modules (How and When) - JavaDeploy
? Java 9 Modularity: Patterns and Practices for Developing Maintainable
Applications

NEW QUESTION 10
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 10
Given:

```
1. class Item {
2.    String name;
3.    public static void display() {
4.       name = "Vase";
5.       System.out.println(name);
6.    }
7.    public void display(String design) {
8.       this.name += name;
9.       System.out.println(name);
10.   }
11. }
12. public class App {
13.    public static void main(String[] args) {
14.       Item i1 = new Item();
15.       i1.display("Flower");
16.    }
17. }
```

Which action enables the code to compile?

A. Replace 15 with item.display ("Flower");
B. Replace 2 with static string name;
C. Replace 7 with public void display (string design) {
D. Replace 3 with private static void display () {

Answer: C

Explanation:
The answer is C because the code fragment contains a syntax error in line 7, where the method display is declared without any parameter type. This causes a compilation error, as Java requires the parameter type to be specified for each method parameter. To fix this error, the parameter type should be added before the parameter name, such as string design. This will enable the code to compile and run without any errors. References:
? Oracle Certified Professional: Java SE 17 Developer
? Java SE 17 Developer
? OCP Oracle Certified Professional Java SE 17 Developer Study Guide
? Java Methods

NEW QUESTION 14
Given the code fragment:

```
// line n1
String input = console.readLine("Input a number: ");
int number = Integer.parseInt(input);

if (number % 2 == 0) {
    System.out.println(number + " is even.");
} else {
    System.out.println(number + " is odd");
}
```

Which code line n1, obtains the java.io.Console object?

A)

```
Console console = Syst      ole(Syst      .in);
                   em.cons          em
```

B)

```
Console console = Cons      Intance( );
                     ole.get
```

C)

```
Console console = Syst      ole();
                   em.cons
```

D)

```
Console console = new Console(System.in);
```

E)

```
Console console = new      (new Inp
                  Console      utStreamReader(System.in));
```

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

Answer: A

Explanation:
The code fragment is trying to obtain the java.io.Console object, which is a class that provides methods to access the character-based console device, if any, associated with the current Java virtual machine. The correct way to obtain the Console object is to call the static method Console console() in the java.lang.System class. This method returns the unique Console object associated with the current Java virtual machine, if any. Therefore, option A is correct, as it calls System.console() and assigns it to a Console variable. References:
? https://docs.oracle.com/javase/17/docs/api/java.base/java/io/Console.html
? https://docs.oracle.com/javase/17/docs/api/java.base/java/lang/System.html#cons ole()
? https://education.oracle.com/products/trackp_OCPJSE17
? https://mylearn.oracle.com/ou/learning-path/java-se-17-developer/99487

NEW QUESTION 17
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 (0 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:

Captions.properties file:

```
user = UserName
```

Captions_en.properties file:

```
user = User name (EN)
```

Captions_US.properties file:

```
message = User name (US)
```

Captions_en_US.properties file:

```
message = User name (EN - US)
```

and the code fragment:

```
Locale.setDefault(Locale.US);
Locale currentLocale = new Locale.Builder().setLanguage("en").build();
ResourceBundle captions = ResourceBundle.getBundle("Captions.properties", currentLocale);
System.out.println(captions.getString("user"));
```

What is the result?

A. User name (US)
B. The program throws a MissingResourceException.
C. User name (EN – US)
D. UserName
E. User name (EN)

Answer: B

Explanation:
The answer is B because the code fragment contains a logical error that causes a MissingResourceException at runtime. The code fragment tries to load a resource bundle with the base name ??Captions.properties?? and the locale ??en_US??. However, there is no such resource bundle available in the classpath. The available resource bundles are:
? Captions.properties
? Captions_en.properties
? Captions_US.properties
? Captions_en_US.properties
The ResourceBundle class follows a fallback mechanism to find the best matching resource bundle for a given locale. It first tries to find the resource bundle with the exact locale, then it tries to find the resource bundle with the same language and script, then it tries to find the resource bundle with the same language, and finally it tries to find the default resource bundle with no locale. If none of these resource bundles are found, it throws a MissingResourceException.
In this case, the code fragment is looking for a resource bundle with the base name ??Captions.properties?? and the locale ??en_US??. The ResourceBundle class will try to find the following resource bundles in order:
? Captions.properties_en_US
? Captions.properties_en
? Captions.properties
However, none of these resource bundles exist in the classpath. Therefore, the ResourceBundle class will throw a MissingResourceException.
To fix this error, the code fragment should use the correct base name of the resource bundle family, which is ??Captions?? without the ??.properties?? extension. For example: ResourceBundle captions = ResourceBundle.getBundle(??Captions??, currentLocale); This will load the appropriate resource bundle for the current locale, which is ??Captions_en_US.properties?? in this case. References:
? Oracle Certified Professional: Java SE 17 Developer
? Java SE 17 Developer
? OCP Oracle Certified Professional Java SE 17 Developer Study Guide
? ResourceBundle (Java Platform SE 8)
? About the ResourceBundle Class (The Java™ Tutorials > Internationalization)

NEW QUESTION 23
Given the code fragment:

```
Integer rank = 4;
switch (rank) {
     case 1,4 -> System.out.println("Range1");
     case 5,8 -> System.out.println("Range2");
     case 9,10 -> System.out.println("Range3");
     default -> System.out.println("Not a valid rank.");
}
```

What is the result?

