Exercise 9

1 Data-driven testing of a dice game implementation

a)

For K2 (c) we were able to specify which Exception is expected to be thrown and what message it should have.
The corresponding test is named `changeDiceStateWithValidDiceNot1Or5` and runs successfully.

For K3 (a) we didn't know which Exception should be thrown so we asserted that a general Exception should be thrown.
The corresponding test is named `changeDiceStateWhenFixedInPrevRoll` and fails. This is because the code does not throw an exception as is expected by the specification of the test scenario.

To make the second test run successfully as well the implementation of the dicegame needs to be altered. The method c does not differentiate between dice which were fixed in this round and those that were fixed in a previous round. Therefore it allows permanently fixed dice (from previous rounds) to be unfixed. Instead an exception should be thrown, whenever someone tries to alter the state of a permanently fixed dice.

b)

We implemented the test cases K3 (b) and K3 (c).
Our path goes from K1 (b) to K2 (a) or (b) (randomized) to the respective test case of K3.

2 Interface Testing

The created tests can be found in the test module at java/org/jabref/model/search/matchers

The **SearchMatcherTest** is the abstract test class of the SearchMatcher interface.
It is extended by:

- **AndMatcherTest**
- **NotOrMatcherTest**
- **OrMatcherTest**
- **NotAndMatcherTest**
- **NotMatcherTest**