**Logbook**

**Bradley Pratt - Computer Games Programming U1664020314**

**Algorithms Processes and D**

**Week 1-**

**package** intArrays;

**import** java.util.Arrays;

**import** java.util.Collections;

**public** **class** CleverRandomListing **extends** RandomListing {

**public** CleverRandomListing (**int** size) {

**super**(size);

}

**protected** **void** randomise() {

Integer[] copy = **new** Integer[getArray().length]; // Creates a new Array from the SortedListing class

**for** (**int** index = 0; index < copy.length; index++) {

copy[index] = index; // Builds the array until it meets the specified length.

}

Collections.*shuffle*(Arrays.*asList*(copy));[[1]](#footnote-1) // Shuffles the array using a method from java.util.collections

**for** (**int** index = 0; index < copy.length; index++) {

getArray()[index] = copy[index]; // Matches the arrays

}

}

**public** **static** **void** main(String[] args) {

RandomListing count = **new** CleverRandomListing(50); // create a new list, as long as the specified length.

System.***out***.println(Arrays.*toString*(count.getArray())); // prints the array to the console

}

}

**public** **class** CleverRandomListingTest **extends** ListingTest {

**private** **static** **long** *testStart*, *testEnd*;

@BeforeClass

**public** **static** **void** setUpBeforeClass() **throws** Exception {

}

@AfterClass

**public** **static** **void** tearDownAfterClass() **throws** Exception {

}

@Rule **public** TestName testName = **new** TestName();

@Before

**public** **void** setUp() **throws** Exception {

*testStart* = System.*nanoTime*();

}

@After

**public** **void** tearDown() **throws** Exception {

*testEnd* = System.*nanoTime*();

System.***out***.println("Test \"" + testName.getMethodName() + "\" took " + (*testEnd*-*testStart*)/1000 + " microseconds");

}

@Test

**public** **void** testOneSize()[[2]](#footnote-2) {

testSize(1,**new** CleverRandomListing(1));

}

1. This was the most prominent solution I found when looking up how to randomise arrays, I decided to use it since efficiency was the goal and it could solve the issue in one line of code. [↑](#footnote-ref-1)
2. The tests go up to a million, the same as the default test classes. [↑](#footnote-ref-2)