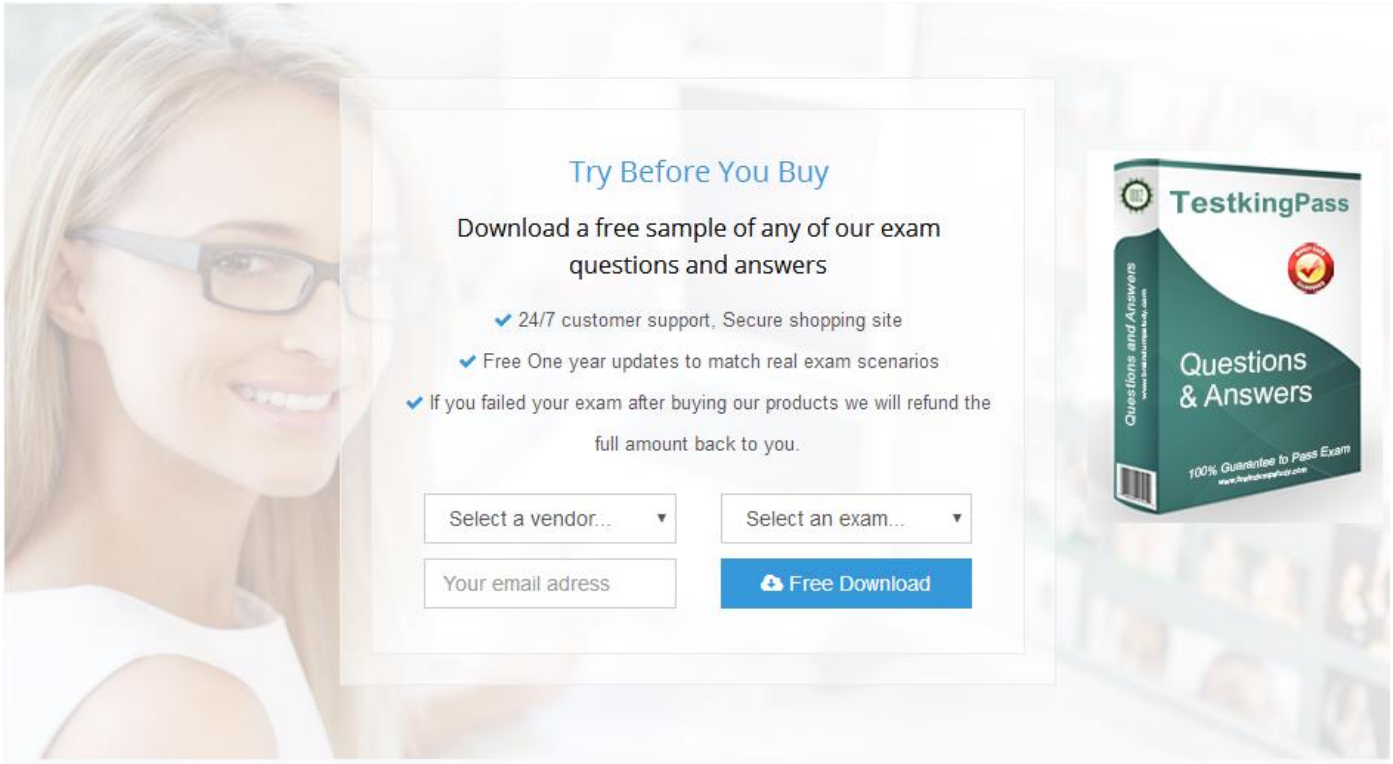
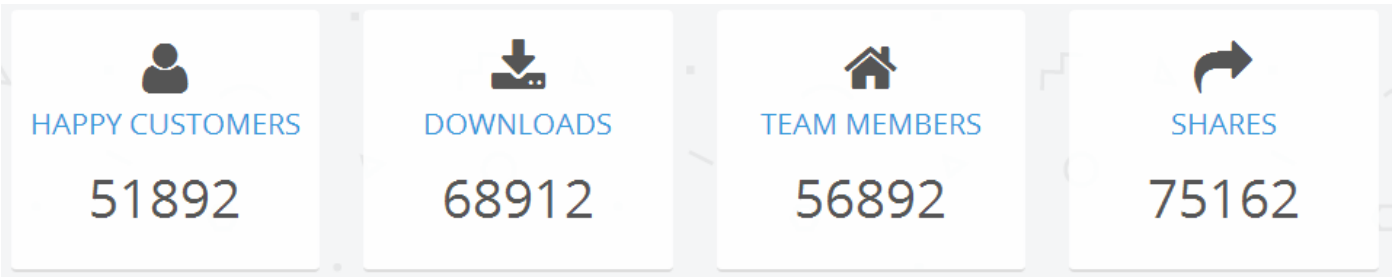
TestkingPass





