Download Oracle 1Z0-829 Exam Dumps Free

Shared by Hayes on 17-06-2026

For More Free Questions and Preparation Resources

Check the Links on Last Page

Question 1

Question Type: MultipleChoice

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 ");
}
```

Options:

A- Option A
B- Option B
C- Option C
D- Option D
E- Option E

Answer:

B, E

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''.

Question 2

Question Type: MultipleChoice

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?

Options:

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 initialized1. 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.Reference:The final Keyword (The Java Tutorials > Learning the Java Language > Classes and Objects)

Question 3

Question Type: MultipleChoice

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?

Options:

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.Reference:Object Serialization - Oracle, [ObjectInputStream (Java SE 17 & JDK 17) - Oracle]

Question 4

Question Type: MultipleChoice

Given:

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

and the code fragment:

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

What is the result

Options:

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

Answer:

C

Explanation:

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

Undefined

Question 5

Question Type: MultipleChoice

Which statement is true?

Options:

A- The tryLock () method returns a boolean indicator immediately regardless if it has or has not managed to acquire the lock.
B- The tryLock () method returns a boolean indicator immediately if it has managed to acquire the lock, otherwise it waits for the lock acquisition.
C- The lock () method returns a boolean indicator immediately if it has managed to acquire the lock, otherwise it waits for the lock acquisition.
D- The Lock () method returns a boolean indicator immediately regardless if it has or has not managed to acquire the lock

Answer:

A

Explanation:

The tryLock () method of the Lock interface is a non-blocking attempt to acquire a lock. It returns true if the lock is available and acquired by the current thread, and false otherwise. It does not wait for the lock to be released by another thread. This is different from the lock () method, which blocks the current thread until the lock is acquired, and does not return any value.Reference: https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/concurrent/locks/Lock.html#tryLock(),3,4,5

Question 6

Question Type: MultipleChoice

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?

Options:

A- Requires cm.transport.vehicle,cars:

B- Provides.com.transport.vehicle.cars.Car with com.transport.vehicle.cars. impt, CatImpl;

C- Non-sealed interface story extends SInt {}

Class Art implements Sint {}

D- Provides.com.transport.vehicle.cars.Car impl,CarImp1 to com.transport.vehicle.cars. Cars

E- exports com.transport.vehicle.cars.Car;

F- Exports com.transport.vehicle.cars;

G- Exports com.transport.vehicle;

Answer:

B, E

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.Reference:

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

Question 7

Question Type: MultipleChoice

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;
    }
}
```

Options:

A- Option A
B- Option B
C- Option C
D- Option D
E- Option E

Answer:

B, E

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

Question 8

Question Type: MultipleChoice

Given the code fragment:

```
String s = "10_00";
Integer s2 = 10_00;
// Line n1
System.out.println(res);
```

Which two statements at Line nl independently enable you to print 1250?

Options:

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:

A, E

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 171. 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 value23. 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.Reference:Binary Literals (The Java Tutorials > Learning the Java Language > Numbers and Strings),Integer (Java SE 17 & JDK 17),Integer (Java SE 17 & JDK 17)