SIT707 Software Quality and Testing

Pass Task: (a) Evidence learning for the topic, (b) Boundary value testing using JUnit

(a) Evidence learning for the topic

You need to demonstrate and provide evidence of your understanding of the topic this week. In this week **Boundary value analysis** and **Equivalence class testing** have been introduced briefly. These concepts are essential to design test cases involving one or more variables. A common example is the next date calculator for a given day/month/year contains 3 variables – day, month and year. Boundary value analysis and Equivalence class testing adopts different approaches in designing test cases which is important to learn this week – it is time for you to reflect on these 2 testing strategies based on the content provided on the unit site as well as in the active learning session.

Submission details

For this task, you will need to go through the weekly learning materials in the unit site and participate in activities during the weekly active learning session.
Submit a pdf combining below items -

- Summary of learning in few slides of each topic item
- Evidence of activities from the active learning session.

You want to focus on the following key ideas, and make sure you can explain them in relation to your submission.

- Boundary value analysis and Equivalence class testing
- Reflect how these test methods can be useful in real world scenarios.

Instructions

For this task, you will need to

1. Go through each topic presented in the unit site for this week including Boundary value analysis, and Equivalence class testing.
2. Create a power-point presentation and summarise learning of each topic into a few slides by including mind-maps, examples, use-cases or any other evidence as suitable.
3. Use the same power-point file to add other topics of the current week as required.
4. Append the same power-point with learning evidence from the active learning session.

(b) Boundary value testing using JUnit

As you now have some understanding of how to write test cases using JUnit, in this task you will write some test cases based on a concept introduced this week – **Boundary value testing**. Given a date of day/month/year, you need to find the next and previous dates of the given date. You will need to use JUnit to write test cases.

Submission details

Use the instructions on the following page to carry out this task's steps.

For this task you will need to study the existing project task3_2P.zip which has DateUtil.java file where DateUtil class represents date information consisting of 3 variables - day, month and year. The value range of **day: 1<=Day<=31**, **month: 1<=Month<=12**, and **year: 1700<=Year<=2024**. DateUtil class has member functions to increment or decrement 1 day from the given day. There is a Main.java file which shows how to create instances of DateUtil and do increment or decrement operations to demonstrate the boundary values for January month at the min and max edges such as 1st and 31st January. You can run Main.java file in Eclipse (*Run As > Java Application*) and understand the output. There is a test file DateUtilTest.java in test package (*Run As > JUnit Test*) which shows test cases for the month of January as a guide on how you should create test cases for the rest 11 months of a choice of year.

Submit a pdf combining below items -

- A screenshot of your Eclipse IDE's (i) JUnit tab which shows test statistics including Runs, Errors and Failures and (ii) Eclipse console which shows outputs.
- Your program's source code for tests (DateUtilTest.java)
- A screenshot of your GitHub page where your latest project folder is pushed.

You want to focus on the following key ideas, and make sure you can explain them in relation to your program.