http://www.testkingpass.com

Reliable test dumps & stable pass king & valid test questions

Exam : **1z0-830**

Title : Java SE 21 Developer Professional

Vendor : Oracle

Version : DEMO

NO.1 Given:
java
Object input = 42;
String result = switch (input) {
case String s -> "It's a string with value: " + s;
case Double d -> "It's a double with value: " + d;
case Integer i -> "It's an integer with value: " + i;
};
System.out.println(result);
What is printed?
A. null
B. It's a string with value: 42
C. It's a double with value: 42
D. It's an integer with value: 42
E. It throws an exception at runtime.
F. Compilation fails.
Answer: F
Explanation:
* Pattern Matching in switch
* The switch expression introduced inJava 21supportspattern matchingfor different types.
* However,a switch expression must be exhaustive, meaningit must cover all possible cases or provide a default case.
* Why does compilation fail?
* input is an Object, and the switch expression attempts to pattern-match it to String, Double, and Integer.
* If input had been of another type (e.g., Float or Long), there would beno matching case, leading to anon-exhaustive switch.
* Javarequires a default caseto ensure all possible inputs are covered.
* Corrected Code (Adding a default Case)
java
Object input = 42;
String result = switch (input) {
case String s -> "It's a string with value: " + s;
case Double d -> "It's a double with value: " + d;
case Integer i -> "It's an integer with value: " + i;
default -> "Unknown type";
};
System.out.println(result);
* With this change, the codecompiles and runs successfully.
* Output:
vbnet
It's an integer with value: 42
Thus, the correct answer is:Compilation failsdue to a missing default case.
References:
* Java SE 21 - Pattern Matching for switch

* Java SE 21 - switch Expressions

NO.2 Given:
java
List<String> l1 = new ArrayList<>(List.of("a", "b"));
List<String> l2 = new ArrayList<>(Collections.singletonList("c"));
Collections.copy(l1, l2);
l2.set(0, "d");
System.out.println(l1);
What is the output of the given code fragment?
A. [a, b]
B. [d, b]
C. [c, b]
D. An UnsupportedOperationException is thrown
E. An IndexOutOfBoundsException is thrown
F. [d]
Answer: C
Explanation:
In this code, two lists l1 and l2 are created and initialized as follows:
* l1 Initialization:
* Created using List.of("a", "b"), which returns an immutable list containing the elements "a" and "b".
* Wrapped with new ArrayList<>(...) to create a mutable ArrayList containing the same elements.
* l2 Initialization:
* Created using Collections.singletonList("c"), which returns an immutable list containing the single element "c".
* Wrapped with new ArrayList<>(...) to create a mutable ArrayList containing the same element.
State of Lists Before Collections.copy:
* l1: ["a", "b"]
* l2: ["c"]
Collections.copy(l1, l2):
The Collections.copy method copies elements from the source list (l2) into the destination list (l1). The destination list must have at least as many elements as the source list; otherwise, an IndexOutOfBoundsException is thrown.
In this case, l1 has two elements, and l2 has one element, so the copy operation is valid. After copying, the first element of l1 is replaced with the first element of l2:
* l1 after copy: ["c", "b"]
l2.set(0, "d"):
This line sets the first element of l2 to "d".
* l2 after set: ["d"]
Final State of Lists:
* l1: ["c", "b"]
* l2: ["d"]
The System.out.println(l1); statement outputs the current state of l1, which is ["c", "b"]. Therefore, the correct answer is C: [c, b].

