Method being tested: getUser(String username), from DataAccess
Test coverage: decision coverage for both decision statements, and loop coverage for the loop

Test Inputs:

The first decision's "false" branch can be tested with either an empty user cache, or by passing the method a null parameter

The first decision's "true" branch can be tested by passing the method a String, while the user cache is not empty.

The loop's 0-iteration test case is not possible, since at least one User object (meaning 1 iteration) must exist to reach the loop.

The loop's 1-iteration test case can be tested by loading the user cache with a single User object, then passing the method a String.

The loop's 2-iteration test case can be tested by loading the user cache with two Users objects, then passing the method a String.

The loop's many-iteration test case can be tested by loading the user cache with 100 User objects, then passing the method a String.

The second decision's "true" branch is implicitly tested in the above three loop tests, as long as the passed String matches the username of a User object in the user cache.

The second decision's "false" branch is implicitly tested in the 2-iteration and many-iterations tests, as long as the String passed does not match the username of the first User object in the User cache.

<u>The above coverage can be condensed into the following tests:</u>
1. Test the method before loading any users **(Decision 1:false)**
2. Test the method with 1 User object, passing the last User object’s username as a parameter **(Loop 1:1-iteration, Decision 1:true, Decision 2:true)**
3. Test the method with 2 User objects, passing the last User object’s username as a parameter **(Loop 1: 2-iterations, Decision 1:true, Decision 2:false, Decision 2:true)**
4. Test the method with 100 User objects, passing the last User object’s username as a parameter **(Loop 1: many-iterations, Decision 1:true, Decision 2:false, Decision 2:true)**

Tests:

<u>noUsersLoadedTest()</u> - tests the method with no User objects loaded into the user cache
<u>iterateUsersLoopTest()</u> - tests the method with a parameterized number of User objects loaded into the user cache (1, 2, 100)

Test Results:

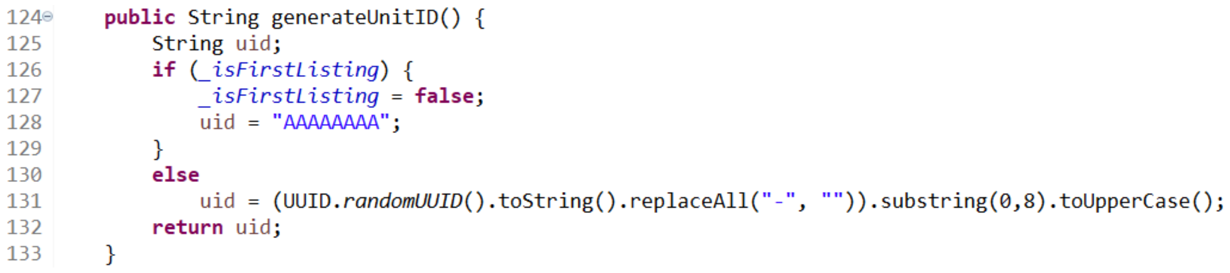
```
│
├─ JUnit Jupiter ✔
│  └─ getUserTest ✔
│     ├─ Decision 1 "false" test ✔
│     └─ Decision 1 "true" test, Loop 1 loop coverage test, Decision 2 "true" and "false"... ✔
│        ├─ [1] 1 ✔
│        ├─ [2] 2 ✔
│        └─ [3] 100 ✔
└─ JUnit Vintage ✔

Test run finished after 124 ms
[         4 containers found      ]
[         0 containers skipped    ]
[         4 containers started    ]
[         0 containers aborted    ]
[         4 containers successful ]
[         0 containers failed     ]
[         4 tests found           ]
[         0 tests skipped         ]
[         4 tests started         ]
[         0 tests aborted         ]
[         4 tests successful      ]
[         0 tests failed          ]
```

Method being tested: generateUnitID(), from Listing
Test coverage: 100% statement coverage of the generateUnitID() for an object flagged as the first listing and an object flagged as a non-first listing.

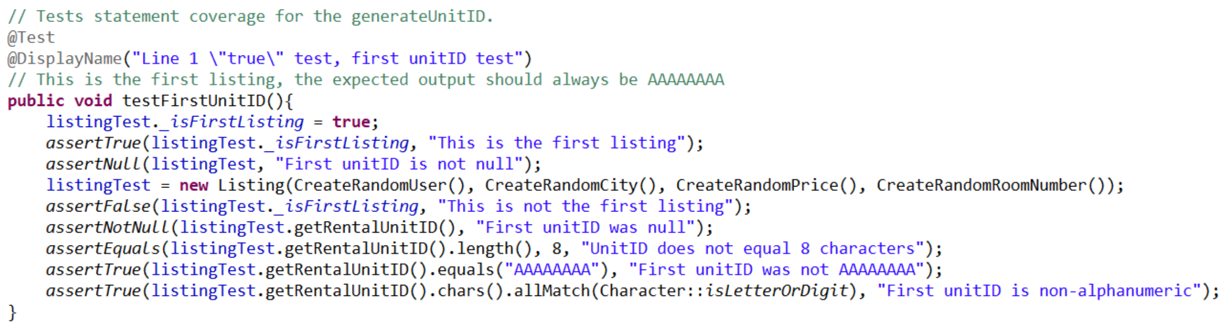
Tests:

