Software Testing: Boundary Value Analysis & Equivalence Class Testing

COURSE : SIT707

NAME: TOMIN JOSE (224353043)

DATE : 13/03/2024

Overview of Software Testing

- Software testing ensures software reliability and correctness.
- Two key testing methods: Boundary Value Analysis (BVA) & Equivalence Class Testing (ECT).
- Helps in identifying test cases efficiently.

Boundary Value Analysis

- BVA focuses on testing at the boundaries of input values.
- Assumption: Errors often occur at boundary values.
- Example: If input range is 1-100, test cases include 1, 2, 99, 100.

Example of Boundary Value Analysis

- Consider a function accepting input between 10-50.
- Test cases: {10, 11, 25, 49, 50}.
- This ensures both lower and upper boundary conditions are tested.

Limitations of Boundary Value Analysis

- Assumes input variables are independent.
- May generate redundant test cases.
- Does not consider logical relationships among inputs.

Equivalence Class Testing (ECT)

- ECT groups inputs into classes that behave similarly.
- Reduces redundant test cases while maintaining effectiveness.
- Example: If input is an age range (0-120), classes may be {0-17}, {18-64}, {65-120}.

Example of Equivalence Class Testing

- Consider a grading system (A, B, C, D, F) for scores 0-100.
- Instead of testing every score, use representative cases: {0-59 → F}, {60-69 → D}, {70-79 → C}, etc.

BVA vs. ECT

- BVA focuses on boundary values, while ECT groups inputs.
- BVA generates more test cases, ECT reduces redundancy.
- Combining both can improve test coverage.

Use Cases of BVA & ECT

- BVA is useful for numeric and range-based validations (e.g., login age restrictions).
- ECT is ideal for categorical inputs (e.g., product categories in an e-commerce site).
- Both are essential in software testing strategies.

JUnit

- JUnit is a popular framework used for unit testing in Java.
- It helps developers test individual pieces of code quickly and easily.
- Makes use of annotations like @Test to define test methods.
- Ensures the code works as expected before deployment.

Test Suites in JUnit

- Combine multiple test classes into one suite.
- Execute all tests in order using the test suite.
- Makes large test projects easy to manage and run.
- Example: @RunWith(Suite.class) and @Suite.SuiteClasses({Class1.class, Class2.class})

Object-Oriented Programming (OOP) in Java

- Java follows OOP principles like C++, but with its own syntax.
- Key OOP pillars:
  - **Encapsulation** (hiding data),
  - **Abstraction** (simplifying complexity),
  - **Inheritance** (using existing code structures),
  - **Polymorphism** (same method, different forms).