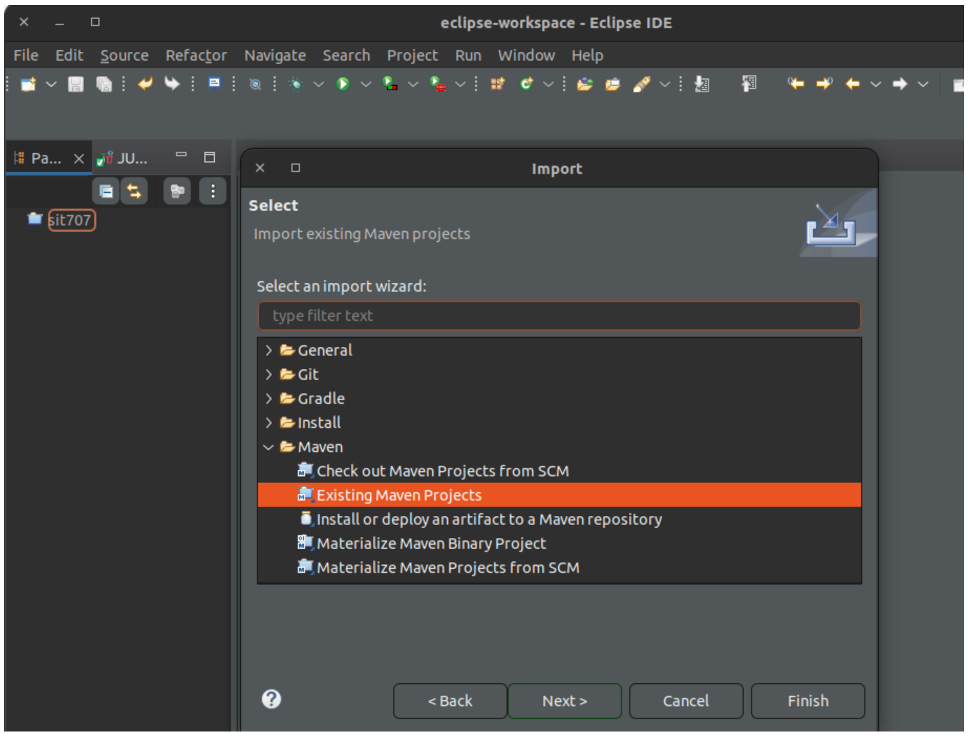
- Boundary value testing
- Unit test case creation using JUnit.

Instructions

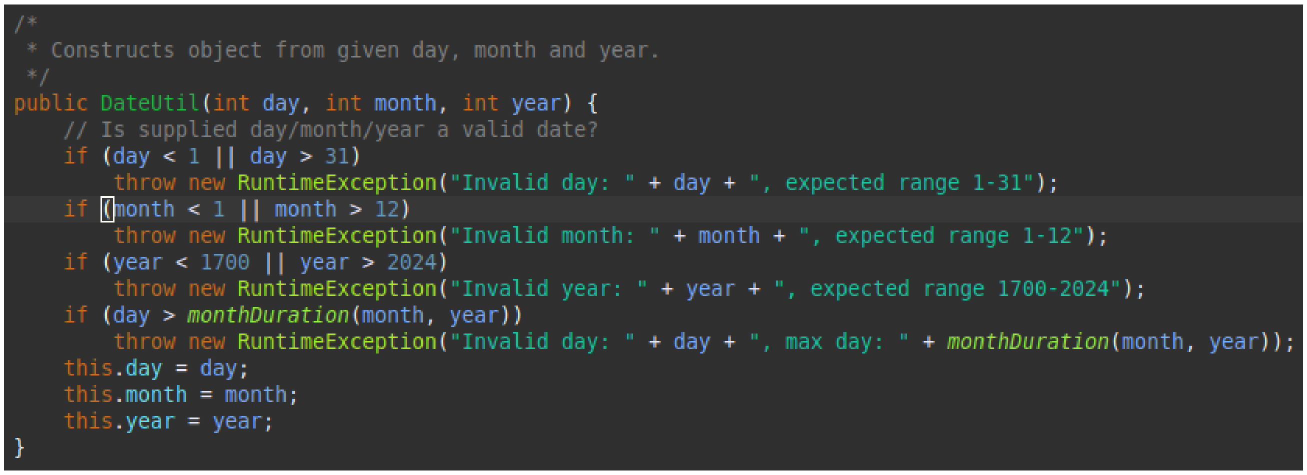
For this task you will need to

1. Download task3_1P.zip Java project and unzip it in a common folder (say, java_projects) which you will be using to store all the weekly projects.