For the method being tested we are able to achieve 100% coverage by splitting the method into two tests: A Listing object where the _isFirstListing flag = true, and a Listing object where _isFirst Listing flag = false.



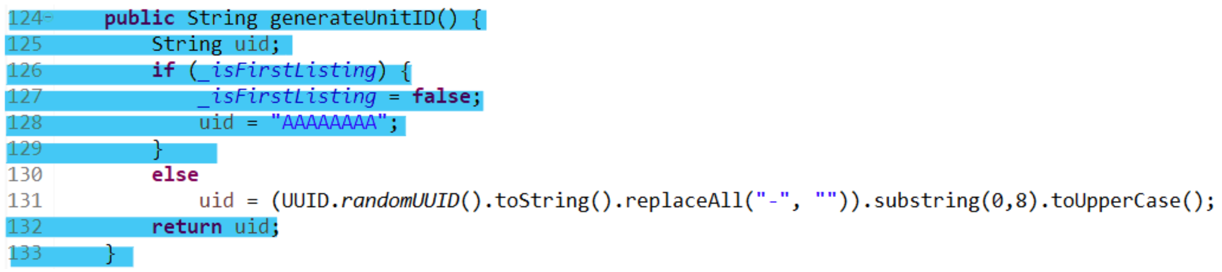
```
    public String generateUnitID() {
        String uid;
        if (_isFirstListing) {
            _isFirstListing = false;
            uid = "AAAAAAAA";
        }
        else
            uid = (UUID.randomUUID().toString().replaceAll("-", "")).substring(0,8).toUpperCase();
        return uid;
    }
```

To test the initial condition, we create a test that contains an object with the “_isFirstListing” = true. When the object is created, the method should set the object’s .isFirstListing attribute to false and set its ID to "AAAAAAAA". We test for the following: was the “_isFirstListing” attribute set to false? Is the ID empty? Is the id string of length 8? Is the id for the first listing equal to "AAAAAAAA"? And finally, is the ID an alphanumeric string?



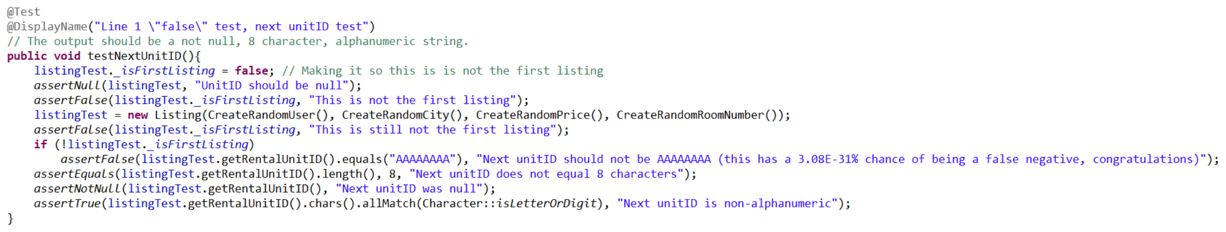
```
// Tests statement coverage for the generateUnitID.
@Test
@DisplayName("Line 1 \"true\" test, first unitID test")
// This is the first listing, the expected output should always be AAAAAAAA
public void testFirstUnitID(){
    listingTest._isFirstListing = true;
    assertTrue(listingTest._isFirstListing, "This is the first listing");
    assertNull(listingTest, "First unitID is not null");
    listingTest = new Listing(CreateRandomUser(), CreateRandomCity(), CreateRandomPrice(), CreateRandomRoomNumber());
    assertFalse(listingTest._isFirstListing, "This is not the first listing");
    assertNotNull(listingTest.getRentalUnitID(), "First unitID was null");
    assertEquals(listingTest.getRentalUnitID().length(), 8, "UnitID does not equal 8 characters");
    assertTrue(listingTest.getRentalUnitID().equals("AAAAAAAA"), "First unitID was not AAAAAAAA");
    assertTrue(listingTest.getRentalUnitID().chars().allMatch(Character::isLetterOrDigit), "First unitID is non-alphanumeric");
}
```

