IT Skill Test GIC Myanmar

Duration: 30 Minutes Total Questions: 8

---

1. You're working on a Java application that processes large amounts of data. The following method is causing an OutOfMemoryError. What's the most likely cause and how would you fix it?

```
public static List<String> processData(String[] data) {
    List<String> result = new ArrayList<>();
    for (String item : data) {
        result.add(item.toUpperCase());
    }
    return result;
}
```

   A.Use a LinkedList instead of an ArrayList

   B.Increase the heap size using -Xmx JVM argument

   C.Process the data in batches instead of all at once

   D.Use a StringBuilder instead of String concatenation

2. You're reviewing code and come across the following snippet. What potential issue should you be concerned about?

```
public class UserService {
    private static final String API_KEY = "abc123xyz";

    public void authenticateUser(String username, String password) {
        // Authentication logic here
    }
}
```

   A.The API_KEY is not declared as final

B. The API_KEY is hardcoded in the source code

C. The authenticateUser method is not static

D. The class should be declared as final

3. You're developing a multi-threaded application in Java. Which of the following code snippets demonstrates the correct way to make a method thread-safe?

A.
```
public void updateCounter() {
    counter++;
}
```

B.
```
public synchronized void updateCounter() {
    counter++;
}
```

C.
```
public void updateCounter() {
    synchronized(this) {
        counter++;
    }
}
```

D.
```
public static void updateCounter() {
    counter++;
}
```

4. You're reviewing a colleague's Java code and notice the following pattern repeated in several places:

```
public void processData(String data) {
    if (data != null) {
        if (!data.isEmpty()) {
            // Process data
        } else {
            throw new IllegalArgumentException("Data is empty");
        }
    } else {
        throw new IllegalArgumentException("Data is null");
    }
}
```

What suggestion would you make to improve this code?

A.Use Optional<String> as the parameter type

B.Use the @NotNull annotation on the parameter

C.Use Objects.requireNonNull() at the start of the method

D.Use Apache Commons Lang's StringUtils for null and empty checks

5. You're working on a Java application that needs to perform some cleanup operations when it shuts down. Which of the following is the best way to ensure your cleanup code is executed?

A.Use System.exit() to terminate the application and perform cleanup

B.Implement a shutdown hook using Runtime.getRuntime().addShutdownHook()

C.Use a finally block in the main method

D.Implement the AutoCloseable interface and use try-with-resources

6. You are debugging a Java application that processes customer orders. The following method is supposed to calculate the total price of an order, including tax, but it's not working correctly. Identify the bug in the code:

```
public double calculateTotalPrice(double subtotal, double taxRate) {
    double tax = subtotal * taxRate;
    double total = subtotal + tax;
    return total;
    total = Math.round(total * 100) / 100.0;
}
```

A.double tax = subtotal * taxRate;

B.double total = subtotal + tax;

C.return total;

D. total = Math.round(total * 100) / 100.0;

7. In a Java web application, you're implementing a caching mechanism to improve performance. Which of the following code snippets represents the best practice for implementing a thread-safe singleton cache?

A.
```
public class Cache {
      private static Cache instance;
      private Cache() {}
      public static Cache getInstance() {
          if (instance == null) {
              instance = new Cache();
          }
          return instance;
      }
  }
```

B.
```
public class Cache {
      private static Cache instance = new Cache();
      private Cache() {}
      public static Cache getInstance() {
          return instance;
      }
  }
```

C.
```
public class Cache {
      private static Cache instance;
      private Cache() {}
      public static synchronized Cache getInstance() {
          if (instance == null) {
              instance = new Cache();
          }
          return instance;
      }
  }
```

D.
```
public class Cache {
      private static volatile Cache instance;
```

```
        private Cache() {}
        public static Cache getInstance() {
            if (instance == null) {
                synchronized (Cache.class) {
                    if (instance == null) {
                        instance = new Cache();
                    }
                }
            }
            return instance;
        }
    }
```

8. You're working on a Java application that processes large amounts of data. The following method is meant to filter a list of integers and return only the even numbers. However, it's causing an OutOfMemoryError when processing very large lists. Identify the most efficient fix for this issue:

```
public List<Integer> filterEvenNumbers(List<Integer> numbers) {
    List<Integer> result = new ArrayList<>();
    for (Integer num : numbers) {
        if (num % 2 == 0) {
            result.add(num);
        }
    }
    return result;
}
```

```
A.public List<Integer> filterEvenNumbers(List<Integer> numbers) {
      return numbers.parallelStream()
                .filter(num -> num % 2 == 0)
                .collect(Collectors.toList());
  }
```

```
B.public List<Integer> filterEvenNumbers(List<Integer> numbers) {
      return numbers.stream()
                .filter(num -> num % 2 == 0)
```

```
                    .collect(Collectors.toList());
    }
```

C.public Stream<Integer> filterEvenNumbers(List<Integer> numbers) {

```
        return numbers.stream()
                    .filter(num -> num % 2 == 0);
    }
```

D.public List<Integer> filterEvenNumbers(List<Integer> numbers) {

```
        List<Integer> result = new LinkedList<>();
        for (Integer num : numbers) {
            if (num % 2 == 0) {
                result.add(num);
            }
        }
        return result;
    }
```