Oracle

Exam Questions 1z0-829

Java SE 17 Developer

NEW QUESTION 1
Given:

```
class Product {
    String name; double price;
    Product(String s, double d) {
        this.name = s;
        this.price = d;
    }
}
class ElectricProduct extends Product {
    ElectricProduct(String name, double price) {
        super(name, price);
    }
}
```

and the code fragment:

```
List<Product> p = List.of(
new ElectricProduct("CellPhone",100),
new ElectricProduct("ToyCar",90),
new ElectricProduct("Motor",200),
new ElectricProduct("Fan",300)
);

DoubleSummaryStatistics sts = p.stream().filter(a -> a instanceof ElectricProduct)
                                        .collect(Collectors.summarizingDouble(a ->
a.price));
String s1 = p.stream().filter(a -> a instanceof Product)
                      .collect(Collectors.mapping(p2 -> p2.name, Collectors.joining(",")));
System.out.println(sts.getMax());
System.out.println(s1);
```

A. 300.00CellPhone,ToyCar,Motor,Fan
B. 100.00CellPhone,ToyCar,Motor,Fan
C. 100.00 CellPhone,ToyCar
D. 300.00CellPhone.ToyCar

Answer: A

Explanation:
The code fragment is using the Stream API to perform a reduction operation on a list of ElectricProduct objects. The reduction operation consists of three parts: an identity value, an accumulator function, and a combiner function. The identity value is the initial value of the result, which is 0.0 in this case. The accumulator function is a BiFunction that takes two arguments: the current result and the current element of the stream, and returns a new result. In this case, the accumulator function is (a,b) -> a + b.getPrice (), which means that it adds the price of each element to the current result. The combiner function is a BinaryOperator that takes two partial results and combines them into one. In this case, the combiner function is (a,b) -> a + b, which means that it adds the two partial results together.
The code fragment then applies a filter operation on the stream, which returns a new stream that contains only the elements that match the given predicate. The predicate is p -
> p.getPrice () > 10, which means that it selects only the elements that have a price greater than 10. The code fragment then applies a map operation on the filtered stream, which returns a new stream that contains the results of applying the given function to each element. The function is p -> p.getName (), which means that it returns the name of each element.
The code fragment then calls the collect method on the mapped stream, which performs a mutable reduction operation on the elements of the stream using a Collector. The Collector is Collectors.joining (??,??), which means that it concatenates the elements of the stream into a single String, separated by commas.
The code fragment then prints out the result of the reduction operation and the result of the collect operation, separated by a new line. The result of the reduction operation is 300.00, which is the sum of the prices of all ElectricProduct objects that have a price greater than 10. The result of the collect operation is CellPhone,ToyCar,Motor,Fan, which is the concatenation of the names of all ElectricProduct objects that have a price greater than 10. Therefore, the output of the code fragment is: 300.00 CellPhone,ToyCar,Motor,Fan
References: Stream (Java SE 17 & JDK 17) - Oracle, Collectors (Java SE 17 & JDK 17) - Oracle

NEW QUESTION 2
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 3
Given the code fragment:

```
ExecutorService executorService = Executors.newSingleThreadExecutor();
Set<Callable<String>> workers = new HashSet<Callable<String>>();
workers.add(new Callable<String>() {
    public String call() throws Exception {
        return "1";
    }
});
workers.add(new Callable<String>() {
    public String call() throws Exception {
        return "2";
    }
});
workers.add(new Callable<String>() {
    public String call() throws Exception {
        return "3";
    }
});
```

Which code fragment invokes all callable objects in the workers set?

A)

```
List<Future<String>> futures = executorService.invokeAny(workers);
for(Future<String> future : futures){
    System.out.println(future.get());
}
```

B)

```
executorService.submit(cThreads);
```

C)

```
List<Future<String>> futures = executorService.invokeAll(workers);
for(Future<String> future : futures){
    System.out.println(future.get());
}
```

D)

```
for (int i=0; i<3;i++) {
    String result = executorService.invokeAny(cThreads);
    System.out.println(result);
}
```

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

Answer: C

Explanation:
The code fragment in Option C invokes all callable objects in the workers set by using the ExecutorService??s invokeAll() method. This method takes a collection of Callable objects and returns a list of Future objects representing the results of the tasks. The other options are incorrect because they either use the wrong method (invokeAny() or submit()) or have syntax errors (missing parentheses or
semicolons). References: AbstractExecutorService (Java SE 17 & JDK 17) - Oracle

NEW QUESTION 4
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, 3, 4, 5

NEW QUESTION 5
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 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 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 9
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 10
Given the code fragment:

```
int a = 2;
int b = ~a;
int c = a^b;
boolean d = a < b & a > c++;
System.out.println(d + " " + c);
boolean e = a > b && a > c++;
System.out.println(e + " " + c);
```

What is the result?

