

# Oracle

## Exam Questions 1z0-829

Java SE 17 Developer



**NEW QUESTION 1**

Given:

```
import java.io.Serializable;
public class Software implements Serializable {
    private String title;
    public Software(String title) {
        this.title = title;
        System.out.print("Software ");
    }
    public String toString() { return title; }
}

public class Game extends Software {
    private int players;
    public Game(String title, int players) {
        super(title);
        this.players = players;
        System.out.print("Game ");
    }
    public String toString() { return super.toString()+" "+players; }
}

import java.io.*;
public class AppStore {
    public static void main(String[] args) {
        Software s = new Game("Chess", 2);
        try(ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream("game.ser"))) {
            out.writeObject(s);
        } catch (Exception e) {
            System.out.println("write error");
        }
        try(ObjectInputStream in = new ObjectInputStream(new FileInputStream("game.ser"))) {
            s = (Software)in.readObject();
        } catch (Exception e) {
            System.out.println("read error");
        }
        System.out.println(s);
    }
}
```

What is the result?

- A. Software Game Chess 0
- B. Software Game Software Game Chess 2
- C. Software game write error
- D. Software Game Software Game chess 0
- E. Software Game Chess 2
- F. Software Game read error

**Answer:** B**Explanation:**

The answer is B because the code uses the writeObject and readObject methods of the ObjectOutputStream and ObjectInputStream classes to serialize and deserialize the Game object. These methods use the default serialization mechanism, which writes and reads the state of the object's fields, including the inherited ones. Therefore, the title field of the Software class is also serialized and deserialized along with the players field of the Game class. The toString method of the Game class calls the toString method of the Software class using super.toString(), which returns the value of the title field. Hence, when the deserialized object is printed, it shows Software Game Software Game Chess 2.

References:

- ? Oracle Certified Professional: Java SE 17 Developer
- ? Java SE 17 Developer
- ? OCP Oracle Certified Professional Java SE 17 Developer Study Guide
- ? Serialization and Deserialization in Java with Example

**NEW QUESTION 2**

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 3**

Given:

```
interface IFace {
    public void m1();
    public default void m2() {
        System.out.println("m2");
    }
    public static void m3() {
        System.out.println("m3");
    }
    private void m4() {
        System.out.println("m4");
    }
}

class MyC implements IFace {
    public void m1() {
        System.out.println("Hello");
    }
}
```

Which two method invocation execute?

- A. IFace myclassobj = new Myc (); myclassObj.m3 ();
- B. Ifnce.m3 ();
- C. iFace mucloassObj = new Myc (); myClassObj.m4();
- D. new MyC() .m2 ();
- E. IFace .,4():
- F. IFace.m2();

**Answer:** DE

**Explanation:**

The code given is an interface and a class that implements the interface. The interface has three methods, m1(), m2(), and m3(). The class has one method, m1(). The only two method invocations that will execute are D and E. D is a call to the m2() method in the class, and E is a call to the m3() method in the interface.

References: [https://education.oracle.com/products/trackp\\_OCPJSE17](https://education.oracle.com/products/trackp_OCPJSE17), 3, 4, 5

**NEW QUESTION 4**

Given the product class:

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

And the shop class:

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

What is the result?

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

**Answer:** E

**Explanation:**

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

**NEW QUESTION 5**

Given the code fragment:

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

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

Which set of class definitions compiles?

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

**Answer:** C



**Explanation:**

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

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

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

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

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

References:

? Oracle Certified Professional: Java SE 17 Developer

? Java SE 17 Developer

? OCP Oracle Certified Professional Java SE 17 Developer Study Guide

? Sealed Classes and Interfaces in Java 15 | Baeldung

? Sealed Class in Java - Javatpoint

**NEW QUESTION 6**

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 7

Given:

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

public interface Car {
    int getSpeed();
}

and

package com.transport.vehicle.cars.impl;

import com.transport.vehicle.cars.Car;

public class CarImpl implements Car {
    private int speed;

    public CarImpl() {
        this(10);
    }

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

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

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

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

**Answer:** BE**Explanation:**

The answer is B and E because the module-info file should include a

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

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

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

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

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

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

? Oracle Certified Professional: Java SE 17 Developer

? Java SE 17 Developer

? OCP Oracle Certified Professional Java SE 17 Developer Study Guide

? Java Modules - Service Interface Module - GeeksforGeeks

? Java Service Provider Interface | Baeldung

### NEW QUESTION 8

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 9

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 10**

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 10**

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

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

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

What is the output?

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

**Answer: A**

**Explanation:**

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

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

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

Hence, the output is true false. References:

- ? Oracle Certified Professional: Java SE 17 Developer
- ? Java SE 17 Developer
- ? OCP Oracle Certified Professional Java SE 17 Developer Study Guide
- ? ZoneId (Java Platform SE 8 )
- ? ZonedDateTime (Java Platform SE 8 )
- ? Time Zone & Clock Changes in Chicago, Illinois, USA
- ? Daylight Saving Time Changes 2023 in Chicago, USA

**NEW QUESTION 13**

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 predicate<sup>1</sup>. 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 value<sup>2</sup>.

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

**NEW QUESTION 15**

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 20**

.....



## Thank You for Trying Our Product

### We offer two products:

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

2nd - Questions and Answers in PDF Format

### 1z0-829 Practice Exam Features:

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

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