Evaluating the Legibility of Generated unit Tests

This form is part of the research of the master's student in the Graduate Program in Computer Science at UFCG, Wesley Brenno, supervised by professors Everton Alves and Melina Mongiovi. This questionnaire aims to evaluate legibility aspects of generated unit test. First, we ask some background questions, then we will present a java class and some snippets of unit test codes and questions related to your impressions regarding the artifacts.

* Required

1. Email *

Participant Background

First, let us know a little bit about you and your professional experience.

2. What country do you currently live in? *

3. What is your current position? *

Examples: Software Engineer, Software Architect, Test Analyst, Undergraduate Student...

4. What long have you been working with Java development? *

Mark only one oval.

	1	2	3	4	5
Years	◯	◯	◯	◯	◯

5. How often do you write unit tests? *

Mark only one oval.

- ◯ Very often
- ◯ Often
- ◯ Sometimes
- ◯ Rarely
- ◯ Never

6. Have you ever used unit test generation tools? *

Mark only one oval.

- ◯ Yes
- ◯ No

7. If you answered "Yes" in the previous question, please list the tools that you used for unit tests generation.

Check all that apply.

- ☐ EvoSuite
- ☐ Randoop

Other: ☐ _____

8. To go to the next section, please select the option corresponding to the last digit of your ID number that is different from 0. This information will only be used to randomly balance questions among participants. *

Example: If your identity number is 1,234,567, you should check option 7. If your identity number is 1,234,560, you should check option 6.

Mark only one oval.

- 1 *Skip to question 9*
- 2 *Skip to question 67*
- 3 *Skip to question 21*
- 4 *Skip to question 75*
- 5 *Skip to question 33*
- 6 *Skip to question 79*
- 7 *Skip to question 45*
- 8 *Skip to question 87*
- 9 *Skip to question 57*

FixedOrderComparator - Evaluating test names

In the following questions, there are some unit tests referring to the FixedOrderComparator class (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/0/FixedOrderComparator.java)

9. Given the unit test below for FixedOrderComparator class, indicate your level of agreement with the suggested test name "testAddAsEqualWithNull" *

```
@Test
public void test() {
    collections.comparators.FixedOrderComparator fixedOrderComparator0 =
            new collections.comparators.FixedOrderComparator();
    java.lang.Object obj2 = null;
    try {
        boolean boolean3 = fixedOrderComparator0.addAsEqual
                ((java.lang.Object) (-1L), obj2);
        org.junit.Assert.fail(
                "Expected exception of type java.lang.IllegalArgumentException; "
                + "message: -1 not known to "
                + "collections.comparators.FixedOrderComparator@33afa13b");
    } catch (java.lang.IllegalArgumentException e) {
    }
}
```

Mark only one oval.

- ◯ Strongly disagree - This test name is completely inappropriate and undescriptive.
- ◯ Disagree - This test name is somewhat inappropriate and undescriptive.
- ◯ Neutral - Neither agree or disagree with this test name.
- ◯ Agree - This test name is somewhat appropriate and descriptive.
- ◯ Strongly Agree - This test name is completely appropriate and descriptive.

10. Please, justify your answer for the previous question

11. Given the unit test below for FixedOrderComparator class, select the most appropriate/descriptive name from the names below. *

```
@Test
public void test()  {
    collections.comparators.FixedOrderComparator fixedOrderComparator0 =
            new collections.comparators.FixedOrderComparator();
    boolean boolean1 = fixedOrderComparator0.isLocked();
    try {
        int int4 = fixedOrderComparator0.
                compare((java.lang.Object) (-1), (java.lang.Object) (-1));
        org.junit.Assert.fail(
                "Expected exception of type java.lang.IllegalArgumentException; "
                + "message: Attempting to compare unknown object -1");
    } catch (java.lang.IllegalArgumentException e) {
    }
    org.junit.Assert.assertTrue
    ("'" + boolean1 + "' != '" + false + "'", boolean1 == false);
}
```

Mark only one oval.

- ( ) mustThrowIllegalArgumentExceptionIfComparingTwoUnknownObjects
- ( ) testCompareThrowsIllegalArgumentException
- ( ) test

12. Please, justify your answer for the previous question

13. Given the following list of candidate names, select the one that you consider the best name for a test that excecises line 274 from the FixedOrderComparator class. (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/FixedOrderComparator.java) *

Mark only one oval.

- testCompareThrowsIllegalArgumentExceptionAndCreatesFixedOrderComparatorTakingObjectArray
- mustThrowIllegalArgumentExceptionIfCompareUnknownObject
- test

14. Please, explain here your answer for the previous question

Skip to question 63

FixedOrderComparator - Evaluating test names

In the following questions, there are some unit tests referring to the FixedOrderComparator class (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/0/FixedOrderComparator.java)

15. Given the unit test below for FixedOrderComparator class, indicate your level of agreement with the suggested test name "testCompareThrowsIllegalArgumentExceptionAndCreatesFixedOrderComparatorTakingObjectArray" *

```
@Test
public void test14()  {
    collections.comparators.FixedOrderComparator fixedOrderComparator0 =
            new collections.comparators.FixedOrderComparator();
    boolean boolean1 = fixedOrderComparator0.isLocked();
    boolean boolean3 = fixedOrderComparator0.add((java.lang.Object) 'a');
    java.lang.Object[] objArray6 = new java.lang.Object[] { (short) 0, "hi!" };
    collections.comparators.FixedOrderComparator fixedOrderComparator7 =
            new collections.comparators.FixedOrderComparator(
            objArray6);
    try {
        int int9 = fixedOrderComparator0
                .compare((java.lang.Object) objArray6, (java.lang.Object) "hi!");
        org.junit.Assert.fail(
                "Expected exception of type java.lang.IllegalArgumentException; "
                + "message: Attempting to "
                + "compare unknown object [Ljava.lang.Object;@57a4d5ee");
    } catch (java.lang.IllegalArgumentException e) {
    }
    org.junit.Assert.assertTrue("'" + boolean1 + "' != '" + false + "'",
            boolean1 == false);
    org.junit.Assert.assertTrue("'" + boolean3 + "' != '" + true + "'",
            boolean3 == true);
    org.junit.Assert.assertNotNull(objArray6);
}
```

Mark only one oval.

