5.1 Task 1: Randoop Code Coverage of NextDate

5.1.1 If you were not present in the lab session, you will first have to complete section 4 above (i.e., using the default parameter settings of Randoop).
In-lab command and NextDate coverage:

```
"C:\Program Files\Java\jdk1.8.0_241\bin\java.exe" -cp
"C:\Users\Tanel\Desktop\randoop\randoop-all-4.2.2.jar;C:\Users\Tanel\Desktop\Tanel\Reposi
tories\tarkvaratest\Kodutoo4\NextDate-2\target\classes" randoop.main.Main gentests
--testclass="ee.ut.cs.swt.nextdate.NextDate"
--junit-output-dir="C:\Users\Tanel\Desktop\Tanel\Repositories\tarkvaratest\Kodutoo4\NextDat
e-2\src\test\java\ee\ut\cs\swt\nextdate"
```

NextDate

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
run(int, int, int)		9%		17%	17	21	39	41	0	1
isLeapYear(int)		0%		0%	4	4	3	3	1	1
isThirtyOneDayMonth(int)		0%		0%	6	6	1	1	1	1
isThirtyDayMonth(int)		0%		0%	5	5	1	1	1	1
isDecember(int)		0%		0%	2	2	1	1	1	1
isFebruary(int)		0%		0%	2	2	1	1	1	1
NextDate(int, int, int)		100%		n/a	0	1	0	2	0	1
Total	202 of 219	7%	61 of 68	10%	36	41	46	50	5	7

5.1.2 In addition to the test suite generated with default Randoop parameters, generate test
suites using the following values for the time-limit parameter: 3, 30, 60. Other parameters keep default values. NB! Take a screenshot of each Randoop command. You will need to submit them! NBB! Details about the reporting can be found in Section 6 (Submission and Grading) below.
time-limit=3 command and NextDate coverage:

```
"C:\Program Files\Java\jdk1.8.0_241\bin\java.exe" -cp
"C:\Users\Tanel\Desktop\randoop\randoop-all-4.2.2.jar;C:\Users\Tanel\Desktop\Tanel\Reposi
tories\tarkvaratest\Kodutoo5\lab5nextdate\target\classes" randoop.main.Main gentests
--testclass="ee.ut.cs.swt.nextdate.NextDate"
--junit-output-dir="C:\Users\Tanel\Desktop\Tanel\Repositories\tarkvaratest\Kodutoo5\lab5nex
tdate\src\test\java\ee\ut\cs\swt\nextdate" --time-limit=3
```

lab5nextdate > ee.ut.cs.swt.nextdate > NextDate

NextDate

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
run(int, int, int)		9%		17%	17	21	39	41	0	1
isLeapYear(int)		0%		0%	4	4	3	3	1	1
isThirtyOneDayMonth(int)		0%		0%	6	6	1	1	1	1
isThirtyDayMonth(int)		0%		0%	5	5	1	1	1	1
isDecember(int)		0%		0%	2	2	1	1	1	1
isFebruary(int)		0%		0%	2	2	1	1	1	1
NextDate(int, int, int)		100%		n/a	0	1	0	2	0	1
Total	202 of 219	7%	61 of 68	10%	36	41	46	50	5	7

time-limit=30 command and NexDate coverage:

```
"C:\Program Files\Java\jdk1.8.0_241\bin\java.exe" -cp
"C:\Users\Tanel\Desktop\randoop\randoop-all-4.2.2.jar;C:\Users\Tanel\Desktop\Tanel\Reposi
tories\tarkvaratest\Kodutoo5\lab5nextdate\target\classes" randoop.main.Main gentests
--testclass="ee.ut.cs.swt.nextdate.NextDate"
--junit-output-dir="C:\Users\Tanel\Desktop\Tanel\Repositories\tarkvaratest\Kodutoo5\lab5nex
tdate\src\test\java\ee\ut\cs\swt\nextdate" --time-limit=30
```

