CHAPTER 18

■ ■ ■

Stack Walking

QUESTIONS AND EXERCISES

1. Is the following statement true or false?

   Each thread in Java maintains its own stack in which each Java method invocation by the thread is represented as a frame on the stack.

   Answer:

   Yes

2. What is stack walking?

   Answer:

   Stack walking (or stack traversal) is the process of traversing the stack frames of a thread and inspecting the frames' contents.

3. What is the fully qualified name of the class that supports the Stack-Walking API in JDK9?

   Answer:

   The `java.lang.StackWalker` class.

4. Name the class whose instances represents a frame on the stack of a thread.

   Answer:

   The `java.lang.StackTraceElement` class.

5. JDK9 added an interface named `StackWalker.StackFrame`. What does an instance of this interface represent?

   Answer:

   An instance of the `StackWalker.StackFrame` interface represents a stack frame.

   Prior to JDK 9, an instance of the `StackTraceElement` class was used to represent a stack frame. The Stack-Walking API in JDK 9 uses an instance of the `StackWalker.StackFrame` interface to represent a stack frame.

   There are no concrete implementation class of the `StackWalker.StackFrame` interface for you to use directly. The Stack-Walking API in the JDK provides you instances of the interface when you retrieve stack frames.

6. Explain the difference in behaviors of the `StackWalker` instance with respect to the following three options that you can use to configure it:
   `RETAIN_CLASS_REFERENCE, SHOW_HIDDEN_FRAMES, and SHOW_REFLECT_FRAMES`

   Answer:

   - If the `RETAIN_CLASS_REFERENCE` option is specified, the frames returned by the `StackWalker` will contain the reference of the `Class` object of the declaring class of the method represented by the frame. You also need to specify this option if you want to get the `Class` object's reference of the caller of a method. By default, this option is absent.
   - By default, implementation specific and reflection frames are not included in the stream of frames returned by the `StackWalker` class. Use the `SHOW_HIDDEN_FRAMES` option to include all hidden frames.
   - If `SHOW_REFLECT_FRAMES` option is specified, the stream of frames returned by the `StackWalker` class includes the reflection frames. Using this option may still hide the implementation specific frames, which you can show using the `SHOW_HIDDEN_FRAMES` option.

7. The following snippet of code obtains a `StackWalker` instance:

   ```
   // Get a StackWalker with the default configuration
   StackWalker sw1 = StackWalker.getInstance();
   ```

   Will this `StackWalker` include hidden frames and retain class references?

   Answer:

No

8. When the following `Test` class is run, it throws an `IllegalCallerException`. Explain the reason for this exception.

```
// Test.java
package com.jdojo.stackwalker.exercises;

import static java.lang.StackWalker.Option.RETAIN_CLASS_REFERENCE;

public class Test {
    public static void main(String[] args) {
        StackWalker stackWalker =
            StackWalker.getInstance(RETAIN_CLASS_REFERENCE);
        Class<?> callerCls = stackWalker.getCallerClass();
        System.out.println(callerCls);
    }
}
```

Answer:

When the `Test` class is run, the `main()` method is called by the JVM and there is no caller frame on the stack, which results in an `IllegalStateException`.

9. Is the following statement true or false?

The `getCallerClass()` method of the `StackWalker` class filters out all hidden and reflection frames while finding the caller class, irrespective of the options used to obtain the `StackWalker` instance.

Answer:

True

10. When a security manager is installed, what `RuntimePermission` must be granted to create a `StackWalker` with the `RETAIN_CLASS_REFERENCE` option?

Answer:

The codebase using the `RETAIN_CLASS_REFERENCE` option must be granted a `java.lang.RuntimePermission` with a value of `"getStackWalkerWithClassReference"`.

11. What will be the output when the following class `Test2` is run?

```
// Test2.java
package com.jdojo.stackwalker.exercises;

public class Test2 {
    public static void main(String[] args) {
        StackWalker.getInstance()
            .forEach(f -> System.out.println(f.getClassName()));
    }
}
```

Answer:

```
com.jdojo.stackwalker.exercises.Test2
```