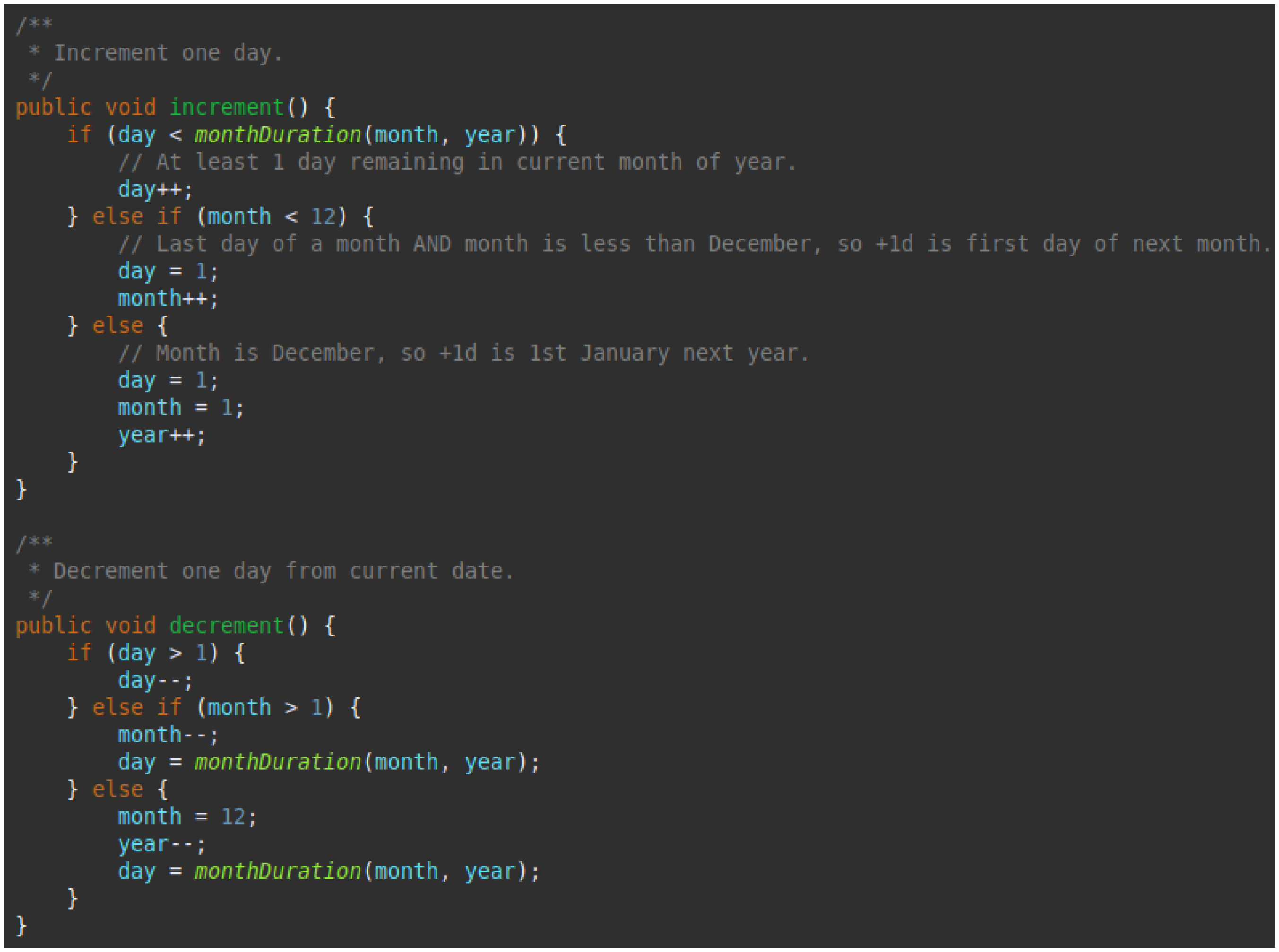
2. Import the project (as a *maven* project) in Eclipse IDE


eclipse-workspace - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Pa... JU...
sit707
Import
Select
Import existing Maven projects
Select an import wizard:
type filter text
General
Git
Gradle
Install
Maven
Check out Maven Projects from SCM
Existing Maven Projects
Install or deploy an artifact to a Maven repository
Materialize Maven Binary Project
Materialize Maven Projects from SCM
< Back
Next >
Cancel
Finish

3. Observe DateUtil class structure in DateUtil.java file which contains a constructor and other functions such as increment and decrement.