NextDate

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
run(int, int, int)		9%		17%	17	21	39	41	0	1
isLeapYear(int)		0%		0%	4	4	3	3	1	1
isThirtyOneDayMonth(int)		0%		0%	6	6	1	1	1	1
isThirtyDayMonth(int)		0%		0%	5	5	1	1	1	1
isDecember(int)		0%		0%	2	2	1	1	1	1
isFebruary(int)		0%		0%	2	2	1	1	1	1
NextDate(int, int, int)		100%		n/a	0	1	0	2	0	1
Total	202 of 219	7%	61 of 68	10%	36	41	46	50	5	7

time-limit=60 command and NexDate coverage:

```
"C:\Program Files\Java\jdk1.8.0_241\bin\java.exe" -cp
"C:\Users\Tanel\Desktop\randoop\randoop-all-4.2.2.jar;C:\Users\Tanel\Desktop\Tanel\Reposi
tories\tarkvaratest\Kodutoo5\lab5nextdate\target\classes" randoop.main.Main gentests
--testclass="ee.ut.cs.swt.nextdate.NextDate"
--junit-output-dir="C:\Users\Tanel\Desktop\Tanel\Repositories\tarkvaratest\Kodutoo5\lab5nex
tdate\src\test\java\ee\ut\cs\swt\nextdate" --time-limit=60
```

NextDate

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
run(int, int, int)		9%		17%	17	21	39	41	0	1
isLeapYear(int)		0%		0%	4	4	3	3	1	1
isThirtyOneDayMonth(int)		0%		0%	6	6	1	1	1	1
isThirtyDayMonth(int)		0%		0%	5	5	1	1	1	1
isDecember(int)		0%		0%	2	2	1	1	1	1
isFebruary(int)		0%		0%	2	2	1	1	1	1
NextDate(int, int, int)		100%		n/a	0	1	0	2	0	1
Total	202 of 219	7%	61 of 68	10%	36	41	46	50	5	7

5.1.3 Report the results as in the example table at the appendix. You do not need to submit the generated tests, but you must store them somewhere so you can show them on request.

Number of tests generated	Instruction & Branch Coverage	Maven test goal execution time	Additional parameters used*	Randoop log screenshot file name
95	IC 7% BC 10%	6.082 s	--time-limit=3	time3.png
1605	IC 7% BC 10%	5.896 s	--time-limit=30	time30.png
3377	IC 7% BC 10%	6.798 s	--time-limit=60	time60.png

5.1.4 Explain why you got identical coverage results in task 5.1.2.
Randoop by default has only a small set of primitive values from which it chooses test inputs. For example, the primitive int type seed values are:
int: -1, 0, 1, 10, 100.
This is the reason behind the identical coverage results - the pool of values randoop chose from is not very large and most combinations result in an invalid input result.
5.1.5 Improve code coverage by using additional Randoop parameters (hint: literals). Make at least 2 attempts to improve code coverage. NB! You will also need to submit the test suite generated in this subtask!
First literals attempt command and NextDate coverage:

```
"C:\Program Files\Java\jdk1.8.0_241\bin\java.exe" -cp
"C:\Users\Tanel\Desktop\randoop\randoop-all-4.2.2.jar;C:\Users\Tanel\Desktop\Tanel\Reposi
tories\tarkvaratest\Kodutoo5\lab5nextdate\target\classes" randoop.main.Main gentests
--testclass="ee.ut.cs.swt.nextdate.NextDate"
--junit-output-dir="C:\Users\Tanel\Desktop\Tanel\Repositories\tarkvaratest\Kodutoo5\lab5nex
tdate\src\test\java\ee\ut\cs\swt\nextdate" --time-limit=3
--literals-file="C:\Users\Tanel\Desktop\Tanel\Repositories\tarkvaratest\Kodutoo5\lab5nextdat
e\literals1.txt"
```

NextDate

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
run(int, int, int)		35%		30%	16	21	31	41	0	1
isLeapYear(int)		0%		0%	4	4	3	3	1	1
isThirtyDayMonth(int)		56%		25%	4	5	0	1	0	1
isDecember(int)		0%		0%	2	2	1	1	1	1
isFebruary(int)		0%		0%	2	2	1	1	1	1
isThirtyOneDayMonth(int)		89%		50%	5	6	0	1	0	1
NextDate(int, int, int)		100%		n/a	0	1	0	2	0	1
Total	138 of 219	36%	49 of 68	27%	33	41	36	50	3	7

Second literals attempt command and NextDate coverage:

NextDate

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
run(int, int, int)		79%		75%	8	21	11	41	0	1
isLeapYear(int)		0%		0%	4	4	3	3	1	1
isThirtyOneDayMonth(int)		100%		100%	0	6	0	1	0	1
isThirtyDayMonth(int)		100%		100%	0	5	0	1	0	1
isDecember(int)		100%		100%	0	2	0	1	0	1
isFebruary(int)		100%		100%	0	2	0	1	0	1
NextDate(int, int, int)		100%		n/a	0	1	0	2	0	1
Total	50 of 219	77%	16 of 68	76%	12	41	14	50	1	7