- Strongly disagree - This test name is completely inappropriate and undescriptive.
- Disagree - This test name is somewhat inappropriate and undescriptive.
- Neutral - Neither agree or disagree with this test name.
- Agree - This test name is somewhat appropriate and descriptive.
- Strongly Agree - This test name is completely appropriate and descriptive.

16. Please, explain here your answer for the previous question

17. Given the unit test below for FixedOrderComparator class, select the most appropriate, descriptive name from names below. *

```
@Test
public void test() {
    collections.comparators.FixedOrderComparator fixedOrderComparator0 =
            new collections.comparators.FixedOrderComparator();
    java.lang.Object obj2 = null;
    try {
        boolean boolean3 = fixedOrderComparator0.addAsEqual
                ((java.lang.Object) (-1L), obj2);
        org.junit.Assert.fail(
                "Expected exception of type java.lang.IllegalArgumentException; "
                + "message: -1 not known to "
                + "collections.comparators.FixedOrderComparator@33afa13b");
    } catch (java.lang.IllegalArgumentException e) {
    }
}
```

Mark only one oval.

- testAddAsEqualWithNull
- mustThrowIllegalArgumentExceptionIfExistingObjectIsNotInTheKnownObjectSet
- test

18. Please, explain here your answer for the previous question

19. Given a list of candidate test names below, select the test name you think will execute line 274 of FixedOrderComparator class. (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/FixedOrderComparator.java) *

Mark only one oval.

- ◯ mustThrowIllegalArgumentExceptionIfComparingTwoUnknownObjects
- ◯ testCompareThrowsIllegalArgumentException
- ◯ test

20. Please, explain here your answer for the previous question

FixedOrderComparator - Evaluating test names

In the following questions, there are some unit tests referring to the FixedOrderComparator class (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/0/FixedOrderComparator.java)

21. Given the unit test below for FixedOrderComparator class, indicate your level of agreement with the suggested test name "testCompareThrowsIllegalArgumentException" *

```
@Test
public void test()  {
    collections.comparators.FixedOrderComparator fixedOrderComparator0 =
            new collections.comparators.FixedOrderComparator();
    boolean boolean1 = fixedOrderComparator0.isLocked();
    try {
        int int4 = fixedOrderComparator0.
                compare((java.lang.Object) (-1), (java.lang.Object) (-1));
        org.junit.Assert.fail(
                "Expected exception of type java.lang.IllegalArgumentException; "
                + "message: Attempting to compare unknown object -1");
    } catch (java.lang.IllegalArgumentException e) {
    }
    org.junit.Assert.assertTrue
    ("'" + boolean1 + "' != '" + false + "'", boolean1 == false);
}
```

Mark only one oval.

- ◯ Strongly disagree - This test name is completely inappropriate and undescriptive.
- ◯ Disagree - This test name is somewhat inappropriate and undescriptive.
- ◯ Neutral - Neither agree or disagree with this test name.
- ◯ Agree - This test name is somewhat appropriate and descriptive.
- ◯ Strongly Agree - This test name is completely appropriate and descriptive.

22. Please, explain here your answer for the previous question

23. Given the unit test below for FixedOrderComparator class, select the most appropriate, descriptive name from names below. *

```
@Test
public void test14()  {
    collections.comparators.FixedOrderComparator fixedOrderComparator0 =
            new collections.comparators.FixedOrderComparator();
    boolean boolean1 = fixedOrderComparator0.isLocked();
    boolean boolean3 = fixedOrderComparator0.add((java.lang.Object) 'a');
    java.lang.Object[] objArray6 = new java.lang.Object[] { (short) 0, "hi!" };
    collections.comparators.FixedOrderComparator fixedOrderComparator7 =
            new collections.comparators.FixedOrderComparator(
            objArray6);
    try {
        int int9 = fixedOrderComparator0
                .compare((java.lang.Object) objArray6, (java.lang.Object) "hi!");
        org.junit.Assert.fail(
                "Expected exception of type java.lang.IllegalArgumentException; "
                + "message: Attempting to "
                + "compare unknown object [Ljava.lang.Object;@57a4d5ee");
    } catch (java.lang.IllegalArgumentException e) {
    }
    org.junit.Assert.assertTrue("'" + boolean1 + "' != '" + false + "'",
            boolean1 == false);
    org.junit.Assert.assertTrue("'" + boolean3 + "' != '" + true + "'",
            boolean3 == true);
    org.junit.Assert.assertNotNull(objArray6);
}
```

Mark only one oval.

- mustThrowIllegalArgumentExceptionIfCompareUnknownObject
- testCompareThrowsIllegalArgumentExceptionAndCreatesFixedOrderComparatorTakingObjectArray
- test14

24. Please, explain here your answer for the previous question

25. Given a list of candidate test names below, select the test name you think will execute line 195 of FixedOrderComparator class. (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/FixedOrderComparator.java) *

Mark only one oval.