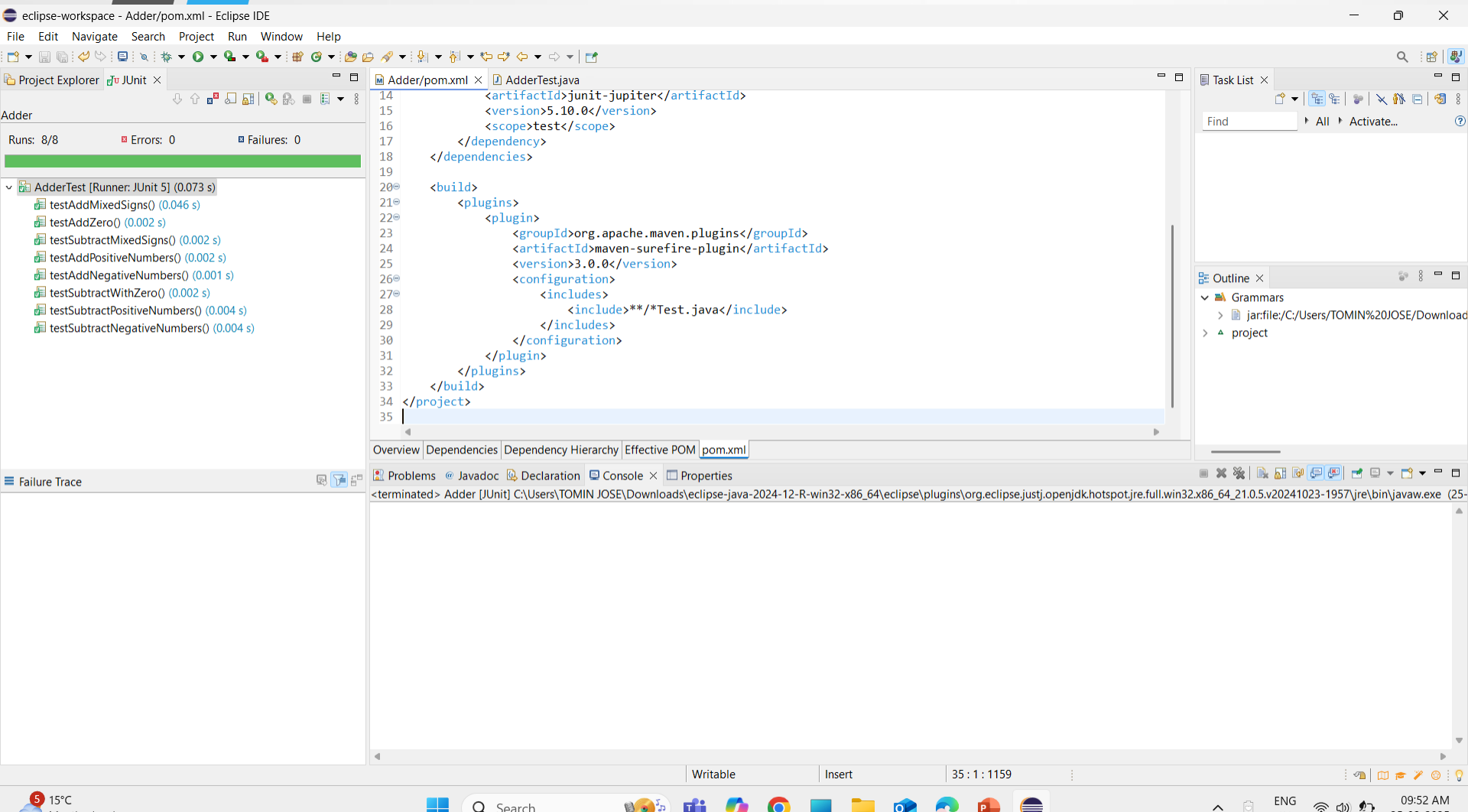
Objects and Classes in Java

- A **class** is a blueprint; an **object** is an instance of that class.
- Common ways to create objects:
  - Using **reference variables** (e.g., FibonacciEngine E1 = new FibonacciEngine();).
  - Through **methods** to modify internal class data.
  - Using **constructors** for initialization at the moment of object creation.
- Store each class in its own file with the same name as the class.