5.1.6 Report the attempts in the table (as in 5.1.3).

Number of tests generated	Instruction & Branch Coverage	Maven test goal execution time	Additional parameters used*	Randoop log screenshot file name
57	IC 36% BC 27%	5.451 s	--time-limit=3 --literals-file="C:\Users\Tanel\Desktop\Tanel\Repositories\tarkvaratest\Kodutoo5\lab5nextdate\literals1.txt"	literals1.png
3227	IC 77% BC 76%	10.657 s	--time-limit=60 --literals-file="C:\Users\Tanel\Desktop\Tanel\Repositories\tarkvaratest\Kodutoo5\lab5nextdate\literals1.txt"	literals2.png

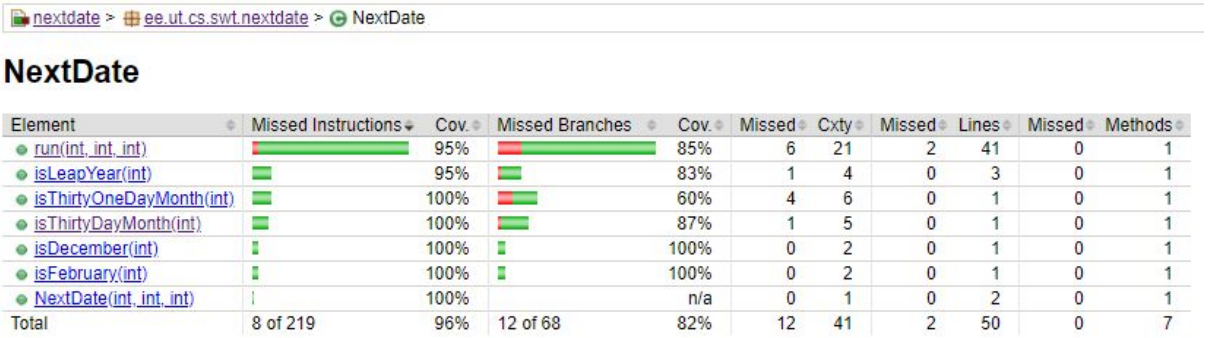
5.1.7 Include the used literals file in the homework submission. You may modify the literals file during testing, but only have to submit the final version.
The literals file is called literals1.txt
5.1.8 Include a screenshot of a Randoop-generated test case with a valid input date in your report.

```
@Test
public void test1778() throws Throwable {
    if (debug)
        System.out.format("%n%s%n", "RegressionTest3.test1778");
    ee.ut.cs.swt.nextdate.NextDate nextDate3 = new ee.ut.cs.swt.nextdate.NextDate((int) (byte) 1,  d: 4,  y: 1);
    java.lang.String str7 = nextDate3.run( month: 8,  day: 7,  year: 2004);
    java.lang.Class<?> wildcardClass8 = nextDate3.getClass();
    // Regression assertion (captures the current behavior of the code)
    org.junit.Assert.assertTrue( message: "'" + str7 + "' != '" + "8/8/2004" + "'", str7.equals("8/8/2004"));
    // Regression assertion (captures the current behavior of the code)
    org.junit.Assert.assertNotNull(wildcardClass8);
}
```

5.1.9 Randoop did not generate error-revealing tests. This could be the case because the program is correct. However, assuming the program was not correct, what could in that case be the reason for not generating an error-revealing test? Report your answer.
It's possible that an error-revealing test is not generated because the possible inputs that randoop uses never reach the branch of code where an error might be thrown. For instance,

say an error would be thrown in a branch where isLeapYear() is true. If the program has no valid leap year input, it could never find that error.

5.1.10 As the last row of the table, for comparison, report your own highest coverage results from Lab 4, where you wrote the tests manually. If you did not do lab 4, you may ask your fellow students for their code coverage report file (a part of the homework of lab 4) on the course Slack channel.