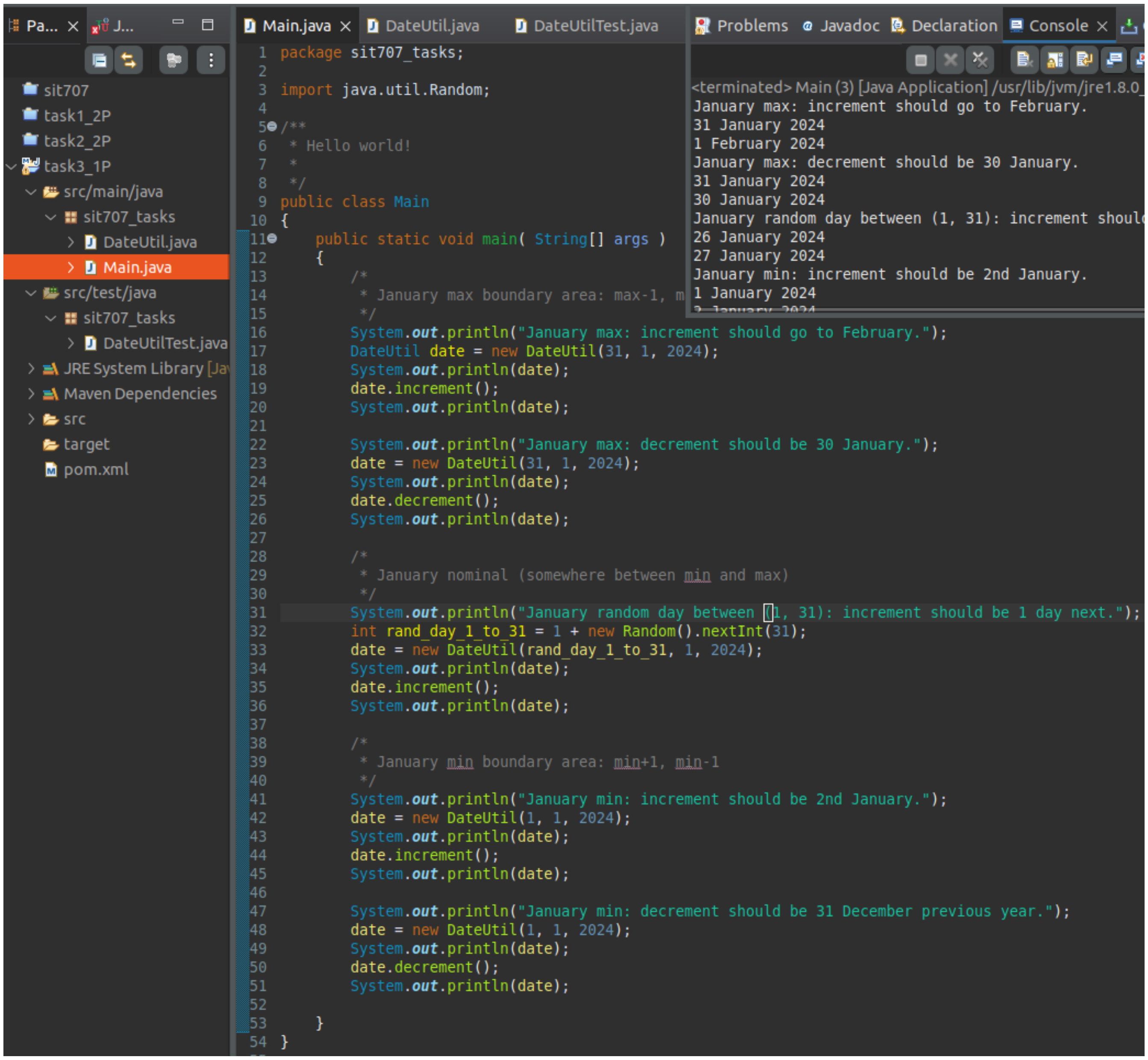


```java
/*
 * Constructs object from given day, month and year.
 */
public DateUtil(int day, int month, int year) {
    // Is supplied day/month/year a valid date?
    if (day < 1 || day > 31)
        throw new RuntimeException("Invalid day: " + day + ", expected range 1-31");
    if (month < 1 || month > 12)
        throw new RuntimeException("Invalid month: " + month + ", expected range 1-12");
    if (year < 1700 || year > 2024)
        throw new RuntimeException("Invalid year: " + year + ", expected range 1700-2024");
    if (day > monthDuration(month, year))
        throw new RuntimeException("Invalid day: " + day + ", max day: " + monthDuration(month, year));
    this.day = day;
    this.month = month;
    this.year = year;
}
```



```java
/**
 * Increment one day.
 */
public void increment() {
    if (day < monthDuration(month, year)) {
        // At least 1 day remaining in current month of year.
        day++;
    } else if (month < 12) {
        // Last day of a month AND month is less than December, so +1d is first day of next month.
        day = 1;
        month++;
    } else {
        // Month is December, so +1d is 1st January next year.
        day = 1;
        month = 1;
        year++;
    }
}

/**
 * Decrement one day from current date.
 */
public void decrement() {
    if (day > 1) {
        day--;
    } else if (month > 1) {
        month--;
        day = monthDuration(month, year);
    } else {
        month = 12;
        year--;
        day = monthDuration(month, year);
    }
}
```