- testAddAsEqualWithNull
- mustThrowIllegalArgumentExceptionIfExistingObjectIsNotInTheKnownObjectSet
- test

26. Please, explain here your answer for the previous question

Skip to question 71

ElitisticListPopulation and ListPopulation - Evaluating test names

In the following questions, there are some unit tests referring to the ElitisticListPopulation class (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/ElitisticListPopulation.java), that implements the ListPopulation abstract class (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/ListPopulation.java)

27. Given the unit test below for ElitisticListPopulation class, indicate your level of agreement with the suggested test name "testFailsToCreateElitisticListPopulationTaking3ArgumentsThrowsNotPositiveExceptio n" *

```
@Test
public void test() throws Throwable {
    math.genetics.Chromosome[] chromosomeArray0 = new math.genetics.Chromosome[] {};
    java.util.ArrayList<math.genetics.Chromosome> chromosomeList1 =
            new java.util.ArrayList<math.genetics.Chromosome>();
    boolean boolean2 = java.util.Collections
            .addAll((java.util.Collection<math.genetics.Chromosome>)
                    chromosomeList1, chromosomeArray0);
    try {
        math.genetics.ElitisticListPopulation elitisticListPopulation5 =
                new math.genetics.ElitisticListPopulation(
                (java.util.List<math.genetics.Chromosome>)
                chromosomeList1, (int) (short) -1, (double) (short) 100);
        org.junit.Assert.fail(
                "Expected exception of type math.exception.NotPositiveException; "
                + "message: population limit has to be positive");
    } catch (math.exception.NotPositiveException e) {
    }
    org.junit.Assert.assertNotNull(chromosomeArray0);
    org.junit.Assert.assertTrue("'" + boolean2 + "' != '" + false + "'", boolean2 == false);
}
```

Mark only one oval.

- Strongly disagree - This test name is completely inappropriate and undescriptive.
- Disagree - This test name is somewhat inappropriate and undescriptive.
- Neutral - Neither agree or disagree with this test name.
- Agree - This test name is somewhat appropriate and descriptive.
- Strongly Agree - This test name is completely appropriate and descriptive.

28. Please, explain here your answer for the previous question

29. Given the unit test below for ElitisticListPopulation class, select the most appropriate, descriptive name from names below. *

```
@Test
public void test() throws Throwable {
    math.genetics.ElitisticListPopulation elitisticListPopulation2 =
            new math.genetics.ElitisticListPopulation(100,
            0.0d);
    java.lang.String str3 = elitisticListPopulation2.toString();
    try {
        elitisticListPopulation2.setElitismRate((double) (byte) -1);
        org.junit.Assert
                .fail("Expected exception of type math.exception.OutOfRangeException; "
                        + "message: elitism rate (-1)");
    } catch (math.exception.OutOfRangeException e) {
    }
    org.junit.Assert.assertTrue("'" + str3 + "' != '" + "[]" + "'", str3.equals("[]"));
}
```

Mark only one oval.

- testSetElitismRateThrowsOutOfRangeExceptionAndToString
- throwExceptionIfAnElitismRateIsSet
- test

30. Please, explain here your answer for the previous question

31. Given a list of candidate test names below, select the test name you think will execute line 236 of ListPopulation class. (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/ListPopulation.java). *

Mark only one oval.

- testIterator
- testIfChromosomeIteratorIsNotNull
- test

32. Please, explain here your answer for the previous question

ElitisticListPopulation and ListPopulation - Evaluating test names

In the following questions, there are some unit tests referring to the ElitisticListPopulation class (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/ElitisticListPopulation.java), that implements the ListPopulation abstract class (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/ListPopulation.java)

33. Given the unit test below for ElitisticListPopulation class, indicate your level of agreement with the suggested test name "testIterator" *

```
@Test
    public void test() throws Throwable {
        math.genetics.ElitisticListPopulation elitisticListPopulation2 =
                new math.genetics.ElitisticListPopulation(100, 0.0d);
        elitisticListPopulation2.setPopulationLimit((int) '4');
        java.util.Iterator<math.genetics.Chromosome> chromosomeItor5 =
                elitisticListPopulation2.iterator();
        java.lang.Class<?> wildcardClass6 = chromosomeItor5.getClass();
        org.junit.Assert.assertNotNull(chromosomeItor5);
        org.junit.Assert.assertNotNull(wildcardClass6);
    }
```

Mark only one oval.

- ◯ Strongly disagree - This test name is completely inappropriate and undescriptive.
- ◯ Disagree - This test name is somewhat inappropriate and undescriptive.
- ◯ Neutral - Neither agree or disagree with this test name.
- ◯ Agree - This test name is somewhat appropriate and descriptive.
- ◯ Strongly Agree - This test name is completely appropriate and descriptive.

34. Please, explain here your answer for the previous question

35. Given the unit test below for ElitisticListPopulation class, select the most appropriate, descriptive name from names below. *

```
@Test
public void test() throws Throwable {
    math.genetics.Chromosome[] chromosomeArray0 = new math.genetics.Chromosome[] {};
    java.util.ArrayList<math.genetics.Chromosome> chromosomeList1 =
            new java.util.ArrayList<math.genetics.Chromosome>();
    boolean boolean2 = java.util.Collections
            .addAll((java.util.Collection<math.genetics.Chromosome>)
                    chromosomeList1, chromosomeArray0);
    try {
        math.genetics.ElitisticListPopulation elitisticListPopulation5 =
                new math.genetics.ElitisticListPopulation(
                (java.util.List<math.genetics.Chromosome>)
                chromosomeList1, (int) (short) -1, (double) (short) 100);
        org.junit.Assert.fail(
                "Expected exception of type math.exception.NotPositiveException; "
                + "message: population limit has to be positive");
    } catch (math.exception.NotPositiveException e) {
    }
    org.junit.Assert.assertNotNull(chromosomeArray0);
    org.junit.Assert.assertTrue("'" + boolean2 + "' != '" + false + "'", boolean2 == false);
}
```

Mark only one oval.

