State not Maintained

In Mockito, when you create a mock object using `mock()`, it creates a "bare-bones" mock without any actual behavior or state. Therefore, when you try to call methods like `put()` on the mock object, it won't perform any actual operation on the map; it will just return default values or behaviors as specified by Mockito.

In your case, when you call `mockMap.put("person", name)`, it doesn't actually add the `"person"` key with the `name` map value to the mock map. Instead, it returns default values (which are usually `null` for method return types). Therefore, the size of the mock map remains 0.

If you want to add behavior to the mock map so that it behaves like a real map, you can use Mockito's `when().thenReturn()` syntax to specify the behavior. However, for complex behavior like nested maps, you might need to use a different approach, such as using a real map implementation or a custom mock with the desired behavior.

Here's how you might simulate the behavior of putting values into a map using Mockito:

```java
import org.junit.jupiter.api.Test;
import java.util.HashMap;
import java.util.Map;
import static org.mockito.Mockito.*;

class ExampleTest {

    @Test
    void testMockMapBehavior() {
        // Create a mock map
        Map<String, Object> mockMap = mock(Map.class);

        // Define behavior for put method
        when(mockMap.put(anyString(), any())).thenReturn(null); // For simplicity,
always return null

        // Create nested maps
        Map<String, Object> firstName = new HashMap<>();
        firstName.put("firstName", "pratik");
        Map<String, Object> name = new HashMap<>();
        name.put("name", firstName);

        // Add values to the mock map
        mockMap.put("person", name);
        // Verify behavior
        verify(mockMap).put("person", name);

        // Check size of the mock map
        System.out.println("Size of mock map: " + mockMap.size()); // Output: 0
    }
}
```

In this example, we use `when().thenReturn()` to define the behavior of the `put` method on the mock map. We then add values to the mock map and verify that the behavior is as expected. However, note that the size of the mock map remains 0 because Mockito does not maintain state like a real map would.