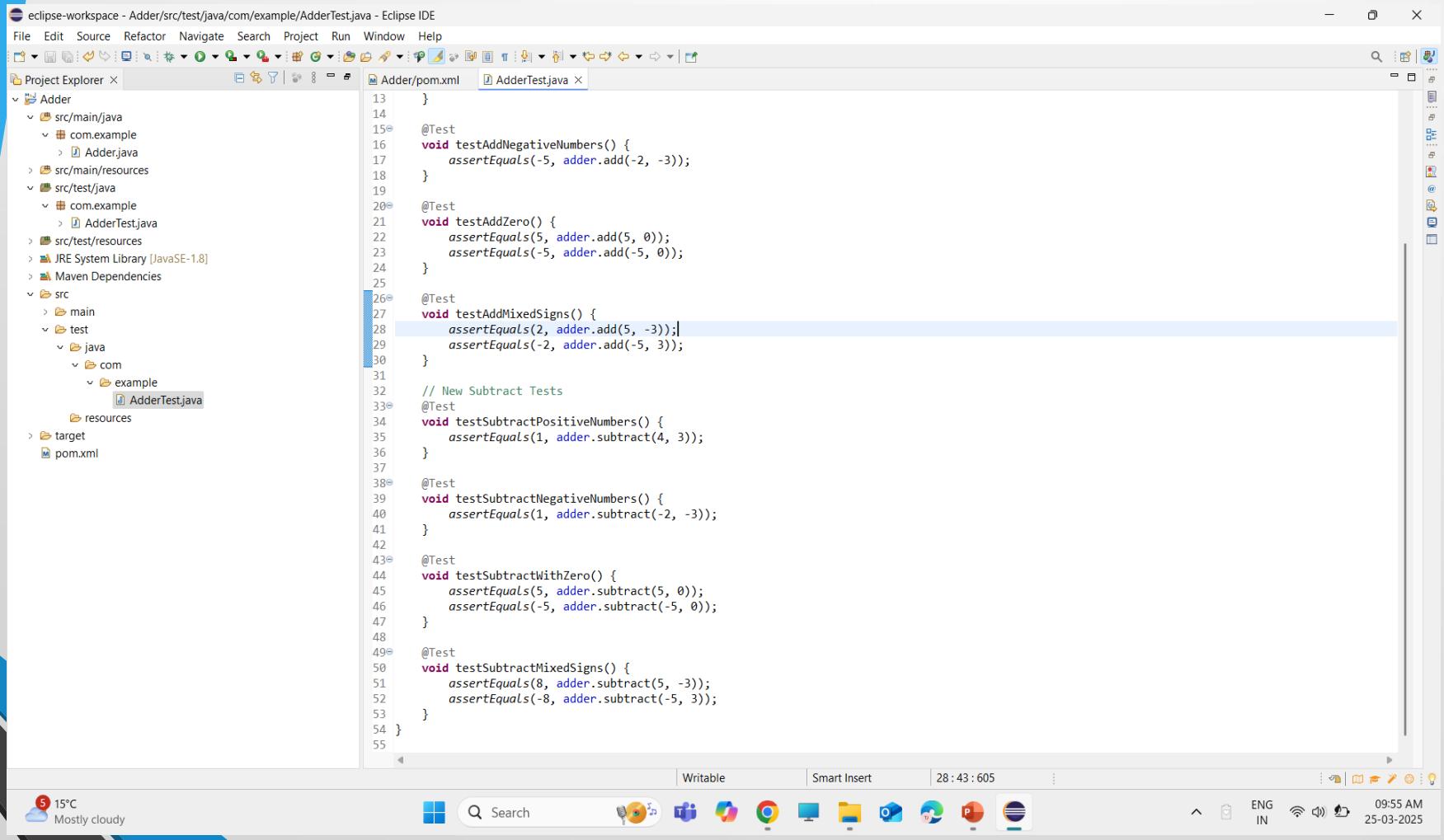
Java Methods

- A **method** is a reusable block of code designed to perform specific tasks.
- Two key types:
  - **Predefined methods** (available from Java libraries, no object creation required).
  - **User-defined methods** (custom methods created by programmers).
- Java supports **method overloading** (multiple methods with the same name but different parameters).

Evidence of activities from the active learning session



Subtractor Test Cases



References

1. JUnit. (2024). *JUnit 5 User Guide*. Available at: https://junit.org/junit5/docs/current/user-guide/ (Accessed: 25 March 2025).
2. Oracle. (2024). *Java Documentation*. Available at: https://docs.oracle.com/javase/ (Accessed: 25 March 2025).
3. Javatpoint. (2024). *Method in Java*. Available at: https://www.javatpoint.com/method-in-java (Accessed: 25 March 2025).