- testFailsToCreateElitisticListPopulationTaking3ArgumentsThrowsNotPositiveException
- throwExceptionIfPopulationLimitHasNonPositiveValue
- test

36. Please, explain here your answer for the previous question

37. Given a list of candidate test names below, select the test name you think will execute line 89 of ElitisticListPopulation class. (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/ElitisticListPopulation.java). *

Mark only one oval.

- throwExceptionIfAnElitismRateIsSet
- testSetElitismRateThrowsOutOfRangeExceptionAndToString
- test

38. Please, explain here your answer for the previous question

Skip to question 83

ElitisticListPopulation and ListPopulation - Evaluating test names

In the following questions, there are some unit tests referring to the ElitisticListPopulation class (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/ElitisticListPopulation.java), that implements the ListPopulation abstract class (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/ListPopulation.java)

39. Given the unit test below for ElitisticListPopulation class, indicate your level of agreement with the suggested test name "testSetElitismRateThrowsOutOfRangeExceptionAndToString" *

```
@Test
public void test() throws Throwable {
    math.genetics.ElitisticListPopulation elitisticListPopulation2 =
            new math.genetics.ElitisticListPopulation(100,
            0.0d);
    java.lang.String str3 = elitisticListPopulation2.toString();
    try {
        elitisticListPopulation2.setElitismRate((double) (byte) -1);
        org.junit.Assert
                .fail("Expected exception of type math.exception.OutOfRangeException; "
                        + "message: elitism rate (-1)");
    } catch (math.exception.OutOfRangeException e) {
    }
    org.junit.Assert.assertTrue("'" + str3 + "' != '" + "[]" + "'", str3.equals("[]"));
}
```

Mark only one oval.

- ◯ Strongly disagree - This test name is completely inappropriate and undescriptive.
- ◯ Disagree - This test name is somewhat inappropriate and undescriptive.
- ◯ Neutral - Neither agree or disagree with this test name.
- ◯ Agree - This test name is somewhat appropriate and descriptive.
- ◯ Strongly Agree - This test name is completely appropriate and descriptive.

40. Please, explain here your answer for the previous question

41. Given the unit test below for ElitisticListPopulation class, select the most appropriate, descriptive name from names below. *

```
@Test
    public void test() throws Throwable {
        math.genetics.ElitisticListPopulation elitisticListPopulation2 =
                new math.genetics.ElitisticListPopulation(100, 0.0d);
        elitisticListPopulation2.setPopulationLimit((int) '4');
        java.util.Iterator<math.genetics.Chromosome> chromosomeItor5 =
                elitisticListPopulation2.iterator();
        java.lang.Class<?> wildcardClass6 = chromosomeItor5.getClass();
        org.junit.Assert.assertNotNull(chromosomeItor5);
        org.junit.Assert.assertNotNull(wildcardClass6);
    }
```

Mark only one oval.

- ( ) testIterator
- ( ) testIfChromosomeIteratorIsNotNull
- ( ) test

42. Please, explain here your answer for the previous question

43. Given a list of candidate test names below, select the test name you think will execute line 64 of ListPopulation class. (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/ListPopulation.java). *

Mark only one oval.

- ( ) throwExceptionIfPopulationLimitHasNonPositiveValue
- ( ) testFailsToCreateElitisticListPopulationTaking3ArgumentsThrowsNotPositiveException
- ( ) test

44. Please, explain here your answer for the previous question

ComparatorChain - Evaluating test names

In the following questions, there are some unit tests referring to the ComparatorChain class (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/ComparatorChain.java)

45. Given the unit test below for ComparatorChain class, indicate your level of agreement with the suggested test name "testSetComparatorTaking3ArgumentsWithNegativeAndFalse" *

```
@Test
public void test() throws Throwable {
    collections.comparators.ComparatorChain<java.lang.Comparable<java.lang.String>>
    strComparableComparatorChain0 =
    new collections.comparators.ComparatorChain<java.lang.Comparable<java.lang.String>>();
    int int1 = strComparableComparatorChain0.size();
    collections.comparators.ComparatorChain<java.lang.Comparable<java.lang.String>>
    strComparableComparatorChain3 =
    new collections.comparators.ComparatorChain<java.lang.Comparable<java.lang.String>>();
    int int4 = strComparableComparatorChain3.size();
    try {
        strComparableComparatorChain0.setComparator((-1),
                (java.util.Comparator<java.lang.Comparable<java.lang.String>>)
                strComparableComparatorChain3, false);
        org.junit.Assert.fail(""
                + "Expected exception of type java.lang.ArrayIndexOutOfBoundsException; "
                + "message: -1");
    } catch (java.lang.ArrayIndexOutOfBoundsException e) {
    }
    org.junit.Assert.assertTrue("'" + int1 + "' != '" + 0 + "'", int1 == 0);
    org.junit.Assert.assertTrue("'" + int4 + "' != '" + 0 + "'", int4 == 0);
}
```

Mark only one oval.