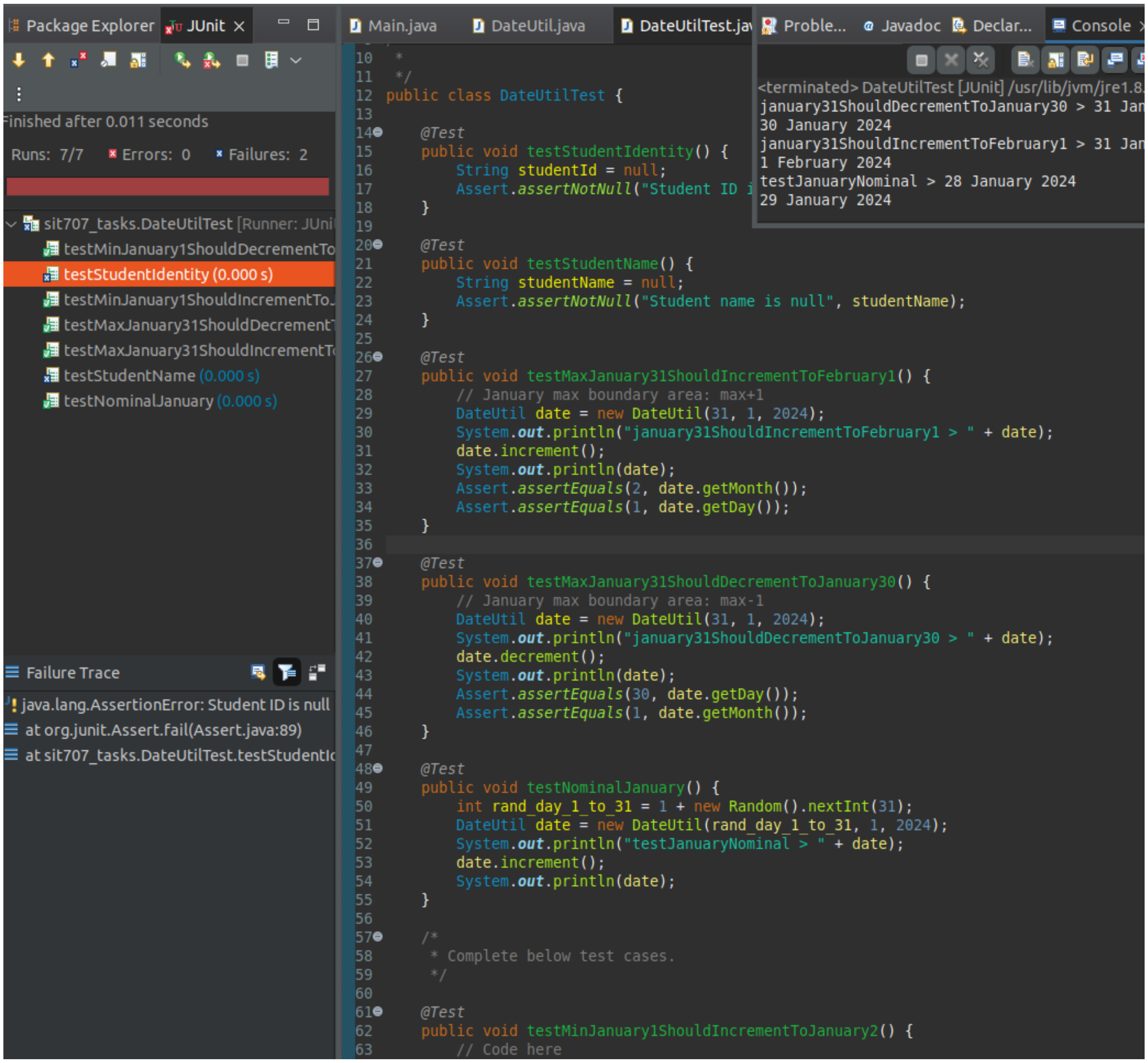
4. Main.java file shows how to construct a DateUtil object at the min or max edges of a month and use member functions (such as increment, decrement) to observe how the DateUtil class progresses back or forth by 1 day. You can run Main.java (*Run As > Java Application*) and observe the output to understand how to create test cases.



```
package sit707_tasks;

import java.util.Random;

/**
 * Hello world!
 *
 */
public class Main
{
    public static void main( String[] args )
    {
        /*
         * January max boundary area: max-1, m
         */
        System.out.println("January max: increment should go to February.");
        DateUtil date = new DateUtil(31, 1, 2024);
        System.out.println(date);
        date.increment();
        System.out.println(date);

        System.out.println("January max: decrement should be 30 January.");
        date = new DateUtil(31, 1, 2024);
        System.out.println(date);
        date.decrement();
        System.out.println(date);

        /*
         * January nominal (somewhere between min and max)
         */
        System.out.println("January random day between (1, 31): increment should be 1 day next.");
        int rand_day_1_to_31 = 1 + new Random().nextInt(31);
        date = new DateUtil(rand_day_1_to_31, 1, 2024);
        System.out.println(date);
        date.increment();
        System.out.println(date);

        /*
         * January min boundary area: min+1, min-1
         */
        System.out.println("January min: increment should be 2nd January.");
        date = new DateUtil(1, 1, 2024);
        System.out.println(date);
        date.increment();
        System.out.println(date);

        System.out.println("January min: decrement should be 31 December previous year.");
        date = new DateUtil(1, 1, 2024);
        System.out.println(date);
        date.decrement();
        System.out.println(date);

    }
}
```

```
<terminated> Main (3) [Java Application] /usr/lib/jvm/jre1.8.0_
January max: increment should go to February.
31 January 2024
1 February 2024
January max: decrement should be 30 January.
31 January 2024
30 January 2024
January random day between (1, 31): increment shoul
26 January 2024
27 January 2024
January min: increment should be 2nd January.
1 January 2024
2 January 2024
```

