Test Design Assignment: NextDate test case design

SOFE 3980U - Software Quality

Akshat Kapoor: 100781511

CRN: 73385

1. Test Case table:

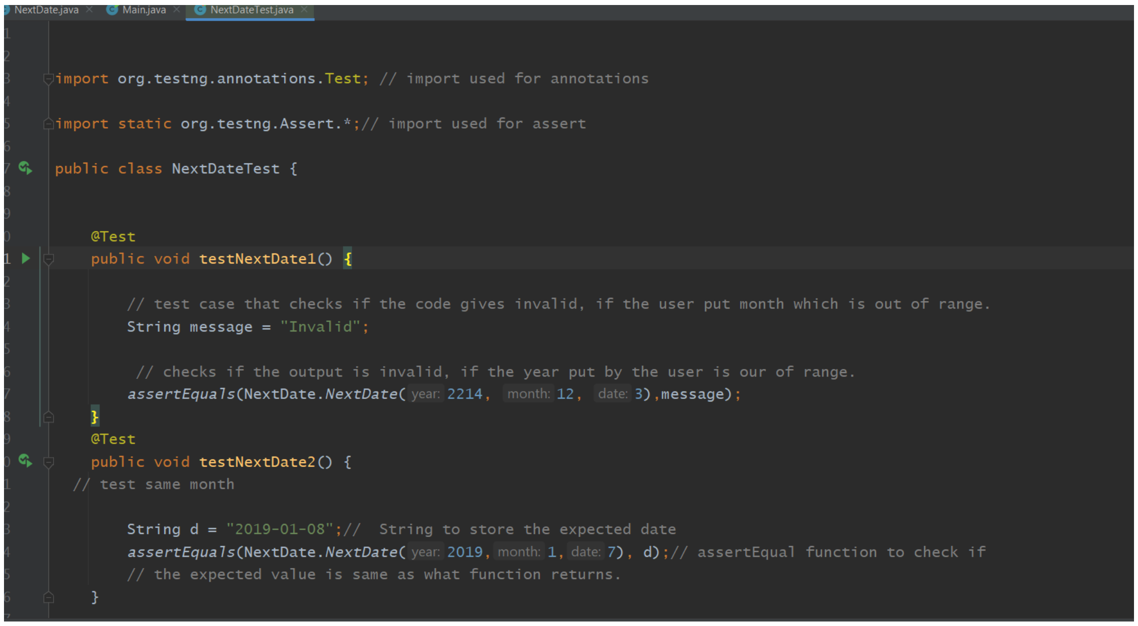
Test Number	Input (yyyy-mm-DD)	Expected outcome (yyyy-mm-DD)
1.	2214-12-03	Invalid, the year you put is out of range
2.	2019-01-07	2019-01-08
3.	2018-02-28	2018-03-01
4.	2024-02-29	2024-03-01
5.	2014-12-31	2015-01-01
6.	2022-02-29	Invalid, year is not a leap year

Input, domain, characteristics table:

Input	domain	Block	Values	Characteristics
Year	1812<=year<=2212	1812>Year Year > 2212 Year >= 1812 Year<=2212	Invalid Invalid Valid Valid	Year of the date that use puts It should be between the following range 1812<=year<=2212
Month	1<=Month<=12	Month < 1 Month > 12 Month >=1 Month <= 12	Invalid Invalid Valid Valid	Month of the date that user inputs, it should be in following range 1<=Month<=12