- ◯ Strongly disagree - This test name is completely inappropriate and undescriptive.
- ◯ Disagree - This test name is somewhat inappropriate and undescriptive.
- ◯ Neutral - Neither agree or disagree with this test name.
- ◯ Agree - This test name is somewhat appropriate and descriptive.
- ◯ Strongly Agree - This test name is completely appropriate and descriptive.

46. Please, explain here your answer for the previous question

47. Given the unit test below for ComparatorChain class, select the most appropriate, descriptive name from names below. *

```
@Test
public void test() throws Throwable {
    collections.comparators.ComparatorChain<java.lang.Comparable<java.lang.String>>
    strComparableComparatorChain0 =
    new collections.comparators.ComparatorChain<java.lang.Comparable<java.lang.String>>();
    int int1 = strComparableComparatorChain0.size();
    collections.comparators.ComparatorChain<java.lang.Comparable<java.lang.String>>
    strComparableComparatorChain3 =
    new collections.comparators.ComparatorChain<java.lang.Comparable<java.lang.String>>();
    collections.comparators.ComparatorChain<java.lang.Comparable<java.lang.String>>
    strComparableComparatorChain4 =
    new collections.comparators.ComparatorChain<java.lang.Comparable<java.lang.String>>();
    strComparableComparatorChain3.
    addComparator((java.util.Comparator<java.lang.Comparable<java.lang.String>>)
            strComparableComparatorChain4);
    collections.comparators.ComparatorChain<java.lang.Comparable<java.lang.String>>
    strComparableComparatorChain7 =
    new collections.comparators.ComparatorChain<java.lang.Comparable<java.lang.String>>
    ((java.util.Comparator<java.lang.Comparable<java.lang.String>>)
            strComparableComparatorChain3, true);
    try {
        strComparableComparatorChain0.setComparator((int) (short) -1,
                (java.util.Comparator<java.lang.Comparable<java.lang.String>>)
                strComparableComparatorChain3);
        org.junit.Assert.fail(""
                + "Expected exception of type java.lang.ArrayIndexOutOfBoundsException; "
                + "message: -1");
    } catch (java.lang.ArrayIndexOutOfBoundsException e) {
    }
    org.junit.Assert.assertTrue("'" + int1 + "' != '" + 0 + "'", int1 == 0);
```

Mark only one oval.

- testSetComparatorTaking2ArgumentsThrowsArrayIndexOutOfBoundsException
- Should_ThrowException_When_SetComparatorInInvalidNegativeIndex
- test

48. Please, explain here your answer for the previous question

49. Given a list of candidate test names below, select the test name you think will execute line 166 of ComparatorChain class. (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/ComparatorChain.java). *

Mark only one oval.

- testSetComparatorTaking3ArgumentsWithPositiveAndTrue
- Should_ThrowException_When_SetComparatorInInvalidIndex
- test

50. Please, explain here your answer for the previous question

Skip to question 91

ComparatorChain - Evaluating test names

In the following questions, there are some unit tests referring to the ComparatorChain class (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/ComparatorChain.java)

51. Given the unit test below for ComparatorChain class, indicate your level of agreement with the suggested test name "testSetComparatorTaking3ArgumentsWithPositiveAndTrue" *

```
@Test
public void test() throws Throwable {
    collections.comparators.ComparatorChain<java.lang.Comparable<java.lang.String>>
    strComparableComparatorChain0 =
    new collections.comparators.ComparatorChain<java.lang.Comparable<java.lang.String>>();
    java.lang.Class<?> wildcardClass1 = strComparableComparatorChain0.getClass();
    java.util.Comparator<java.lang.Comparable<java.lang.String>> strComparableComparator2 =
            strComparableComparatorChain0.reversed();
    int int3 = strComparableComparatorChain0.size();
    collections.comparators.ComparatorChain<java.lang.Comparable<java.lang.String>>
    strComparableComparatorChain5 =
    new collections.comparators.ComparatorChain<java.lang.Comparable<java.lang.String>>();
    java.lang.Class<?> wildcardClass6 = strComparableComparatorChain5.getClass();
    java.util.Comparator<java.lang.Comparable<java.lang.String>> strComparableComparator7 =
            strComparableComparatorChain5.reversed();
    collections.comparators.ComparatorChain<java.lang.Comparable<java.lang.String>>
    strComparableComparatorChain8 =
    new collections.comparators.ComparatorChain<java.lang.Comparable<java.lang.String>>();
    java.lang.Class<?> wildcardClass9 = strComparableComparatorChain8.getClass();
    java.util.Comparator<java.lang.Comparable<java.lang.String>> strComparableComparator10 =
            strComparableComparatorChain8.reversed();
    strComparableComparatorChain5.addComparator(strComparableComparator10, true);
    collections.comparators.ComparatorChain<java.lang.Comparable<java.lang.String>>
    strComparableComparatorChain13 =
    new collections.comparators.
    ComparatorChain<java.lang.Comparable<java.lang.String>>(strComparableComparator10);
    try {
        strComparableComparatorChain0.setComparator(
                (int) (byte) 100, (java.util.Comparator<java.lang.Comparable<java.lang.String>>)
                strComparableComparatorChain13, true);
        org.junit.Assert.fail("Expected exception of type java.lang.IndexOutOfBoundsException; "
                + "message: Index: 100, Size: 0");
    } catch (java.lang.IndexOutOfBoundsException e) {
    }
    org.junit.Assert.assertNotNull(wildcardClass1);
    org.junit.Assert.assertNotNull(strComparableComparator2);
    org.junit.Assert.assertTrue("'" + int3 + "' != '" + 0 + "'", int3 == 0);
    org.junit.Assert.assertNotNull(wildcardClass6);
    org.junit.Assert.assertNotNull(strComparableComparator7);
    org.junit.Assert.assertNotNull(wildcardClass9);
    org.junit.Assert.assertNotNull(strComparableComparator10);
}
```

Mark only one oval.