A. Range 1Range 2Range 3
B. Range1Note a valid rank.
C. Range 1Range 2Range 3Range 1Not a valida rank
D. Range 1

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 rank and executes the corresponding case statement. In this case, the value of rank is 4, so the first case statement is executed, printing ??Range1??. The second and third case statements are also executed, printing ??Range2?? and ??Range3??. The default case statement is also executed, printing ??Not a valid rank??. References: Java Language Changes - Oracle Help Center

NEW QUESTION 26
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 28

Given:

```
public class Weather {
      public enum Forecast {
            SUNNY, CLOUDY, RAINY;
            @Override
            public String toString() { return "SNOWY";}
      }

      public static void main(String[] args) {
            System.out.print(Forecast.SUNNY.ordinal() + " ");
            System.out.print(Forecast.valueOf("cloudy".toUpperCase()));
      }
}
```

What is the result?

A. 1 RAINY
B. Compilation fails
C. 1 Snowy
D. 0 CLOUDY
E. 0 Snowy

Answer: E

Explanation:
The code is defining an enum class called Forecast with three values: SUNNY, CLOUDY, and RAINY. The toString() method is overridden to always return ??SNOWY??. In the main method, the ordinal value of SUNNY is printed, which is 0, followed by the value of CLOUDY converted to uppercase, which is ??CLOUDY??. However, since the
toString() method of Forecast returns ??SNOWY?? regardless of the actual value, the output
will be ??0 SNOWY??. References: Enum (Java SE 17 & JDK 17), Enum.EnumDesc (Java SE 17 & JDK 17)

NEW QUESTION 33
Given the code fragment:

```
// Login time:2021-01-12T21:58:18.817Z
Instant loginTime = Instant.now();
Thread.sleep(1000);

// Logout time:2021-01-12T21:58:19.880Z
Instant logoutTime = Instant.now();

loginTime = loginTime.truncatedTo(ChronoUnit.MINUTES);   // line n1
logoutTime = logoutTime.truncatedTo(ChronoUnit.MINUTES);

if (logoutTime.isAfter(loginTime))
    System.out.println("Logged out at: " + logoutTime);
else
    System.out.println("Can't logout");
```

What is the result?

A. Logged out at: 2021-0112T21:58:19.880z
B. Logged out at: 2021-01-12T21:58:00z
C. A compilation error occurs at Line n1.
D. Can??t logout

Answer: B

Explanation:
The code fragment is using the Java SE 17 API to get the current time and then truncating it to minutes. The result will be the current time truncated to minutes, which is why option B is correct. 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/17/docs/api/java.base/java/time/Instant.html#truncatedTo(java.time.temporal.TemporalUnit)

NEW QUESTION 36
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 41
Given the directory structure:

```
module1:
        p1\
           Doc.java
        p2\
           Util.java
```

Given the definition of the Doc class:

```
package p1;
    public sealed class Doc permits WordDoc {
}
```

Which two are valid definition of the wordDoc class?

A. Package p1;Public non-sealed class wordDoc extends Doc ()
B. Package p1;Public class wordDoc extends Doc ()
C. Package p1, p2;Public non-sealed class WordDoc extends Doc ()
D. Package p1, p2;Public sealed class WordDoc extends Doc ()
E. Package p1,non-sealed abstract class WordDoc extends Doc ()
F. Package p1;Public final class WordDoc extends Doc ()

Answer: AF

Explanation:
The correct answer is A and F because the wordDoc class must be a non-sealed class or a final class to extend the sealed Doc class. Option B is incorrect because the wordDoc class must be non-sealed or final. Option C is incorrect because the wordDoc class cannot be in a different package than the Doc class. Option D is incorrect because the wordDoc class cannot be a sealed class. Option E is incorrect because the wordDoc class cannot be an abstract class.
References: Oracle Certified Professional: Java SE 17 Developer, 3 Sealed Classes - Oracle Help Center

NEW QUESTION 43
.....

THANKS FOR TRYING THE DEMO OF OUR PRODUCT

Visit Our Site to Purchase the Full Set of Actual 1z0-829 Exam Questions With Answers.

We Also Provide Practice Exam Software That Simulates Real Exam Environment And Has Many Self-Assessment Features. Order the 1z0-829 Product From:

https://www.2passeasy.com/dumps/1z0-829/

Money Back Guarantee

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