Oracle 1Z0-829 Practice Questions

Shared by Beard on 10-11-2026

For More Free Questions and Preparation Resources

Check the Links on Last Page

Question 1

Question Type: MultipleChoice

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?

Options:

A- int sum = listOfNumbers. parallelStream () reduce (5, Integer:: sum) ;
B- int sum = listOfNumbers. Stream () reduce (5, (a, b) -> a + b) ;
C- int sum = listOfNumbers. Stream () reduce (Integer:: sum) ; +5;
D- int sum = listOfNumbers. parallelStream () reduce ({m, n) -> m +n) orElse (5) +5;
E- int sum = listOfNumbers. 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: Optional.of(6) + 5 = 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.Reference:

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()

Question 2

Question Type: MultipleChoice

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?

Options:

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 order1.The compareTo() method is used to implement the Comparable interface, which defines the natural order of objects of a class2.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()3.Reference:Comparator (Java SE 17 & JDK 17),Comparable (Java SE 17 & JDK 17),Stream (Java SE 17 & JDK 17)

Question 3

Question Type: MultipleChoice

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?

Options:

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.Reference:

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)

Question 4

Question Type: MultipleChoice

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?

Options:

A- At line n1, implement the java.io, Serializable interface.
B- At line n3, replace readObject (0 with readLine().
C- At Line n3, replace Car with LuxurayCar.
D- At Line n1, implement the java.io.AutoCloseable interface
E- At line n2, in the main method signature, add throws IOException, ClassCastException.
F- At line n2, in the main method signature, add throws IoException, ClassNotFoundException.

Answer:

F

Explanation:

The code fragment is trying to read an object from a file using the ObjectInputStream class. This class throws an IOException and a ClassNotFoundException. To handle these exceptions, the main method signature should declare that it throws these exceptions. Otherwise, the code will not compile. If the main method throws these exceptions, the code will print Wagon : 200, which is the result of calling the toString method of the LuxuryCar object that was written to the file.Reference:ObjectInputStream (Java SE 17 & JDK 17) - Oracle,ObjectOutputStream (Java SE 17 & JDK 17) - Oracle

Question 5

Question Type: MultipleChoice

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?

Options:

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.

Question 6

Question Type: MultipleChoice

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?

Options:

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:Reference:AtomicInteger (Java SE 17 & JDK 17) - Oracle

P2P exams

Question 7

Question Type: MultipleChoice

Which statement is true about migration?

Options:

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

P2P exams

Answer:

B

Explanation:

The answer is B because a bottom-up migration is a strategy for modularizing an existing application by moving its dependencies to the module path one by one, starting from the lowest-level libraries and ending with the application itself. This way, each module can declare its dependencies on other modules using the module-info.java file, and benefit from the features of the Java Platform Module System (JPMS), such as reliable configuration, strong encapsulation, and service loading.

Option A is incorrect because a top-down migration is a strategy for modularizing an existing application by moving it to the module path first, along with its dependencies as automatic

modules. Automatic modules are non-modular JAR files that are treated as modules with some limitations, such as not having a module descriptor or a fixed name. A top-down migration allows the application to use the module path without requiring all of its dependencies to be modularized first.

Option C is incorrect because a top-down migration does not require any specific order of migrating modules, as long as the application is moved first and its dependencies are moved as automatic modules. A bottom-up migration, on the other hand, requires the required modules to migrate before the modules that depend on them.

Option D is incorrect because unnamed modules are not automatic modules in any migration strategy. Unnamed modules are modules that do not have a name or a module descriptor, such as classes loaded from the class path or dynamically generated classes. Unnamed modules have unrestricted access to all other modules, but they cannot be accessed by named modules, except through reflection with reduced security checks.Reference:

Oracle Certified Professional: Java SE 17 Developer

Java SE 17 Developer

OCP Oracle Certified Professional Java SE 17 Developer Study Guide

Migrating to Modules (How and When) - JavaDeploy

Java 9 Modularity: Patterns and Practices for Developing Maintainable Applications

Question 8

Question Type: MultipleChoice

Given:

```
public class Test {
   public String attach1(List<String> data) {
       return data.parallelStream().reduce("w", (n,m) -> n+m, String::concat);
   }
   public String attach2(List<String> data) {
      return data.parallelStream().reduce((1, p)-> 1+p).get();
   }

   public static void main(String[] args) {
      Test t = new Test();
      var list = List.of("Table", "Chair");
      String x= t.attach1(list);
      String y= t.attach2(list);
     System.out.print(x+ " "+y);
    }
}
```

What is the result?

Options:

A- Tablechair Tablechair
B- Wtablechair tableChair
C- A RuntimeException is thrown
D- Only one of the objects previously referenced by t1 is eligible for garbage collection.
E- Compilation fails

Answer:

E

Explanation:

The code fragment will fail to compile because the class name and the constructor name do not match. The class name is Furniture, but the constructor name is Wtable. This will cause a syntax error. The correct way to define a constructor is to use the same name as the class name. Therefore, the code fragment should change the constructor name to Furniture or change the class name to Wtable.

Question 9

Question Type: MultipleChoice

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?

Options:

A- int sum = listOfNumbers. parallelStream () reduce (5, Integer:: sum) ;
B- int sum = listOfNumbers. Stream () reduce (5, (a, b) -> a + b) ;
C- int sum = listOfNumbers. Stream () reduce (Integer:: sum) ; +5;
D- int sum = listOfNumbers. parallelStream () reduce ({m, n) -> m +n) orElse (5) +5;
E- int sum = listOfNumbers. 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: Optional.of(6) + 5 = 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.Reference:

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()

To Get Premium Files for 1Z0-829 Visit

https://www.p2pexams.com/products/1z0-829

For More Free Questions Visit

https://www.p2pexams.com/oracle/pdf/1z0-829