- ◯ Strongly disagree - This test name is completely inappropriate and undescriptive.
- ◯ Disagree - This test name is somewhat inappropriate and undescriptive.
- ◯ Neutral - Neither agree or disagree with this test name.
- ◯ Agree - This test name is somewhat appropriate and descriptive.
- ◯ Strongly Agree - This test name is completely appropriate and descriptive.

52. Please, explain here your answer for the previous question

53. Given the unit test below for ComparatorChain class, select the most appropriate, descriptive name from names below. *

```
@Test
public void test() throws Throwable {
    collections.comparators.ComparatorChain<java.lang.Comparable<java.lang.String>>
    strComparableComparatorChain0 =
    new collections.comparators.ComparatorChain<java.lang.Comparable<java.lang.String>>();
    int int1 = strComparableComparatorChain0.size();
    collections.comparators.ComparatorChain<java.lang.Comparable<java.lang.String>>
    strComparableComparatorChain3 =
    new collections.comparators.ComparatorChain<java.lang.Comparable<java.lang.String>>();
    int int4 = strComparableComparatorChain3.size();
    try {
        strComparableComparatorChain0.setComparator((-1),
                (java.util.Comparator<java.lang.Comparable<java.lang.String>>)
                strComparableComparatorChain3, false);
        org.junit.Assert.fail(""
                + "Expected exception of type java.lang.ArrayIndexOutOfBoundsException; "
                + "message: -1");
    } catch (java.lang.ArrayIndexOutOfBoundsException e) {
    }
    org.junit.Assert.assertTrue("'" + int1 + "' != '" + 0 + "'", int1 == 0);
    org.junit.Assert.assertTrue("'" + int4 + "' != '" + 0 + "'", int4 == 0);
}
```

Mark only one oval.

- testSetComparatorTaking3ArgumentsWithNegativeAndFalse
- Should_ThrowException_When_SetComparatorInInvalidNegativeIndexAgain
- test

54. Please, explain here your answer for the previous question

55. Given a list of candidate test names below, select the test name you think will execute line 166 of ComparatorChain class. (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/ComparatorChain.java). *

Mark only one oval.

- testSetComparatorTaking2ArgumentsThrowsArrayIndexOutOfBoundsException
- Should_ThrowException_When_SetComparatorInInvalidNegativeIndex
- test

56. Please, explain here your answer for the previous question

ComparatorChain - Evaluating test names

In the following questions, there are some unit tests referring to the ComparatorChain class (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/ComparatorChain.java)

57. Given the unit test below for ComparatorChain class, indicate your level of agreement with the suggested test name "testSetComparatorTaking2ArgumentsThrowsArrayIndexOutOfBoundsException" *

```
@Test
public void test() throws Throwable {
    collections.comparators.ComparatorChain<java.lang.Comparable<java.lang.String>>
    strComparableComparatorChain0 =
    new collections.comparators.ComparatorChain<java.lang.Comparable<java.lang.String>>();
    int int1 = strComparableComparatorChain0.size();
    collections.comparators.ComparatorChain<java.lang.Comparable<java.lang.String>>
    strComparableComparatorChain3 =
    new collections.comparators.ComparatorChain<java.lang.Comparable<java.lang.String>>();
    collections.comparators.ComparatorChain<java.lang.Comparable<java.lang.String>>
    strComparableComparatorChain4 =
    new collections.comparators.ComparatorChain<java.lang.Comparable<java.lang.String>>();
    strComparableComparatorChain3.
    addComparator((java.util.Comparator<java.lang.Comparable<java.lang.String>>)
            strComparableComparatorChain4);
    collections.comparators.ComparatorChain<java.lang.Comparable<java.lang.String>>
    strComparableComparatorChain7 =
    new collections.comparators.ComparatorChain<java.lang.Comparable<java.lang.String>>
    ((java.util.Comparator<java.lang.Comparable<java.lang.String>>)
            strComparableComparatorChain3, true);
    try {
        strComparableComparatorChain0.setComparator((int) (short) -1,
                (java.util.Comparator<java.lang.Comparable<java.lang.String>>)
                strComparableComparatorChain3);
        org.junit.Assert.fail(""
                + "Expected exception of type java.lang.ArrayIndexOutOfBoundsException; "
                + "message: -1");
    } catch (java.lang.ArrayIndexOutOfBoundsException e) {
    }
    org.junit.Assert.assertTrue("'" + int1 + "' != '" + 0 + "'", int1 == 0);
```

Mark only one oval.

- Strongly disagree - This test name is completely inappropriate and undescriptive.
- Disagree - This test name is somewhat inappropriate and undescriptive.
- Neutral - Neither agree or disagree with this test name.
- Agree - This test name is somewhat appropriate and descriptive.
- Strongly Agree - This test name is completely appropriate and descriptive.