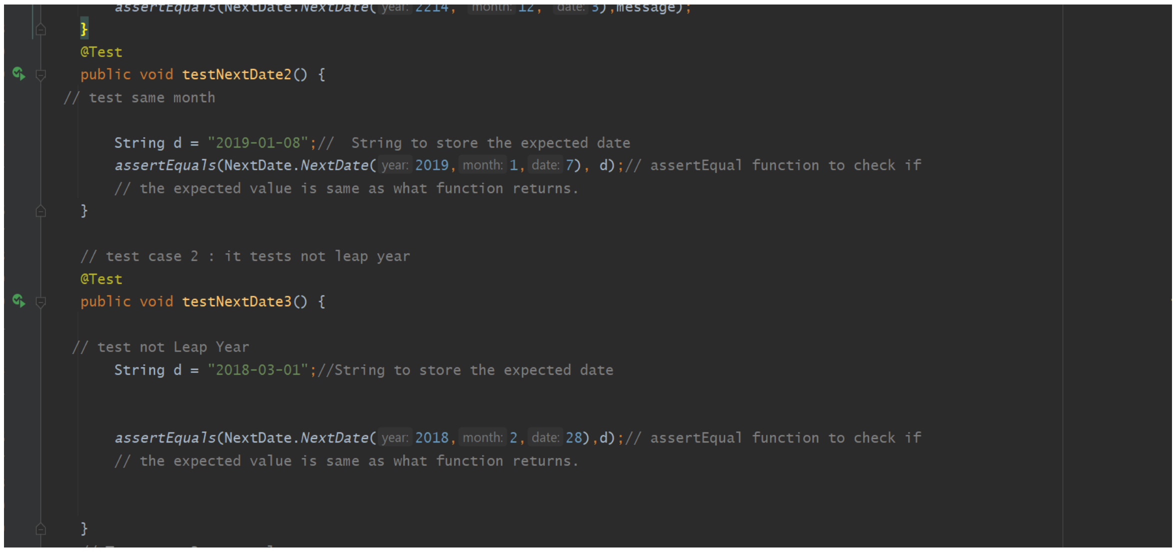
Date	1<=date<=31	date < 1	Invalid	Date represents the day of the date that user inputs and it should be between 1<=date<=31
		date > 31	Invalid	
		date>= one	Valid	
		date<=31	Valid	

2.write a Junit test script (This is a chance to experiment with different Junit annotations to organize your test cases)



```java
import org.testng.annotations.Test; // import used for annotations

import static org.testng.Assert.*;// import used for assert

public class NextDateTest {


    @Test
    public void testNextDate1() {

        // test case that checks if the code gives invalid, if the user put month which is out of range.
        String message = "Invalid";

         // checks if the output is invalid, if the year put by the user is our of range.
        assertEquals(NextDate.NextDate( year: 2214,  month: 12,  date: 3),message);
    }
    @Test
    public void testNextDate2() {
  // test same month

        String d = "2019-01-08";//  String to store the expected date
        assertEquals(NextDate.NextDate( year: 2019, month: 1, date: 7), d);// assertEqual function to check if
        // the expected value is same as what function returns.
    }
```

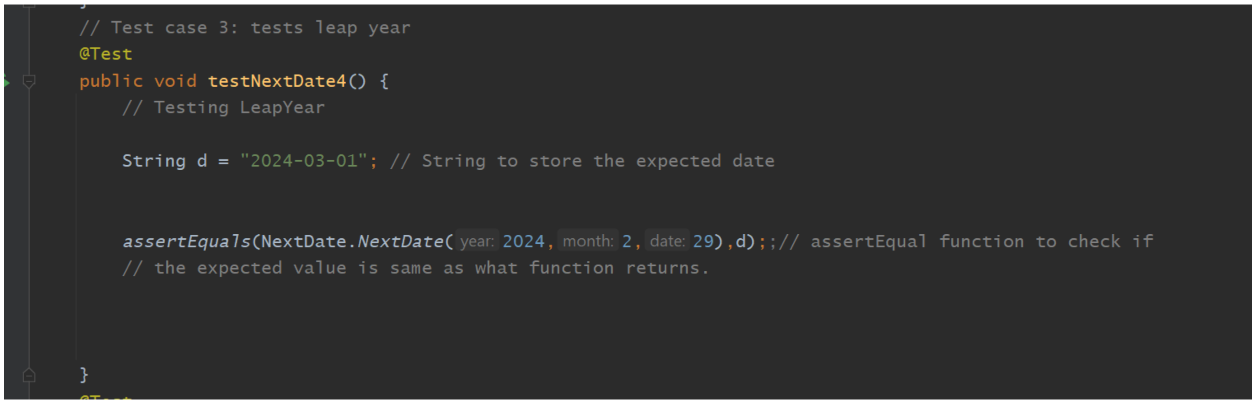
The screenshot above shows the import statements that were used for the JUnit test and test case 1, which tests the invalid month. In this test, we put the year out of the range, so this test case uses the assertEquals statement to check if the function prints out “ïnvalid.” So, when the invalid month is entered into the function, it returns invalid, out of range



```
        assertEquals(NextDate.NextDate( year: 2214,  month: 12,  date: 3),message);
    }
    @Test
    public void testNextDate2() {
  // test same month

        String d = "2019-01-08";//  String to store the expected date
        assertEquals(NextDate.NextDate( year: 2019, month: 1, date: 7), d);// assertEqual function to check if
        // the expected value is same as what function returns.
    }

    // test case 2 : it tests not leap year
    @Test
    public void testNextDate3() {

   // test not Leap Year
        String d = "2018-03-01";//String to store the expected date


        assertEquals(NextDate.NextDate( year: 2018, month: 2, date: 28),d);// assertEqual function to check if
        // the expected value is same as what function returns.


    }
```

The above screenshot shows test cases 2 and 3. Test case 2 (TestNextDate2) tests if the NextDate function returns the correct output. When the date is put, the assertEqualts checks if the function returns the date of the next day that the user put.