NO.3 Given:
java
List<Long> cannesFestivalfeatureFilms = LongStream.range(1, 1945)
.boxed()
.toList();
try (var executor = Executors.newVirtualThreadPerTaskExecutor()) {
cannesFestivalfeatureFilms.stream()
.limit(25)
.forEach(film -> executor.submit(() -> {
System.out.println(film);
}));
}
What is printed?
A. Numbers from 1 to 25 sequentially
B. Numbers from 1 to 25 randomly
C. Numbers from 1 to 1945 randomly
D. An exception is thrown at runtime
E. Compilation fails
Answer: B
Explanation:
* Understanding LongStream.range(1, 1945).boxed().toList();
* LongStream.range(1, 1945) generates a stream of numbersfrom 1 to 1944.
* .boxed() converts the primitive long values to Long objects.
* .toList() (introduced in Java 16)creates an immutable list.
* Understanding Executors.newVirtualThreadPerTaskExecutor()
* Java 21 introducedvirtual threadsto improve concurrency.
* Executors.newVirtualThreadPerTaskExecutor()creates a new virtual thread per submitted task
, allowing highly concurrent execution.
* Execution Behavior
* cannesFestivalfeatureFilms.stream().limit(25) # Limits the stream to thefirst 25 numbers(1 to
25).
* .forEach(film -> executor.submit(() -> System.out.println(film)))
* Each film is printed inside a virtual thread.
* Virtual threads execute asynchronously, meaning numbers arenot guaranteed to print sequentially.
* Output will contain numbers from 1 to 25, but their order is random due to concurrent execution.
* Possible Output (Random Order)
python-repl
3
1
5
2
4
7
25
* The ordermay differ in each rundue to concurrent execution.

Thus, the correct answer is:"Numbers from 1 to 25 randomly."
References:
* Java SE 21 - Virtual Threads
* Java SE 21 - Executors.newVirtualThreadPerTaskExecutor()

NO.4 What do the following print?
java
import java.time.Duration;
public class DividedDuration {
public static void main(String[] args) {
var day = Duration.ofDays(2);
System.out.print(day.dividedBy(8));
}
}
A. PT6H
B. PT0H
C. It throws an exception
D. PT0D
E. Compilation fails
Answer: A
Explanation:
In this code, a Duration object day is created representing a duration of 2 days using the Duration.ofDays(2) method. The dividedBy(long divisor) method is then called on this Duration object with the argument 8.
The dividedBy(long divisor) method returns a copy of the original Duration divided by the specified value. In this case, dividing 2 days by 8 results in a duration of 0.25 days. In the ISO-8601 duration format used by Java's Duration class, this is represented as PT6H, which stands for a period of 6 hours.
Therefore, the output of the System.out.print statement is PT6H.

NO.5 Which of the followingisn'ta correct way to write a string to a file?
A. java
Path path = Paths.get("file.txt");
byte[] strBytes = "Hello".getBytes();
Files.write(path, strBytes);
B. java
try (BufferedWriter writer = new BufferedWriter("file.txt")) {
writer.write("Hello");
}
C. java
try (FileOutputStream outputStream = new FileOutputStream("file.txt")) { byte[] strBytes = "Hello".getBytes(); outputStream.write(strBytes);
}
D. java
try (PrintWriter printWriter = new PrintWriter("file.txt")) {

printWriter.printf("Hello %s", "James");
}
E. None of the suggestions
F. java
try (FileWriter writer = new FileWriter("file.txt")) {
writer.write("Hello");
}
Answer: B
Explanation:
(BufferedWriter writer = new BufferedWriter("file.txt") is incorrect.)
Theincorrect statementisoption Bbecause BufferedWriterdoes nothave a constructor that accepts a String (file name) directly. The correct way to use BufferedWriter is to wrap it around a FileWriter, like this:
java
try (BufferedWriter writer = new BufferedWriter(new FileWriter("file.txt"))) { writer.write("Hello");
}
Evaluation of Other Options:
Option A (Files.write)# Correct
* Uses Files.write() to write bytes to a file.
* Efficient and concise method for writing small text files.
Option C (FileOutputStream)# Correct
* Uses a FileOutputStream to write raw bytes to a file.
* Works for both text and binary data.
Option D (PrintWriter)# Correct
* Uses PrintWriter for formatted text output.
Option F (FileWriter)# Correct
* Uses FileWriter to write text data.
Option E (None of the suggestions)# Incorrect becauseoption Bis incorrect.

