Appendix A Statement Patterns

Constructor	
Description	Constructor statements are the block of code that cause an instance of object to be created. Constructor can be evoked using new operator or a builder. The below code sample 1 and code sample 2 are equivalent. During dynamic analysis, we observed that the new instance is created with class attributes equal to either the value of explicit arguments or default value. So we are able to collect the information about both explicit argument and implicit arguments values during object instantiation. Occasionally, the use of anonymous class provides a specialization of a base class without explicitly defining a new class via the class expression, as shown by code sample 3.
Code Sample 1	`1. Option option = new Option("f", null);`
Code Sample 2	`1. Option option = Option.builder("f").desc(null).build();`

Code Sample 3	`1. GenericItem item = new GenericItem("Test") {`
	`2. @Override`
	`3. public List<Class<? extends State>> getAcceptedDataTypes() {`
	`4. return null;`
	`5. }`
	`6. };`

Template Sentence	This statement instantiates an OBJECT_TYPE with explicit arguments [PARA_NAME equal to VALUE,]* and implicit arguments [PARA_NAME equal to VALUE,]* and overriding methods [METHODS]
Generated Sentence 1	This statement instantiates an "Option" with explicit arguments option equal to "f", description equal to null, and implicit arguments longOpt equal to null, numArgs equal to -1, argName equal to null, required equal to false, optionalArg equal to false, valueSeparator equal to 'u0000', type equal to String.class,
Generated Sentence 2	This statement instantiates an "Option" with explicit arguments option equal to "f", description equal to null, and implicit arguments longOpt equal to null, numArgs equal to -1, argName equal to null, required equal to false, optionalArg equal to false, valueSeparator equal to 'u0000', type equal to String.class,
Generated Sentence 3	This statement instantiates an "GenericItem" with explicit arguments name equal to "Test" and overriding method getAcceptedDataTypes.

Expected Exceptions	
Description	In JUnit, there are three ways to test the expected exception: 1. @Test, optional "expected" attribute, 2. Try-catch and always fail(), and 3. @Rule ExpectedException

Code Sample 1	`@Test(expected=IllegalArgumentException.class)`
	`public void testBuilderInsufficientParams1() {`
	`  Option.builder("f").desc("desc").build();`
	`}`

Code Sample 2	`@Test(timeout = 4000)`
	`public void test031() throws Throwable {`
	`  try {`
	`  Option.builder(",ekya2B)3Q?");`
	`  fail("Expecting exception: IllegalArgumentException");`
	`  } catch(IllegalArgumentException e) {`
	`  verifyException("org.apache.commons.cli.OptionValidator",`
	`  e);`
	`  }`
	`}`

Code Sample 3	`@Rule`
	`public ExpectedException thrown = ExpectedException.none();`
	`@Test`
	`public void testDivisionWithException() {`
	`  thrown.expect(ArithmeticException.class);`
	`  thrown.expectMessage(containsString("/ by zero"));`
	`  int i = 1 / 0;`
	`}`

Template Sentence	Expects EXCEPTION_TYPE when LAST_STATEMENT or STATEMENT_WITHIN_TRY_BLOCK
Generated Sentence 1	Expects IllegalArgumentException when instantiating an "Option" with explicit arguments option equal to "f", description equal to "desc", and implicit arguments longOpt equal to null, numArgs equal to -1, argName equal to null, required equal to false, optionalArg equal to false, valueSeparator equal to 'u0000', type equal to String.class,
Generated Sentence 2	Expects IllegalArgumentException when instantiating an "Option" with explicit arguments option equal to ",ekya2B)3Q?" and implicit arguments description = null, longOpt equal to null, numArgs equal to -1, argName equal to null, required equal to false, optionalArg equal to false, valueSeparator equal to 'u0000', type equal to String.class,
Generated Sentence 3	Expects ArithmeticException when declaring an object of the class "int" with the value of 1 divided by 0.

Appendix B Test Case Scenarios

Prerequisites Setup	
Description	Sometimes, there are prerequisites or preconditions that must be fulfilled prior to executing the test. These prerequisites are annotated by @Autowired, which marks a constructor, field, setter method or config method as to be autowired by Spring's dependency injection facilities, @Before, which is executed before each test, @BeforeClass, which runs once before the entire test fixture. Other prerequisites might not be lead be any annotation, such as variable declaration on line 3. If there is any such prerequisites, we add the summary as the first step in generated scenario.

Code Sample	1. **@Autowired**
	2. private WebApplicationContext wac;
	3. private MockMvc mockMvc;
	4.
	5. **@Before**
	6. public void **setup()** {
	7. this.mockMvc = MockMvcBuilders.webAppContextSetup(wac).build();
	8. }
	9.
	10. @Test
	11. public void test() throws Exception {
	12. assertNotNull(mockMvc);
	13. }

Generated Scenario	1. Declares a WebApplicationContext "wac" and a MockMvc "mockMvc". And sets the value of mockMvc to instantiating an "MockMvc" with explicit arguments webAppContextSetup equal to wac, 2. Tests whether mockMvc is not null.