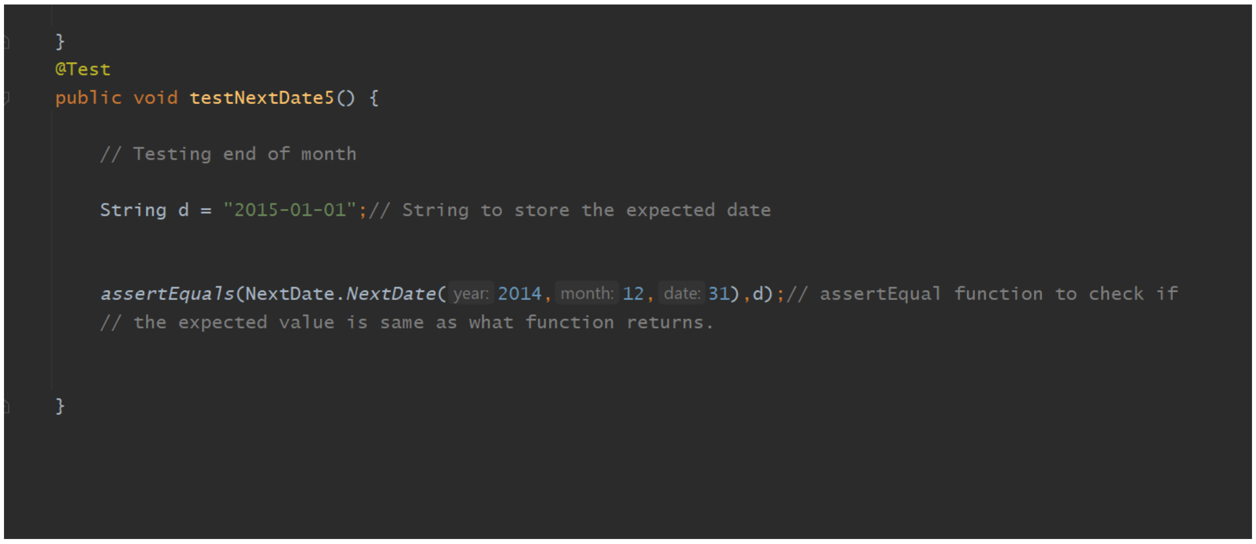
Test case 3(TestNextDate) tests that the function works with the year, which is not a leap year. The NextDate function should return “2018-03-01” because 2018 is not a leap year, and assertEquals check this condition when the user enters 2018-02-28.



```
    // Test case 3: tests leap year
    @Test
    public void testNextDate4() {
        // Testing LeapYear

        String d = "2024-03-01"; // String to store the expected date


        assertEquals(NextDate.NextDate( year: 2024, month: 2, date: 29),d);;// assertEqual function to check if
        // the expected value is same as what function returns.



    }
```

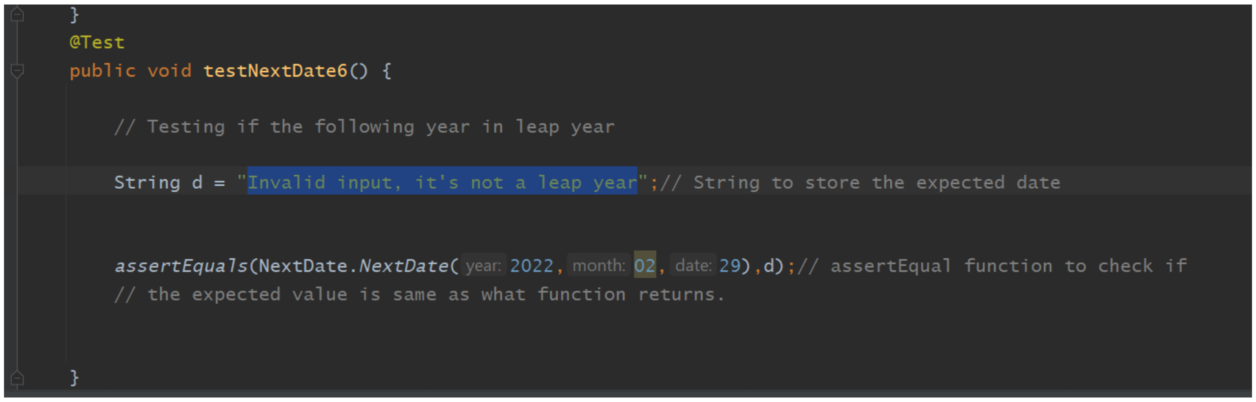
The test case4(testNextDate4) checks if the NextDate function returns the correct date if we enter a date within a leap year. So in this test case, we enter “2024-2-29,” and 2024 is a leap year, and the function should return “2024-03-01”, the next day's date. The assertEquals checks this condition.