Number of tests generated	Instruction & Branch Coverage	Maven test goal execution time	Additional parameters used*	Randoop log screenshot file name
18	IC 96% BC 82%	3.534 s	None	None

Task 2: Randoop Code Coverage of POS

5.2.2

```
C:\Users\Kaarel>"C:\Program Files\Java\jdk1.8.0_241\bin\java.exe" -cp "C:\Users\Kaarel\Desktop\randoop-all-4.2.2.jar;
C:\Users\Kaarel\Desktop\lab5pos_180320\target\classes" randoop.main.Main gentests --testclass="ee.ut.math.tvt.salessy
stem.logic.ShoppingCart" --junit-output-dir="C:\Users\Kaarel\Desktop\lab5pos_180320\src\test\java\ee\ut\math\tvt\sale
ssystem" --npe-on-null-input=ERROR
```

5.2.3

Out of total 1163 tests, 387 were error-revealing. Analyzing these tests, it seems that the source of errors is the scenario where the submitCurrentPurchase method is called on a shopping cart that is created using an uninitialized salesSystemDAO object. This is evident because all of the error-revealing tests have the following lines:

```
// during test generation this statement threw an exception of type
java.lang.NullPointerException in error
shoppingCart1.submitCurrentPurchase();
```

Therefore, the error is caused by the following method:

```
public void submitCurrentPurchase() {
    // TOD decrease quantities of the warehouse stock

    // note the use of transactions. InMemorySalesSystemDAO ignores transactions
    // but when you start using hibernate in lab5, then it will become relevant.
    // what is a transaction? https://stackoverflow.com/q/974596
    dao.beginTransaction();
    try {
        for (SoldItem item : items) {
            dao.saveSoldItem(item);
        }
        dao.commitTransaction();
        items.clear();
    } catch (Exception e) {
        dao.rollbackTransaction();
        throw e;
    }
}
```

What happens is that items with “null” value are added to the shopping cart and if submitCurrentPurchase method is called, the method tries to iterate over items and save them, but since the items have “null” value, an exception is thrown.

5.2.4

One of the error-revealing tests:

```java
@Test
public void test001() throws Throwable {
    if (debug)
        System.out.format("%n%s%n", "ErrorTest0.test001");
    ee.ut.math.tvt.salessystem.dao.SalesSystemDAO salesSystemDAO0 = null;
    ee.ut.math.tvt.salessystem.logic.ShoppingCart shoppingCart1 = new ee.ut.math.tvt.salessystem.logic.ShoppingCart(salesSystemDAO0);
    shoppingCart1.cancelCurrentPurchase();
    shoppingCart1.cancelCurrentPurchase();
    // during test generation this statement threw an exception of type java.lang.NullPointerException in error
    shoppingCart1.submitCurrentPurchase();
}
```

5.2.6

After uncommenting the code block, there is a new error that happens when adding an item with “null” value to the shopping cart and then adding another item that also has “null” value to the cart. This bug is caused by the now-uncommented part of the code which includes the line: `if (existingItem.getId().equals(item.getId()))` What happens there is that after adding the first item, there is an element with “null” value in the shopping cart. After adding the second item, the aforementioned line tries to compare the first and second “null” values by id, but since neither of them has id, NullPointerException is thrown. This error wasn’t thrown before because the line along with the rest of the for-loop was commented out and thus the checking for equality wasn’t done. Below is one of the now-failing tests:

```java
@Test
public void test503() throws Throwable {
    if (debug)
        System.out.format("%n%s%n", "RegressionTest1.test503");
    ee.ut.math.tvt.salessystem.dao.SalesSystemDAO salesSystemDAO0 = null;
    ee.ut.math.tvt.salessystem.logic.ShoppingCart shoppingCart1 = new ee.ut.math.tvt.salessystem.logic.ShoppingCart(salesSystemDAO0);
    java.util.List<ee.ut.math.tvt.salessystem.dataobjects.SoldItem> soldItemList2 = shoppingCart1.getAll();
    ee.ut.math.tvt.salessystem.dataobjects.SoldItem soldItem3 = null;
    shoppingCart1.addItem(soldItem3);
    shoppingCart1.cancelCurrentPurchase();
    ee.ut.math.tvt.salessystem.dataobjects.SoldItem soldItem6 = null;
    shoppingCart1.addItem(soldItem6);
    ee.ut.math.tvt.salessystem.dataobjects.SoldItem soldItem8 = null;
    shoppingCart1.addItem(soldItem8);
    ee.ut.math.tvt.salessystem.dataobjects.SoldItem soldItem10 = null;
    shoppingCart1.addItem(soldItem10);
    shoppingCart1.cancelCurrentPurchase();
    shoppingCart1.cancelCurrentPurchase();
    ee.ut.math.tvt.salessystem.dataobjects.SoldItem soldItem14 = null;
    shoppingCart1.addItem(soldItem14);
    java.lang.Class<?> wildcardClass16 = shoppingCart1.getClass();
    // Regression assertion (captures the current behavior of the code)
    org.junit.Assert.assertNotNull(soldItemList2);
    // Regression assertion (captures the current behavior of the code)
    org.junit.Assert.assertNotNull(wildcardClass16);
}
```

What happens there is that soldItem6 = null is added to the cart and then soldItem0 = null is also added, but when adding soldItem8, the aforementioned equality check is called and since the items cannot be compared, an error is thrown and the test fails (error happens on the line `shoppingCart1.addItem(soldItem8);`).