5. DateUtilTest class in test package contains test cases for DateUtil class to test boundary values for January 2024. Run the test file (*Run As > JUnit Test*) to see the test statistics and the console output. Two test cases to test the minimum range of January are created with empty functions where you need to fill in the code with proper logic.



```
Finished after 0.011 seconds
Runs: 7/7    Errors: 0    Failures: 2

sit707_tasks.DateUtilTest [Runner: JUni
    testMinJanuary1ShouldDecrementTo
    testStudentIdentity (0.000 s)
    testMinJanuary1ShouldIncrementTo
    testMaxJanuary31ShouldDecrement
    testMaxJanuary31ShouldIncrementT
    testStudentName (0.000 s)
    testNominalJanuary (0.000 s)

Failure Trace
java.lang.AssertionError: Student ID is null
at org.junit.Assert.fail(Assert.java:89)
at sit707_tasks.DateUtilTest.testStudentIc
```

```java
 *
 */
public class DateUtilTest {

    @Test
    public void testStudentIdentity() {
        String studentId = null;
        Assert.assertNotNull("Student ID i
    }

    @Test
    public void testStudentName() {
        String studentName = null;
        Assert.assertNotNull("Student name is null", studentName);
    }

    @Test
    public void testMaxJanuary31ShouldIncrementToFebruary1() {
        // January max boundary area: max+1
        DateUtil date = new DateUtil(31, 1, 2024);
        System.out.println("january31ShouldIncrementToFebruary1 > " + date);
        date.increment();
        System.out.println(date);
        Assert.assertEquals(2, date.getMonth());
        Assert.assertEquals(1, date.getDay());
    }

    @Test
    public void testMaxJanuary31ShouldDecrementToJanuary30() {
        // January max boundary area: max-1
        DateUtil date = new DateUtil(31, 1, 2024);
        System.out.println("january31ShouldDecrementToJanuary30 > " + date);
        date.decrement();
        System.out.println(date);
        Assert.assertEquals(30, date.getDay());
        Assert.assertEquals(1, date.getMonth());
    }

    @Test
    public void testNominalJanuary() {
        int rand_day_1_to_31 = 1 + new Random().nextInt(31);
        DateUtil date = new DateUtil(rand_day_1_to_31, 1, 2024);
        System.out.println("testJanuaryNominal > " + date);
        date.increment();
        System.out.println(date);
    }

    /*
     * Complete below test cases.
     */

    @Test
    public void testMinJanuary1ShouldIncrementToJanuary2() {
        // Code here
```

```
<terminated> DateUtilTest [JUnit] /usr/lib/jvm/jre1.8.
january31ShouldDecrementToJanuary30 > 31 Jan
30 January 2024
january31ShouldIncrementToFebruary1 > 31 Jan
1 February 2024
testJanuaryNominal > 28 January 2024
29 January 2024
```