Note that the assertEquals statement checks if the expected output that the user entered is the same as the actual value that the function returns when the arguments are entered into the function.



```
}
@Test
public void testNextDate5() {

    // Testing end of month

    String d = "2015-01-01";// String to store the expected date


    assertEquals(NextDate.NextDate( year: 2014, month: 12, date: 31),d);// assertEqual function to check if
    // the expected value is same as what function returns.


}
```

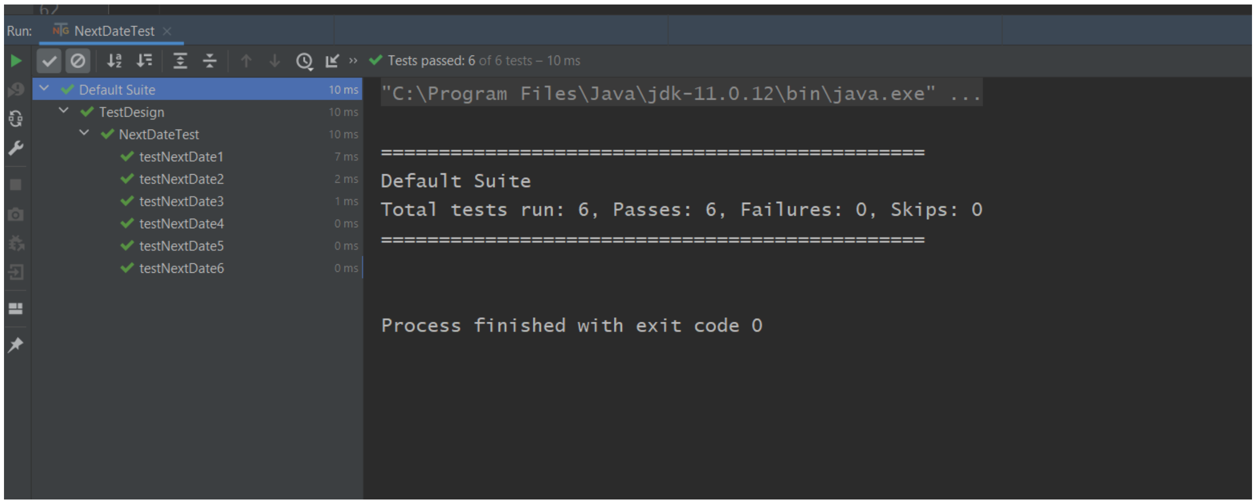
Test case 5 checks whether the function returns the expected output (2015-01-01) for the next year. When the user inputs (2014-12-31), which is the end of the year. So, we check if our code can return next year.



```
}
@Test
public void testNextDate6() {

    // Testing if the following year in leap year

    String d = "Invalid input, it's not a leap year";// String to store the expected date


    assertEquals(NextDate.NextDate( year: 2022, month: 02, date: 29),d);// assertEqual function to check if
    // the expected value is same as what function returns.


}
```

Test case 6, checks of the code can check whether the year entered by the user is a leap year or not. So, in the following test case, we put the date 2022-02-29, and we expect our code to return Ïnvalid input; it's not a leap year” because 2022 is not a leap year.