5.2.7

The Randoop command is the same as before:

```
C:\Users\Kaarel>"C:\Program Files\Java\jdk1.8.0_241\bin\java.exe" -cp "C:\Users\Kaarel\Desktop\randoop-all-4.2.2.jar;
C:\Users\Kaarel\Desktop\lab5pos_180320\target\classes" randoop.main.Main gentests --testclass="ee.ut.math.tvt.salessy
stem.logic.ShoppingCart" --junit-output-dir="C:\Users\Kaarel\Desktop\lab5pos_180320\src\test\java\ee\ut\math\tvt\sale
ssystem" --npe-on-null-input=ERROR
```

Below is a test case from the new test suite that contains similar steps to the test in 5.2.6, meaning that the program tries to add two items to the cart both of which have the value "null":

```
@Test
public void test017() throws Throwable {
    if (debug)
        System.out.format("%n%s%n", "ErrorTest0.test017");
    ee.ut.math.tvt.salessystem.dao.SalesSystemDAO salesSystemDAO0 = null;
    ee.ut.math.tvt.salessystem.logic.ShoppingCart shoppingCart1 = new ee.ut.math.tvt.salessystem.logic.ShoppingCart(salesSystemDAO0);
    shoppingCart1.cancelCurrentPurchase();
    ee.ut.math.tvt.salessystem.dataobjects.SoldItem soldItem3 = null;
    shoppingCart1.addItem(soldItem3);
    java.util.List<ee.ut.math.tvt.salessystem.dataobjects.SoldItem> soldItemList5 = shoppingCart1.getAll();
    java.util.List<ee.ut.math.tvt.salessystem.dataobjects.SoldItem> soldItemList6 = shoppingCart1.getAll();
    ee.ut.math.tvt.salessystem.dataobjects.SoldItem soldItem7 = null;
    // during test generation this statement threw an exception of type java.lang.NullPointerException in error
    shoppingCart1.addItem(soldItem7);
}
```

Randoop handles errors differently for these two cases because in the 5.2.6 test, it was not able to predict that an error will be thrown there since a part of code was commented in later and wasn't taken into account when the test suite was created. However for the 5.2.7 test, Randoop was able to take that part of the code into account when creating the tests and was thus able to mark it accordingly.

5.2.8

To classify tests as error-revealing, the following Randoop command should be used with --checked-exception=ERROR:

```
C:\Users\Kaarel>"C:\Program Files\Java\jdk1.8.0_241\bin\java.exe" -cp "C:\Users\Kaarel\Desktop\randoop-all-4.2.2.jar;
C:\Users\Kaarel\Desktop\lab5pos_180320\target\classes" randoop.main.Main gentests --testclass="ee.ut.math.tvt.salessy
stem.logic.ShoppingCart" --junit-output-dir="C:\Users\Kaarel\Desktop\lab5pos_180320\src\test\java\ee\ut\math\tvt\sale
ssystem" --npe-on-null-input=ERROR --checked-exception=ERROR
```

By default, Randoop uses --checked-exception=EXPECTED and classifies these tests that would be error-revealing as regression tests instead. For example, the following test case is now error-revealing:

```
@Test
public void test006() throws Throwable {
    if (debug)
        System.out.format("%n%s%n", "ErrorTest0.test006");
    ee.ut.math.tvt.salessystem.dao.SalesSystemDAO salesSystemDAO0 = null;
    ee.ut.math.tvt.salessystem.logic.ShoppingCart shoppingCart1 = new ee.ut.math.tvt.salessystem.logic.ShoppingCart(salesSystemDAO0);
    shoppingCart1.cancelCurrentPurchase();
    ee.ut.math.tvt.salessystem.dataobjects.SoldItem soldItem3 = null;
    shoppingCart1.addItem(soldItem3);
    ee.ut.math.tvt.salessystem.dataobjects.SoldItem soldItem5 = null;
    // during test generation this statement threw an exception of type java.lang.NullPointerException in error
    shoppingCart1.addItem(soldItem5);
}
```