A. false 1false 2
B. true 1false 2
C. false 1ture 2
D. falase 0true 1

Answer: B

Explanation:
The code fragment is comparing the values of a, b, and c using the < and > operators. The first comparison, d, is checking if a is less than b and greater than c.

Since a is equal to 2, b is equal to -2, and c is equal to -4, this comparison will evaluate to true. The second comparison, e, is checking if a is greater than b and a is greater than c. Since a is equal to 2, b is equal to -2, and c is equal to -4, this comparison will evaluate to false.
Therefore, the result will be true 1 false 2. References: Operators (The Java™ Tutorials > Learning the Java Language - Oracle

NEW QUESTION 10
Given the course table:

COURSE_ID	COURSE_NAME	COURSE_FEE	COURSE_LEVEL
1021	Java Programmer	400.00	1
1022	Java Architect	600.00	2
1023	Java Master	600.00	2

Given the code fragment:

```
try (Connection con = DriverManager.getConnection(connectionString)) {
    Statement statement = con.createStatement(TYPE_SCROLL_INSENSITIVE,ResultSet.CONCUR_UPDATABLE);
    String qry = "UPDATE course SET course_fee = ? where COURSE_LEVEL = ?";
    PreparedStatement prStmt = con.prepareStatement(qry, TYPE_SCROLL_INSENSITIVE);
    prStmt.setDouble(1,600.00);
    prStmt.setInt(2,2);
    System.out.println(prStmt.executeUpdate());
}
catch(SQLException sqlException) {
    System.out.println(sqlException);
}
```

A. 2
B. false
C. true
D. 1

Answer: C

Explanation:
The code fragment will execute the update statement and set the course fee of the course with ID 1021 to 5000. The executeUpdate method returns an int value that indicates the number of rows affected by the SQL statement. In this case, only one row will be updated, so the result variable will be 1. The if statement will check if the result is greater than 0, which is true, and print ??Updated successfully??. Therefore, the output of the code fragment is true. References: https://education.oracle.com/products/trackp_OCPJSE17, https://mylearn.oracle.com/ou/learning-path/java-se-17-developer/99487, https://docs.oracle.com/en/java/javase/17/docs/api/java.sql/java/sql/Statement.html#execut eUpdate(java.lang.String)

NEW QUESTION 15
Given the content of the in. tart file: 23456789
and the code fragment:

```
char[] buffer = new char[8];
int count = 0;
try(FileReader in = new FileReader("in.txt");
    FileWriter out = new FileWriter("out.txt")) {
        while((count = in.read(buffer)) != -1) {
            out.write(buffer);
        }
}
```

What is the content of the out .txt file?

A. 01234567801234
B. 012345678
C. 0123456789234567
D. 0123456789
E. 012345678901234
F. 01234567

Answer: D

Explanation:
The answer is D because the code fragment reads the content of the in.txt file and writes it to the out.txt file. The content of the in.txt file is ??23456789??. The code fragment uses a char array buffer of size 8 to read the content of the in.txt file. The while loop reads the content of the in.txt file and writes it to the out.txt file until the end of the file is reached. Therefore, the content of the out.txt file will be ??0123456789??.

NEW QUESTION 16
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 19
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 23
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 25
.....

Thank You for Trying Our Product

We offer two products:

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

2nd - Questons and Answers in PDF Format

1z0-829 Practice Exam Features:

* 1z0-829 Questions and Answers Updated Frequently

* 1z0-829 Practice Questions Verified by Expert Senior Certified Staff

* 1z0-829 Most Realistic Questions that Guarantee you a Pass on Your FirstTry

* 1z0-829 Practice Test Questions in Multiple Choice Formats and Updatesfor 1 Year