Test outcomes

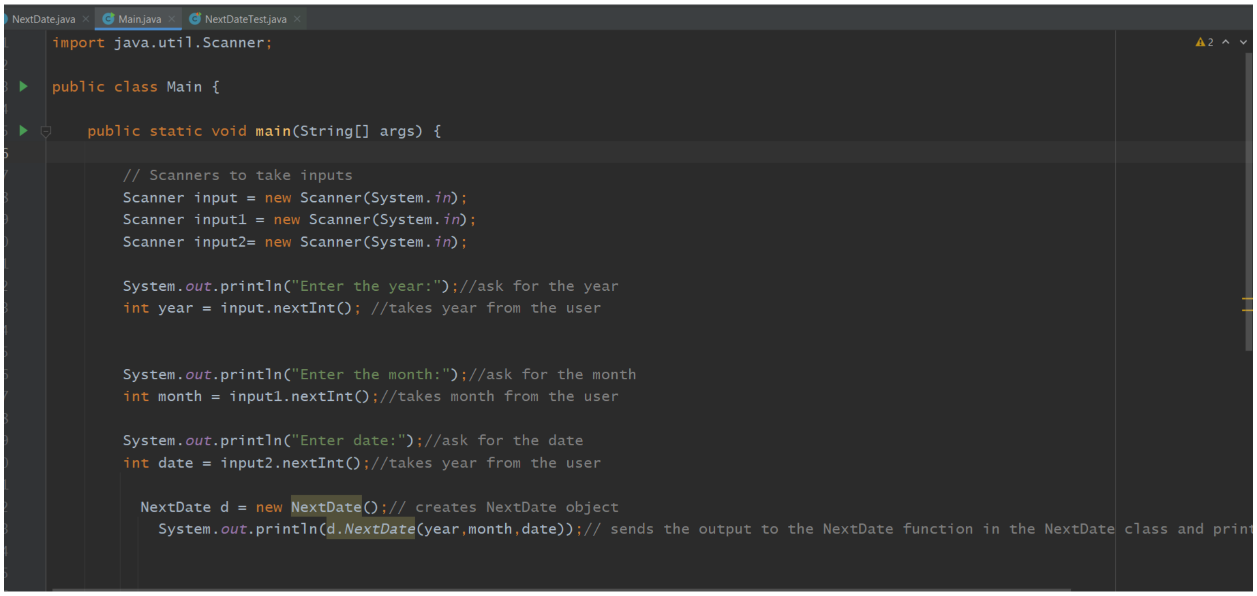


```
"C:\Program Files\Java\jdk-11.0.12\bin\java.exe" ...

===============================================
Default Suite
Total tests run: 6, Passes: 6, Failures: 0, Skips: 0
===============================================


Process finished with exit code 0
```

following screenshot shows that all 6 test cases run successfully.