58. Please, explain here your answer for the previous question

59. Given the unit test below for ComparatorChain class, select the most appropriate, descriptive name from names below. *

```
@Test
public void test() throws Throwable {
    collections.comparators.ComparatorChain<java.lang.Comparable<java.lang.String>>
    strComparableComparatorChain0 =
    new collections.comparators.ComparatorChain<java.lang.Comparable<java.lang.String>>();
    java.lang.Class<?> wildcardClass1 = strComparableComparatorChain0.getClass();
    java.util.Comparator<java.lang.Comparable<java.lang.String>> strComparableComparator2 =
            strComparableComparatorChain0.reversed();
    int int3 = strComparableComparatorChain0.size();
    collections.comparators.ComparatorChain<java.lang.Comparable<java.lang.String>>
    strComparableComparatorChain5 =
    new collections.comparators.ComparatorChain<java.lang.Comparable<java.lang.String>>();
    java.lang.Class<?> wildcardClass6 = strComparableComparatorChain5.getClass();
    java.util.Comparator<java.lang.Comparable<java.lang.String>> strComparableComparator7 =
            strComparableComparatorChain5.reversed();
    collections.comparators.ComparatorChain<java.lang.Comparable<java.lang.String>>
    strComparableComparatorChain8 =
    new collections.comparators.ComparatorChain<java.lang.Comparable<java.lang.String>>();
    java.lang.Class<?> wildcardClass9 = strComparableComparatorChain8.getClass();
    java.util.Comparator<java.lang.Comparable<java.lang.String>> strComparableComparator10 =
            strComparableComparatorChain8.reversed();
    strComparableComparatorChain5.addComparator(strComparableComparator10, true);
    collections.comparators.ComparatorChain<java.lang.Comparable<java.lang.String>>
    strComparableComparatorChain13 =
    new collections.comparators.
    ComparatorChain<java.lang.Comparable<java.lang.String>>(strComparableComparator10);
    try {
        strComparableComparatorChain0.setComparator(
                (int) (byte) 100, (java.util.Comparator<java.lang.Comparable<java.lang.String>>)
                strComparableComparatorChain13, true);
        org.junit.Assert.fail("Expected exception of type java.lang.IndexOutOfBoundsException; "
                + "message: Index: 100, Size: 0");
    } catch (java.lang.IndexOutOfBoundsException e) {
    }
    org.junit.Assert.assertNotNull(wildcardClass1);
    org.junit.Assert.assertNotNull(strComparableComparator2);
    org.junit.Assert.assertTrue("'" + int3 + "' != '" + 0 + "'", int3 == 0);
    org.junit.Assert.assertNotNull(wildcardClass6);
    org.junit.Assert.assertNotNull(strComparableComparator7);
    org.junit.Assert.assertNotNull(wildcardClass9);
    org.junit.Assert.assertNotNull(strComparableComparator10);
}
```

Mark only one oval.

- testSetComparatorTaking3ArgumentsWithPositiveAndTrue
- Should_ThrowException_When_SetComparatorInInvalidIndex
- test

60. Please, explain here your answer for the previous question

61. Given a list of candidate test names below, select the test name you think will execute line 166 of ComparatorChain class. (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/ComparatorChain.java). *

Mark only one oval.

- testSetComparatorTaking3ArgumentsWithNegativeAndFalse
- Should_ThrowException_When_SetComparatorInInvalidNegativeIndexAgain
- test

62. Please, explain here your answer for the previous question

Skip to question 95

FixedOrderComparator - Evaluating readability

In the following questions, there are some unit tests referring to the FixedOrderComparator class (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/0/FixedOrderComparator.java)

Answer the following questions according to the following unit test code snippets:

A code - Original automatically generated test (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/1/A.java).
B code - Automatically refactored test (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/1/B.java).
C code - Automatically renamed test (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/1/C.java)
D code - Automatically renamed and refactored test (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/1/D.java)

63. In your opinion, what is the most readable code snippet? *

Mark only one oval.

- A code
- B code
- C code
- D code
- All have the same readability

64. Please, explain here your answer for the previous question

65. Which of these code snippets would you prefer to add to your test suite for the FixedOrderComparator class? *

Mark only one oval.

- ◯ A code
- ◯ B code
- ◯ C code
- ◯ D code
- ◯ Other: _____

66. Please, explain here your answer for the previous question

_____

_____

_____

_____

_____

FixedOrderComparator - Evaluating readability

In the following questions, there are some unit tests referring to the FixedOrderComparator class (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/0/FixedOrderComparator.java)

Answer the questions bellow according to the following unit test code snippets:

A code - Original automatically generated test (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/2/A.java).
B code - Automatically refactored test (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/2/B.java).
C code - Automatically renamed test (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/2/C.java)
D code - Automatically renamed and refactored test (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/2/D.java)

67. In your opinion, what is the most readable code snippet? *

Mark only one oval.

- A code
- B code
- C code
- D code
- All have the same readability

68. Please, justify your answer to the previous question

_____

_____

_____

_____

_____

69. Which of these code snippets would you prefer to add to your test suite for the FixedOrderComparator class? *

Mark only one oval.

- A code
- B code
- C code
- D code
- Other: _____

70. Please, justify your answer to the previous question

Skip to question 15

FixedOrderComparator - Evaluating readability

In the following questions, there are some unit tests referring to the FixedOrderComparator class (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/0/FixedOrderComparator.java)

Answer the following questions according to the following unit test code snippets:

A code - Original automatically generated test (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/3/A.java).
B code - Automatically refactored test (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/3/B.java).
C code - Automatically renamed test (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/3/C.java)
D code - Automatically renamed and refactored test (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/3/D.java)

71. In your opinion, what is the most readable code snippet? *

Mark only one oval.

- A code
- B code
- C code
- D code
- All have the same readability

72. Please, explain here your answer for the previous question

73. Which of these code snippets would you prefer to add to your test suite for the FixedOrderComparator class? *

Mark only one oval.