NO.6 Given:
java
public class ThisCalls {
public ThisCalls() {
this(true);
}
public ThisCalls(boolean flag) {
this();
}
}
Which statement is correct?
A. It does not compile.
B. It throws an exception at runtime.
C. It compiles.
Answer: A
Explanation:

In the provided code, the class ThisCalls has two constructors:
* No-Argument Constructor (ThisCalls()):
* This constructor calls the boolean constructor with this(true);.
* Boolean Constructor (ThisCalls(boolean flag)):
* This constructor attempts to call the no-argument constructor with this();.
This setup creates a circular call between the two constructors:
* The no-argument constructor calls the boolean constructor.
* The boolean constructor calls the no-argument constructor.
Such a circular constructor invocation leads to a compile-time error in Java, specifically "recursiveconstructor invocation." The Java Language Specification (JLS) states:
"It is a compile-time error for a constructor to directly or indirectly invoke itself through a series of one or more explicit constructor invocations involving this." Therefore, the code will not compile due to this recursive constructor invocation.

NO.7 Given:
java
var counter = 0;
do {
System.out.print(counter + " ");
} while (++counter < 3);
What is printed?
A. 0 1 2 3
B. 0 1 2
C. 1 2 3 4
D. 1 2 3
E. An exception is thrown.
F. Compilation fails.
Answer: B
Explanation:
* Understanding do-while Execution
* A do-while loopexecutes at least oncebefore checking the condition.
* ++counter < 3 increments counterbeforeevaluating the condition.
* Step-by-Step Execution
* Iteration 1:counter = 0, print "0", then ++counter becomes 1, condition 1 < 3 istrue.
* Iteration 2:counter = 1, print "1", then ++counter becomes 2, condition 2 < 3 istrue.
* Iteration 3:counter = 2, print "2", then ++counter becomes 3, condition 3 < 3 isfalse, so loop exits.
* Final Output
0 1 2
Thus, the correct answer is:0 1 2
References:
* Java SE 21 - Control Flow Statements
* Java SE 21 - do-while Loop

NO.8 Given:
java
public class Test {

```
static int count;
synchronized Test() {
count++;
}
public static void main(String[] args) throws InterruptedException {
Runnable task = Test::new;
Thread t1 = new Thread(task);
Thread t2 = new Thread(task);
t1.start();
t2.start();
t1.join();
t2.join();
System.out.println(count);
}
}
```

What is the given program's output?

A. It's either 1 or 2

B. It's either 0 or 1

C. It's always 2

D. It's always 1

E. Compilation fails

Answer: E

Explanation:

In this code, the Test class has a static integer field count and a constructor that is declared with the synchronized modifier. In Java, the synchronized modifier can be applied to methods to control access to critical sections, but it cannot be applied directly to constructors. Attempting to declare a constructor as synchronized will result in a compilation error.

Compilation Error Details:

The Java Language Specification does not permit the use of the synchronized modifier on constructors.

Therefore, the compiler will produce an error indicating that the synchronized modifier is not allowed in this context.

Correct Usage:

If you need to synchronize the initialization of instances, you can use a synchronized block within the constructor:

java

```
public class Test {
static int count;
Test() {
synchronized (Test.class) {
count++;
}
}
public static void main(String[] args) throws InterruptedException {
Runnable task = Test::new;
Thread t1 = new Thread(task);
```

```
Thread t2 = new Thread(task);
t1.start();
t2.start();
t1.join();
t2.join();
System.out.println(count);
}
}
```

In this corrected version, the synchronized block within the constructor ensures that the increment operation on count is thread-safe.

Conclusion:

The original program will fail to compile due to the illegal use of the synchronized modifier on the constructor. Therefore, the correct answer is E: Compilation fails.