3. Corresponding code:

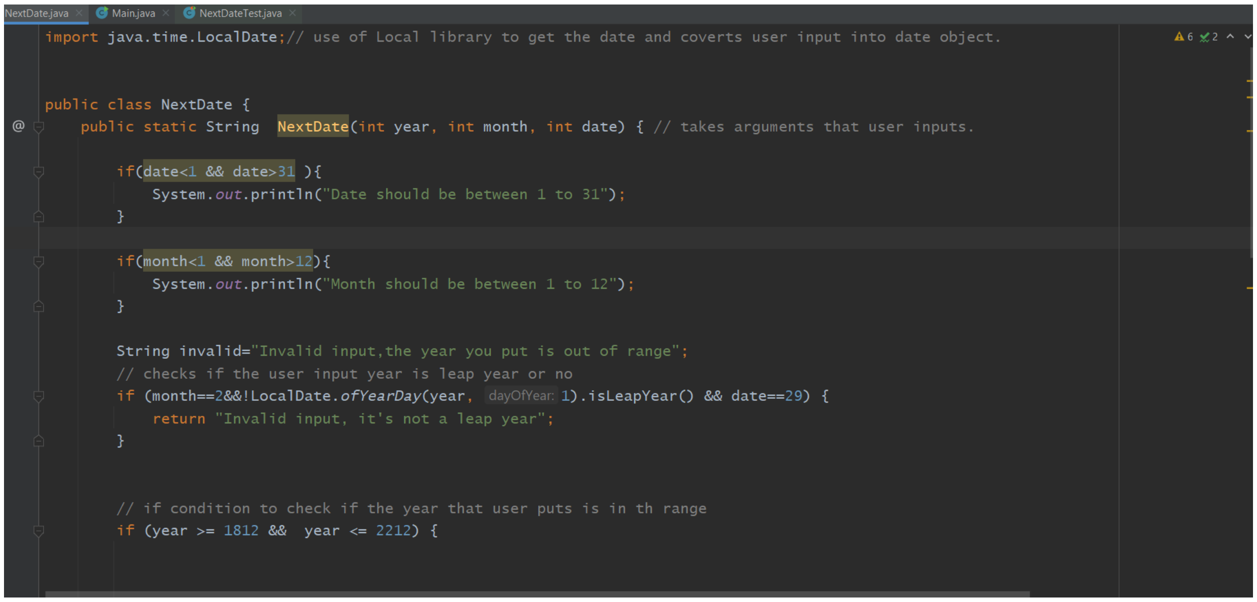
Main.java



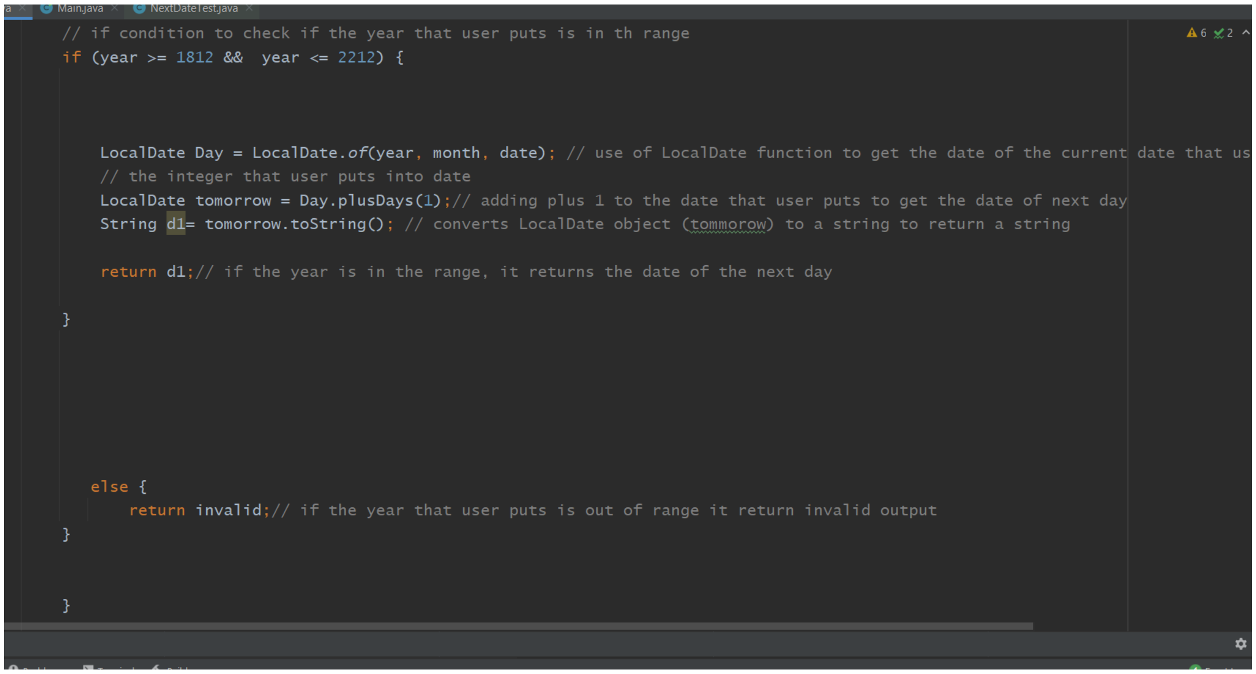
```java
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        // Scanners to take inputs
        Scanner input = new Scanner(System.in);
        Scanner input1 = new Scanner(System.in);
        Scanner input2= new Scanner(System.in);

        System.out.println("Enter the year:");//ask for the year
        int year = input.nextInt(); //takes year from the user


        System.out.println("Enter the month:");//ask for the month
        int month = input1.nextInt();//takes month from the user

        System.out.println("Enter date:");//ask for the date
        int date = input2.nextInt();//takes year from the user

          NextDate d = new NextDate();// creates NextDate object
            System.out.println(d.NextDate(year,month,date));// sends the output to the NextDate function in the NextDate class and print
```

The screenshot above shows the “Main.java” class that asks the user for inputs (year, month, Day) and sends it to the NextDate function to get the date of the next day, and then it prints the answer.