This test covers 5/7 statements.



```
    public String generateUnitID() {
        String uid;
        if (_isFirstListing) {
            _isFirstListing = false;
            uid = "AAAAAAAA";
        }
        else
            uid = (UUID.randomUUID().toString().replaceAll("-", "")).substring(0,8).toUpperCase();
        return uid;
    }
```

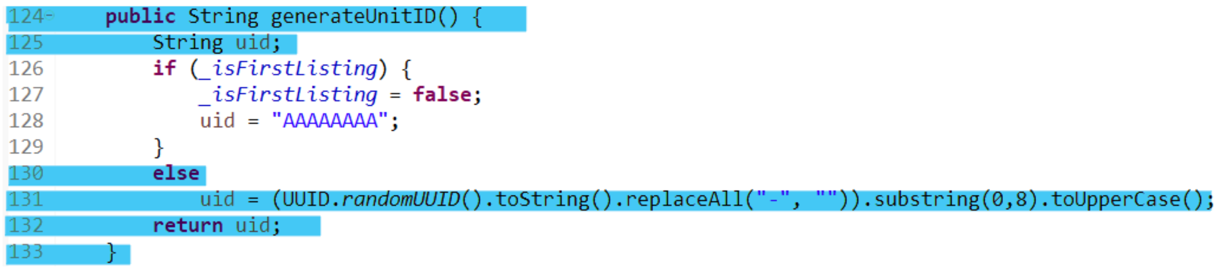
In order to cover the other 2 statements and achieve 100% statement coverage. We also test for an object in which the “_isFirstListing” attribute is set to false.



```java
@Test
@DisplayName("Line 1 \"false\" test, next unitID test")
// The output should be a not null, 8 character, alphanumeric string.
public void testNextUnitID(){
    listingTest._isFirstListing = false; // Making it so this is is not the first listing
    assertNull(listingTest, "UnitID should be null");
    assertFalse(listingTest._isFirstListing, "This is not the first listing");
    listingTest = new Listing(CreateRandomUser(), CreateRandomCity(), CreateRandomPrice(), CreateRandomRoomNumber());
    assertFalse(listingTest._isFirstListing, "This is still not the first listing");
    if (!listingTest._isFirstListing)
        assertFalse(listingTest.getRentalUnitID().equals("AAAAAAAA"), "Next unitID should not be AAAAAAAA (this has a 3.08E-31% chance of being a false negative, congratulations)");
    assertEquals(listingTest.getRentalUnitID().length(), 8, "Next unitID does not equal 8 characters");
    assertNotNull(listingTest.getRentalUnitID(), "Next unitID was null");
    assertTrue(listingTest.getRentalUnitID().chars().allMatch(Character::isLetterOrDigit), "Next unitID is non-alphanumeric");
}
```

To do so, we simply follow the same path as the previous test, but this time we set our object’s “_isFirstListing” attribute to false. We check if this object’s attribute is empty and check if it is the first listing. Then similarly, to the previous test, we create a new listing object which calls on the generateUnitID() method. Just like the previous test, we check for the following: attribute set to false? Is the ID empty? Is the id string of length 8? Is the id for the first listing equal to "AAAAAAAA"? And finally, is the ID an alphanumeric string? We also included a check for the improbable chance that a non-listing ID is set to true.

With this, we are able to cover the remaining 2 statements. Achieving a 100% statement coverage test for the generateUnitID() method.



```java
    public String generateUnitID() {
        String uid;
        if (_isFirstListing) {
            _isFirstListing = false;
            uid = "AAAAAAAA";
        }
        else
            uid = (UUID.randomUUID().toString().replaceAll("-", "")).substring(0,8).toUpperCase();
        return uid;
    }
```

Tests:

<u>testFIrstUnitID()</u> - tests the method to check if it’s generating the correct ID for the first listing.
<u>testNextUnitID()</u> - tests the method to check if it’s generating the correct ID for the succeeding listings.

Test Results:

```
├─ JUnit Jupiter ✔
│  └─ generateUnitIDTest ✔
│     ├─ Decision 2 "true" test, next unitID test ✔
│     └─ Decision 1 "true" test, first unitID test ✔
└─ JUnit Vintage ✔
```

```
Test run finished after 82 ms
[         3 containers found      ]
[         0 containers skipped    ]
[         3 containers started    ]
[         0 containers aborted    ]
[         3 containers successful ]
[         0 containers failed     ]
[         2 tests found           ]
[         0 tests skipped         ]
[         2 tests started         ]
[         0 tests aborted         ]
[         2 tests successful      ]
[         0 tests failed          ]
```