

# ***KillTest***

Higher Quality, Better Service!



## **Q&A**

<http://www.killtest.com>

We offer free update service for one year.

**Exam** : **1Z0-813**

**Title** : Upgrade to Java SE 8 OCP  
(Java SE 6 and all prior  
versions)

**Version** : DEMO

1. Given the code fragment:

```
//line n1
Double d = str.average().getAsDouble();
System.out.println("Average = " + d);
```

Which should be inserted into the line n1 to print Average = 2.5?

- A. Stream str = Stream.of(1, 2, 3, 4);
- B. IntStream str = IntStream.of(1, 2, 3, 4);
- C. DoubleStream str = Stream.of(1.0, 2.0, 3.0, 4.0);
- D. IntStream str = Stream.of(1, 2, 3, 4)

**Answer: B**

**Explanation:**

Use IntStream.

Reference: <https://docs.oracle.com/javase/8/docs/api/java/util/stream/IntStream.html>

2. Given the interface:

```
public interface IdGenerator {
    int getNextId();
}
```

Which class implements IdGenerator in a thread-safe manner, so that no threads can get a duplicate id value during concurrent access?

A.

```
public class Generator implements IdGenerator {
    private int id = 0;
    public int getNextId() {
        return ++id;
    }
}
```

B.

```
public class Generator implements IdGenerator {
    private AtomicInteger id = new AtomicInteger(0);
    public int getNextId() {
        return id.incrementAndGet();
    }
}
```

C.

```
public class Generator implements IdGenerator {
    private volatile int id = 0;
    public int getNextId() {
        return ++id;
    }
}
```

D.

```
public class Generator implements IdGenerator {
    private int id = 0;
    public int getNextId() {
        synchronized(new Generator()) {
            return ++id;
        }
    }
}
```

E.

```
public class Generator implements IdGenerator {
    private int id = 0;
    public int getNextId() {
        synchronized(id) {
            return ++id;
        }
    }
}
```

**Answer: B**

**Explanation:**

An AtomicInteger is used in applications such as atomically incremented counters, and cannot be used as a replacement for an Integer.

However, this class does extend Number to allow uniform access by tools and utilities that deal with numerically-based classes. The incrementAndGet() method atomically increments by one the current value.

Reference: <http://docs.oracle.com/javase/8/docs/api/java/util/concurrent/atomic/AtomicInteger.html#incrementAndGet->

3.Which two statements are true about localizing an application? (Choose two.)

- A. Language codes use lowercase and region codes use uppercase letters.
- B. Resource bundle files include date and currency information.
- C. Language and region-specific programs are created using localized data.
- D. Support for new regional languages does not require recompilation of the code.
- E. Textual elements (messages and GUI labels) are hard-coded on the code.

**Answer: AC**

**Explanation:**

A: The following examples create Locale objects for the French language in Canada, the English language in the U.S. and Great Britain.

```
aLocale = new Locale("fr", "CA");
```

```
bLocale = new Locale("en", "US");
```

```
cLocale = new Locale("en", "GB");
```

C: Localization is the process of adapting an internationalized application to support a specific region or locale.

Incorrect Answers:

B: Resource bundle files does not include date a currency information. Date and currency information are stored in locales, not in resource bundle files.

D: Recompilation is not necessary.

E: Textual elements are not hard-coded on the code.

References:

<https://docs.oracle.com/javase/tutorial/i18n/locale/create.html>

<http://docs.oracle.com/javaee/6/tutorial/doc/bnaxw.html>

4. Given the code fragment:

```
public class Test {
    public static void main(String[] args) {
        Greeter g = (s) -> {
            return s + " Welcome!";
        };
        System.out.println(g.greet("Kathy"));
    }
}
```

Which is the valid definition for the Greeter interface to enable the code fragment to print KathyWelcome!?

A.

```
public interface Greeter {
    public String greet(String name);
}
```

B.

```
public interface Greeter {
    public default String greet(String name) {
        return name;
    }
    public String greet(String name, String salute);
}
```

C.

```
public interface Greeter<T> {
    public static String greet(T name);
}
```

D.

```
public interface Greeter {
    public default String greet(String name);
}
```

**Answer:** A

**Explanation:**

Code example works fine:

```
public class Test {  
  
    public interface Greeter {  
        public String greet(String name);  
    }  
  
    public static void main(String[] args) {  
        Greeter g = (s) -> {  
            return s + "Welcome!";  
        };  
        System.out.println(g.greet("Kathy"));  
    }  
}
```

5. Given the code fragment:

```
public class TestString {  
    public static void main(String[] args) {  
  
        String str=null;  
  
        switch(str){  
            case "":  
                System.out.println("blank"); break;  
            case "null":  
                System.out.println("NULL"); break;  
            default:  
                System.out.println("invalid");  
        }  
    }  
}
```

What is the result?

- A. invalid
- B. An exception is thrown at runtime.
- C. NULL
- D. Compilation fails.
- E. blank

**Answer: B**

**Explanation:**

A java.lang.NullPointerException is through at line switch(str) {.