Same Action Sequential Aggregation	
Description	We observe that it is common for test case to contain a series of same action statements but with different arguments consecutively. We aggregate such statements in the same sequential order as one step for the test case scenario.
Code Sample	1. @Test 2. public void test005(){ 3. Option a = new Option("a", ""); 4. a.**addValue**("b1"); 5. a.**addValue**("b2"); 6. assertEquals(2, a.getValues().length); 7. }
Generated Scenario	1. Instantiates an "Option" with explicit arguments option equal to "a", description equal to "", and implicit arguments longOpt equal to null, numArgs equal to -1, argName equal to null, required equal to false, optionalArg equal to false, valueSeparator equal to 'u0000', type equal to String.class, 2. Adds value "b1" and "b2" to the object "a". 3. Tests whether the length of values of a is equal to 2.

Test Sequential Aggregation

Description

We observe that it is common for test case to contain a series of actions and test statements. We aggregate the action and test statement as one step for the test case scenario.

Code Sample

```
1. @Test
2. public void testClear(){
3.     Option a = new Option("a", "");
4.     option.addValue("a");
5.     assertEquals(1, option.getValuesList().size());
6.     option.clearValues();
7.     assertEquals(0, option.getValuesList().size());
8. }
```

Generated Scenario

1. Instantiates an "Option" with explicit arguments option equal to "a", description equal to "", and implicit arguments longOpt equal to null, numArgs equal to -1, argName equal to null, required equal to false, optionalArg equal to false, valueSeparator equal to 'u0000', type equal to String.class,
2. Tests whether the size of the values list of option is equal 1 after adding value "a" to option.
3. Tests whether the size of the values list of option is equal 0 after clearing value of option.

Code Sample

```
1. @Test
2. public void testClear(){
3.     Option option0 = new Option("a", "");
4.     Option option1 = (Option)option0.clone();
5.     boolean boolean0 = option0.equals(option1);
6.     assertEquals(option0.getArgs(), option1.getArgs());
7.     assertEquals(option0.getId(), option1.getId());
8.     assertEquals(option0.getOpt(), option1.getOpt());
9.     assertEquals(option0.getType(), option1.getType());
10.    assertEquals(option0.getLongOpt(), option1.getLongOpt());
11. }
```

Generated Scenario

1. Instantiates an "Option" with explicit arguments option equal to "a", description equal to "", and implicit arguments longOpt equal to null, numArgs equal to -1, argName equal to null, required equal to false, optionalArg equal to false, valueSeparator equal to 'u0000', type equal to String.class,
2. declares an object of the class "Option" using a casting ((Option) option0.clone())
3. Tests whether the number of argument values, id, name, type, and long name of Option0 is equal to Option1

EvoSuite Exception Message Aggregation

Description

EvoSuite generates test cases that test expected exception in the format of try-catch and fail(). When the exception.getMessage() is implemented with appropriate message, EvoSuite incorporate the message as comment within the test case, as shown in code sample 1 line 8. Such comments conveys useful information. So we add such comments into the test case scenario. When exception.getMessage() is not implemented with any message, as shown in code sample 2 line 8, we ignore the comment.

Code Sample 1

```
1. @Test(timeout = 4000)
2. public void test14()  throws Throwable  {
3.     try {
4.         OptionBuilder.create("%m?");
5.         fail("Expecting exception: IllegalArgumentException");
6.     } catch(IllegalArgumentException e) {
7.         //
8.         // The option '%m?' contains an illegal character : '%'
9.         //
10.        verifyException("org.apache.commons.cli.OptionValidator",
11.            e);
12.    }
13.}
```

Generated Scenario 1

1. Expects IllegalArgumentException when instantiating an "Option" with explicit arguments option equal to "%m?" and implicit description equal to null, arguments longOpt equal to null, numArgs equal to -1, argName equal to null, required equal to false, optionalArg equal to false, valueSeparator equal to 'u0000', type equal to String.class **because The option '%m?' contains an illegal character : '%'**.

Code Sample 2

```
1. @Test(timeout = 4000)
2. public void test14()  throws Throwable  {
3.     try {
4.         OptionBuilder.create("%m?");
5.         fail("Expecting exception: IllegalArgumentException");
6.     } catch(IllegalArgumentException e) {
7.         //
8.         // no message in exception (getMessage() returned null)
9.         //
10.        verifyException("org.apache.commons.cli.OptionValidator",
11.            e);
12.    }
13.}
```

Generated Scenario 2

1. Expects IllegalArgumentException when instantiating an "Option" with explicit arguments option equal to "%m?" and implicit description equal to null, arguments longOpt equal to null, numArgs equal to -1, argName equal to null, required equal to false, optionalArg equal to false, valueSeparator equal to 'u0000', type equal to String.class,