

## 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:

```
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:

```
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, 0:motorcycle, 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 4

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 5

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 6**

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. ABBBCDE // 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 7**

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 8**

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 9

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 slnd {} 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 slnd 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 the code fragment:

```
List<Integer> listOfNumbers = List.of(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);
```

Which code fragment returns different values?

- A. int sum = listOfNumber
- B. parallelStream () reduce (5, Integer:: sum) ;
- C. int sum = listOfNumber
- D. Stream () reduce (5, (a, b) -> a + b) ;
- E. int sum = listOfNumber
- F. Stream () reduce ( Integer:: sum) ; +5;
- G. int sum = listOfNumber
- H. parallelStream () reduce ({m, n} -> m +n) orElse (5) +5;
- I. int sum = listOfNumber
- J. Stream () reduce (0, Integer:: sum) + 5

**Answer: C**

#### Explanation:

The answer is C because the code fragment uses a different syntax and logic for the reduce operation than the other options. The reduce method in option C takes a single parameter, which is a BinaryOperator that combines two elements of the stream into one. The method returns an Optional, which may or may not contain a value depending on whether the stream is empty or not. The code fragment then adds 5 to the result of the reduce method, regardless of whether it is present or not. This may cause an exception if the Optional is empty, or produce a different value than the other options if the Optional is not empty.

The other options use a different syntax and logic for the reduce operation. They all take two parameters, which are an identity value and a BinaryOperator that combines an element of the stream with an accumulator. The method returns the final accumulator value, which is equal to the identity value if the stream is empty, or the result of applying the BinaryOperator to all elements of the stream otherwise. The code fragments then add 5 to the result of the reduce method, which will always produce a valid value.

For example, suppose listOfNumbers contains [1, 2, 3]. Then, option A will perform the following steps:

- ? Initialize accumulator to identity value 5
- ? Apply BinaryOperator Integer::sum to accumulator and first element: 5 + 1 = 6
- ? Update accumulator to 6
- ? Apply BinaryOperator Integer::sum to accumulator and second element: 6 + 2 = 8
- ? Update accumulator to 8
- ? Apply BinaryOperator Integer::sum to accumulator and third element: 8 + 3 = 11
- ? Update accumulator to 11
- ? Return final accumulator value 11



? Add 5 to final accumulator value:  $11 + 5 = 16$

Option B will perform the same steps as option A, except using a lambda expression instead of a method reference for the BinaryOperator. Option D will perform the same steps as option A, except using parallelStream instead of stream, which may change the order of applying the BinaryOperator but not the final result.

Option E will perform the same steps as option A, except using identity value 0 instead of 5.

Option C, however, will perform the following steps:

? Apply BinaryOperator Integer::sum to first and second element:  $1 + 2 = 3$

? Apply BinaryOperator Integer::sum to previous result and third element:  $3 + 3 = 6$

? Return Optional containing final result value 6

? Add 5 to Optional value:  $\text{Optional.of}(6) + 5 = \text{Optional.of}(11)$

As you can see, option C produces a different value than the other options, and also uses a different syntax and logic for the reduce operation. References:

? Oracle Certified Professional: Java SE 17 Developer

? Java SE 17 Developer

? OCP Oracle Certified Professional Java SE 17 Developer Study Guide

? Guide to Stream.reduce()

#### 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 13

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 17.1. 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 16

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 20

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 order<sup>1</sup>. The compareTo() method is used to implement the Comparable interface, which defines the natural order of objects of a class<sup>2</sup>. 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()<sup>3</sup>. References: Comparator (Java SE 17 & JDK 17), Comparable (Java SE 17 & JDK 17), Stream (Java SE 17 & JDK 17)

#### NEW QUESTION 23

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 26**

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 29**

Given:



```
public class Test {  
    public static void main(String[] args) {  
        final int x = 2;  
        int y = x;  
        while (y<3) {  
            switch (y) {  
                case 0+x:  
                    y++;  
                case 1:  
                    y++;  
            }  
        }  
        System.out.println(y);  
    }  
}
```

What is the result?

- A. 4
- B. 2
- C. 6
- D. Nothing is printed because of an indefinite loop.
- E. Compilation fails.
- F. 5
- G. A runtime exception is thrown.
- H. 3

**Answer:** E

**Explanation:**

The code will not compile because the variable `x` is declared as final and then it is being modified in the switch statement. This is not allowed in Java. A final variable is a variable whose value cannot be changed once it is initialized<sup>1</sup>. The switch statement tries to assign different values to `x` depending on the value of `y`, which violates the final modifier. The compiler will report an error: The final local variable x cannot be assigned. It must be blank and not using a compound assignment. References: The final Keyword (The Java™ Tutorials > Learning the Java Language > Classes and Objects)

**NEW QUESTION 32**

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://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 37**

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  
? Zoneld (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 39

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://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\(java.time.temporal.TemporalUnit\)](https://docs.oracle.com/javase/17/docs/api/java.base/java.time/Instant.html#truncatedTo(java.time.temporal.TemporalUnit))

#### NEW QUESTION 42

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 47**

.....



## 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