NextDate.java



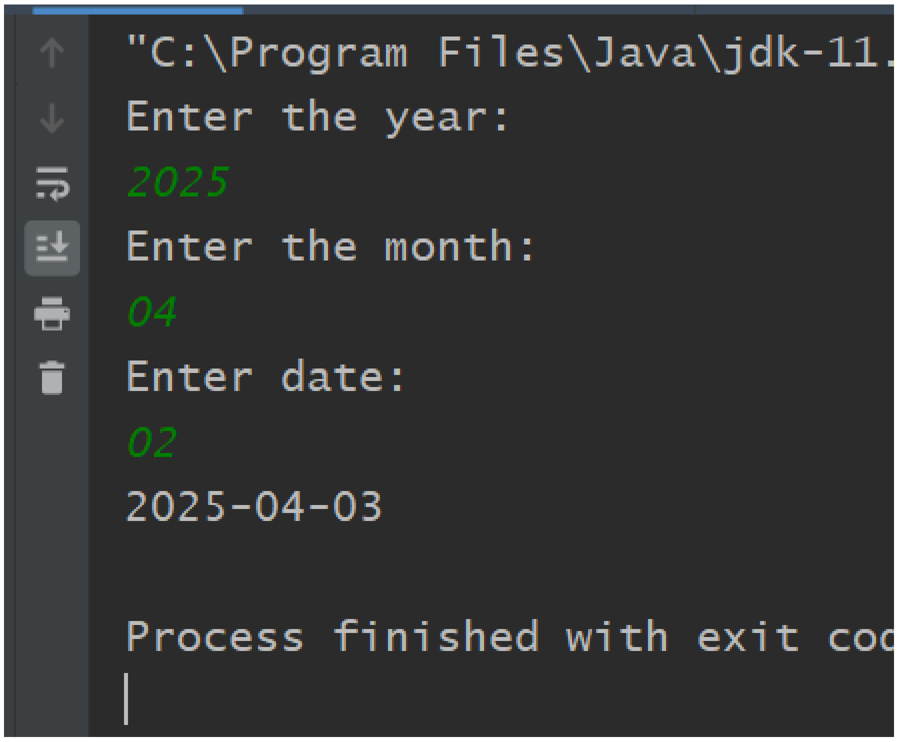
```java
import java.time.LocalDate;// use of Local library to get the date and coverts user input into date object.


public class NextDate {
    public static String  NextDate(int year, int month, int date) { // takes arguments that user inputs.

        if(date<1 && date>31 ){
            System.out.println("Date should be between 1 to 31");
        }

        if(month<1 && month>12){
            System.out.println("Month should be between 1 to 12");
        }

        String invalid="Invalid input,the year you put is out of range";
        // checks if the user input year is leap year or no
        if (month==2&&!LocalDate.ofYearDay(year,  dayOfYear: 1).isLeapYear() && date==29) {
            return "Invalid input, it's not a leap year";
        }


        // if condition to check if the year that user puts is in th range
        if (year >= 1812 &&  year <= 2212) {
```



```java
        // if condition to check if the year that user puts is in th range
        if (year >= 1812 &&  year <= 2212) {



            LocalDate Day = LocalDate.of(year, month, date); // use of LocalDate function to get the date of the current date that us
            // the integer that user puts into date
            LocalDate tomorrow = Day.plusDays(1);// adding plus 1 to the date that user puts to get the date of next day
            String d1= tomorrow.toString(); // converts LocalDate object (tommorow) to a string to return a string

            return d1;// if the year is in the range, it returns the date of the next day

        }







           else {
               return invalid;// if the year that user puts is out of range it return invalid output
        }


        }
```

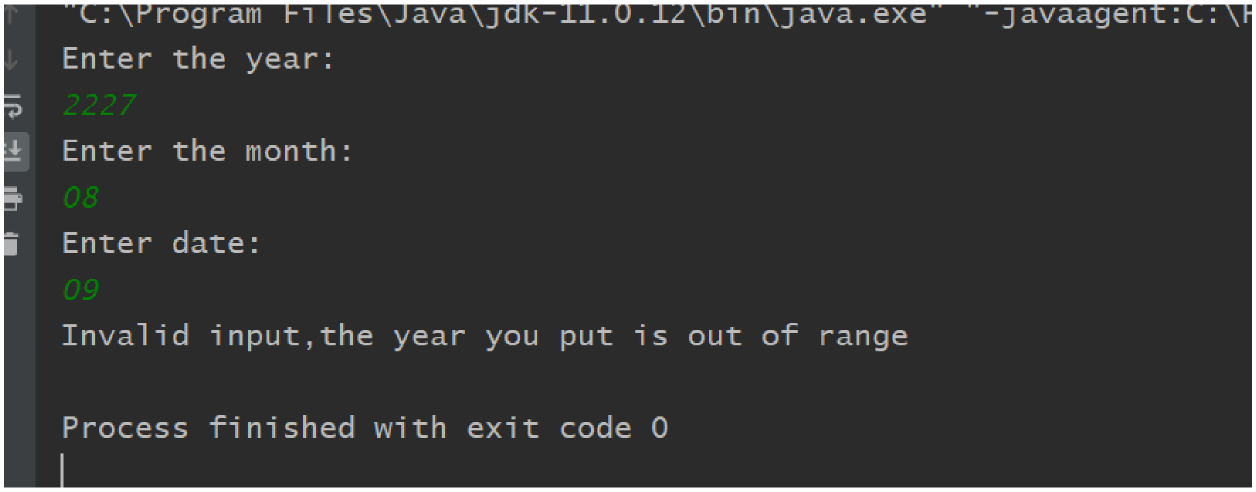
The above Screenshot shows the NextDate class that calculates the next day's date. It uses the LocalDate function to calculate the date of the next day, and the data is stored in the tomorrow variable, which is then converted into a string, and if the year is in the range, it prints the date of the next day. If the year entered by the user is out of range, it prints the “Invalid Input, the year you put is out of the range.”It takes three arguments (year, month, and day).In addition, it also checks if the year entered by the user is a leap year or not and if the values that the user enters (year, month, Day) are in the range.