- A code
- B code
- C code
- D code
- Other: _____

74. Please, explain here your answer for the previous question

ElitisticListPopulation and ListPopulation - Evaluating readability

In the following questions, there are some unit tests referring to the ElitisticListPopulation class (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/ElitisticListPopulation.java), that implements the ListPopulation abstract class (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/ListPopulation.java)

Answer the following questions according to the following unit test code snippets:

A code - Original automatically generated test (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/4/A.java).
B code - Automatically refactored test (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/4/B.java).
C code - Automatically renamed test (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/4/C.java)
D code - Automatically renamed and refactored test (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/4/D.java)

75. In your opinion, what is the most readable code snippet? *

Mark only one oval.

- ◯ A code
- ◯ B code
- ◯ C code
- ◯ D code
- ◯ All have the same readability

76. Please, explain here your answer for the previous question

77. Which of these code snippets would you prefer to add to your test suite for the ListPopulation class? *

Mark only one oval.

- A code
- B code
- C code
- D code
- Other: _____

78. Please, explain here your answer for the previous question

_____

_____

_____

_____

_____

Skip to question 27

ElitisticListPopulation and ListPopulation - Evaluating readability

In the following questions, there are some unit tests referring to the ElitisticListPopulation class (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/ElitisticListPopulation.java), that implements the ListPopulation abstract class (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/ListPopulation.java)

Answer the following questions according to the following unit test code snippets:

A code - Original automatically generated test (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/5/A.java).
B code - Automatically refactored test (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/5/B.java).
C code - Automatically renamed test (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/5/C.java)
D code - Automatically renamed and refactored test (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/5/D.java)

79. In your opinion, what is the most readable code snippet? *

Mark only one oval.

- A code
- B code
- C code
- D code
- All have the same readability

80. Please, explain here your answer for the previous question

_____

_____

_____

_____

_____

81. Which of these code snippets would you prefer to add to your test suite for the ListPopulation class? *

Mark only one oval.

- A code
- B code
- C code
- D code
- Other: _____

82. Please, explain here your answer for the previous question

Skip to question 39

ElitisticListPopulation and ListPopulation - Evaluating readability

In the following questions, there are some unit tests referring to the ElitisticListPopulation class (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/ElitisticListPopulation.java), that implements the ListPopulation abstract class (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/ListPopulation.java)

Answer the following questions according to the following unit test code snippets:

A code - Original automatically generated test (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/6/A.java).
B code - Automatically refactored test (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/6/B.java).
C code - Automatically renamed test (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/6/C.java)
D code - Automatically renamed and refactored test (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/6/D.java)

83. In your opinion, what is the most readable code snippet? *

Mark only one oval.

- A code
- B code
- C code
- D code
- All have the same readability

84. Please, explain here your answer for the previous question

85. Which of these code snippets would you prefer to add to your test suite for the ListPopulation class? *

Mark only one oval.

- A code
- B code
- C code
- D code
- Other: _____

86. Please, explain here your answer for the previous question

ComparatorChain - Evaluating readability

In the following questions, there are some unit tests referring to the ComparatorChain class (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/ComparatorChain.java)

Answer the following questions according to the following unit test code snippets:

A code - Original automatically generated test (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/7/A.java).
B code - Automatically refactored test (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/7/B.java).
C code - Automatically renamed test (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/7/C.java)
D code - Automatically renamed and refactored test (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/7/D.java)

87. In your opinion, what is the most readable code snippet? *

Mark only one oval.

- A code
- B code
- C code
- D code
- All have the same readability

88. Please, explain here your answer for the previous question

89. Which of these code snippets would you prefer to add to your test suite for the ComparatorChain class? *

Mark only one oval.

- A code
- B code
- C code
- D code
- Other: _____

90. Please, explain here your answer for the previous question

_____

_____

_____

_____

_____

Skip to question 51

ComparatorChain - Evaluating readability

In the following questions, there are some unit tests referring to the ComparatorChain class (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/ComparatorChain.java)

Answer the following questions according to the following unit test code snippets:

A code - Original automatically generated test (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/8/A.java).
B code - Automatically refactored test (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/8/B.java).
C code - Automatically renamed test (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/8/C.java)
D code - Automatically renamed and refactored test (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/8/D.java)

91. In your opinion, what is the most readable code snippet? *

Mark only one oval.

- A code
- B code
- C code
- D code
- All have the same readability

92. Please, explain here your answer for the previous question

_____

_____

_____

_____

_____

93. Which of these code snippets would you prefer to add to your test suite for the ComparatorChain class? *

Mark only one oval.

- A code
- B code
- C code
- D code
- Other: _____

94. Please, explain here your answer for the previous question

ComparatorChain - Evaluating readability

In the following questions, there are some unit tests referring to the ComparatorChain class (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/ComparatorChain.java)

Answer the following questions according to the following unit test code snippets:

A code - Original automatically generated test - (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/9/A.java).
B code - Automatically refactored test (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/9/B.java).
C code - Automatically renamed test (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/9/C.java)
D code - Automatically renamed and refactored test (https://github.com/WesleyBrenno/unit-tests-readability/blob/main/9/D.java)

95. In your opinion, what is the most readable code snippet? *

Mark only one oval.

- A code
- B code
- C code
- D code
- All have the same readability

96. Please, explain here your answer for the previous question

97. Which of these code snippets would you prefer to add to your test suite for the ComparatorChain class? *

Mark only one oval.

- A code
- B code
- C code
- D code

98. Please, explain here your answer for the previous question