6. There are 2 test failures that you need to correct by providing your name and id in the first 2 test cases.
7. Possible test cases can be listed in the table below. You will need to populate the 2nd table (orange color header, calculates next date) like the 1st table (green color header, calculates previous date) with missing test cases.

Test case ID	Day	Month	Year	Expected Previous
1A	1	6	1994	31-5-1994

2A	2	6	1994	1-6-1994
3A	15	6	1994	14-6-1994
4A	30	6	1994	29-6-1994
5A	31	6	1994	Invalid Date
6A	15	1	1994	14-1-1994
7A	15	2	1994	14-2-1994
8A	15	11	1994	14-11-1994
9A	15	12	1994	14-12-1994
10A	15	6	1700	14-6-1700
11A	15	6	1701	14-6-1701
12A	15	6	2023	14-6-2023
13A	15	6	2024	14-6-2024

Test case ID	Day	Month	Year	Expected Next
1B	1	6	1994	2-6-1994
2B	2	6	1994	3-6-1994
13B	15	6	2024	16-6-2024

8. **Extra test case:** create additional test cases for February with a leap year condition.
9. Run the test, take screenshot of test statistic.
10. Upload your folder to your GitHub account and take a screenshot.

Your Task

Your task is to:

1. Study Main.java to understand how to use DateUtil to see behavior of calendar at the edges for the month of January 2024.
2. Create test cases as shown in the tables above by analysing the boundary values of day/month/year ranges.
3. DateUtilTest.java contains sample test cases for January, the logic taken from the Main.java file.
4. You need to program test cases according to the cases you prepared in the tables. You can look in to provided test codes as done partially for January in provided zip file.

Submit your work

When you are ready, login to OnTrack and submit your pdf which consolidates all the items mentioned in the submission detail section above. Remember to save and backup your work.

Complete your work

After your submission, your OnTrack reviewer (tutor) will review your submission and give you feedback in about 5 business days. Your reviewer may further ask you some questions on the weekly topics and/or about your submissions. You are required to address your OnTrack reviewer's questions as a form of task discussions. Please frequently login to OnTrack for the task ***Discuss/Demonstrate*** or ***Resubmit*** equivalent to fix your work (if needed) based on the feedback to get your task signed as ***Complete***.