Code output:

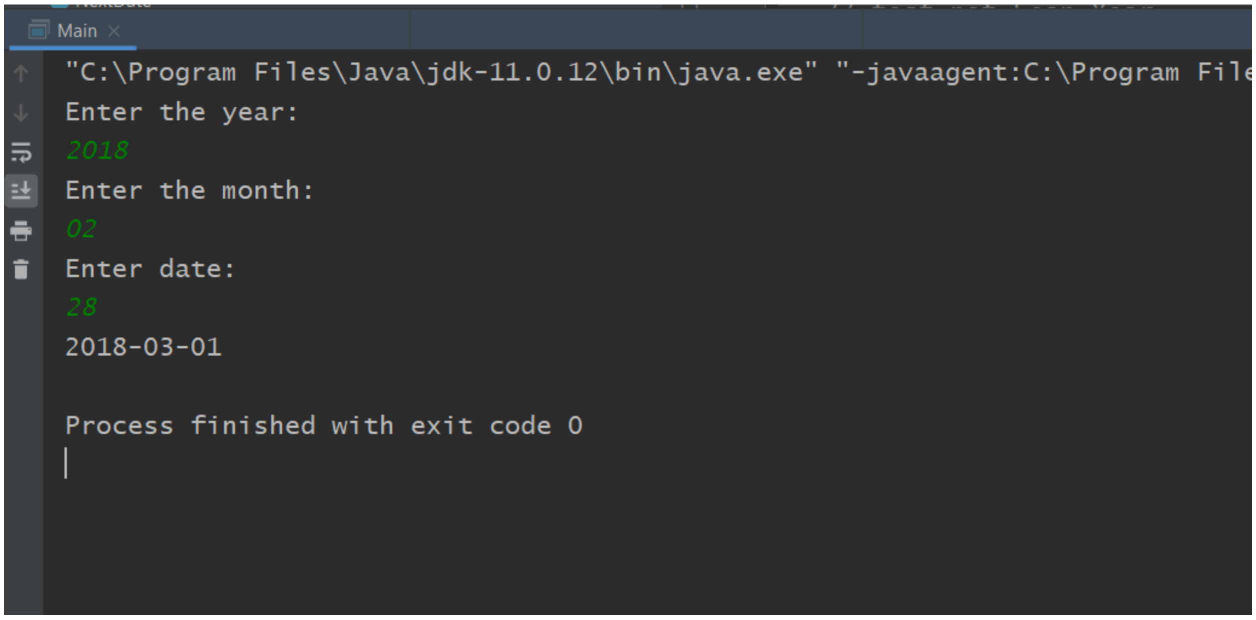


```
"C:\Program Files\Java\jdk-11.
Enter the year:
2025
Enter the month:
04
Enter date:
02
2025-04-03

Process finished with exit coc
```

When the year entered is out of range:



```
"C:\Program Files\Java\jdk-11.0.12\bin\java.exe" "-javaagent:C:\P
Enter the year:
2227
Enter the month:
08
Enter date:
09
Invalid input,the year you put is out of range

Process finished with exit code 0
```



```
"C:\Program Files\Java\jdk-11.0.12\bin\java.exe" "-javaagent:C:\Program File
Enter the year:
2018
Enter the month:
02
Enter date:
28
2018-03-01

Process finished with exit code 0
```