Thank You

Evidence of activities from the active learning session. (Code)
Adder.java:

```
package com.example;


public class Adder {
   public int add(int a, int b) {
      return a + b;
   }


   public int subtract(int a, int b) {
      return a - b;
   }
}
```

Addertest.java:

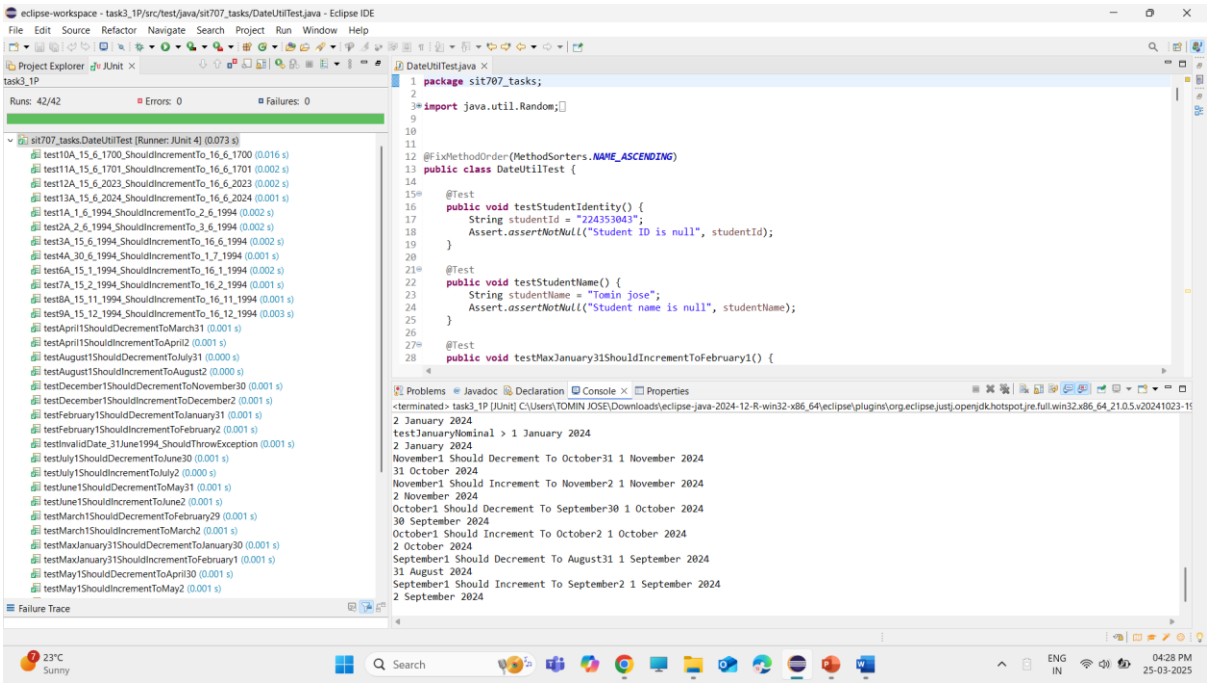
```
package com.example;


import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;


public class AdderTest {


   private Adder adder = new Adder();


   @Test
   void testAddPositiveNumbers() {
      assertEquals(5, adder.add(2, 3));
   }
```

```
@Test
void testAddNegativeNumbers() {
    assertEquals(-5, adder.add(-2, -3));
}

@Test
void testAddZero() {
    assertEquals(5, adder.add(5, 0));
    assertEquals(-5, adder.add(-5, 0));
}

@Test
void testAddMixedSigns() {
    assertEquals(2, adder.add(5, -3));
    assertEquals(-2, adder.add(-5, 3));
}

// New Subtract Tests
@Test
void testSubtractPositiveNumbers() {
    assertEquals(1, adder.subtract(4, 3));
}

@Test
void testSubtractNegativeNumbers() {
    assertEquals(1, adder.subtract(-2, -3));
}
```

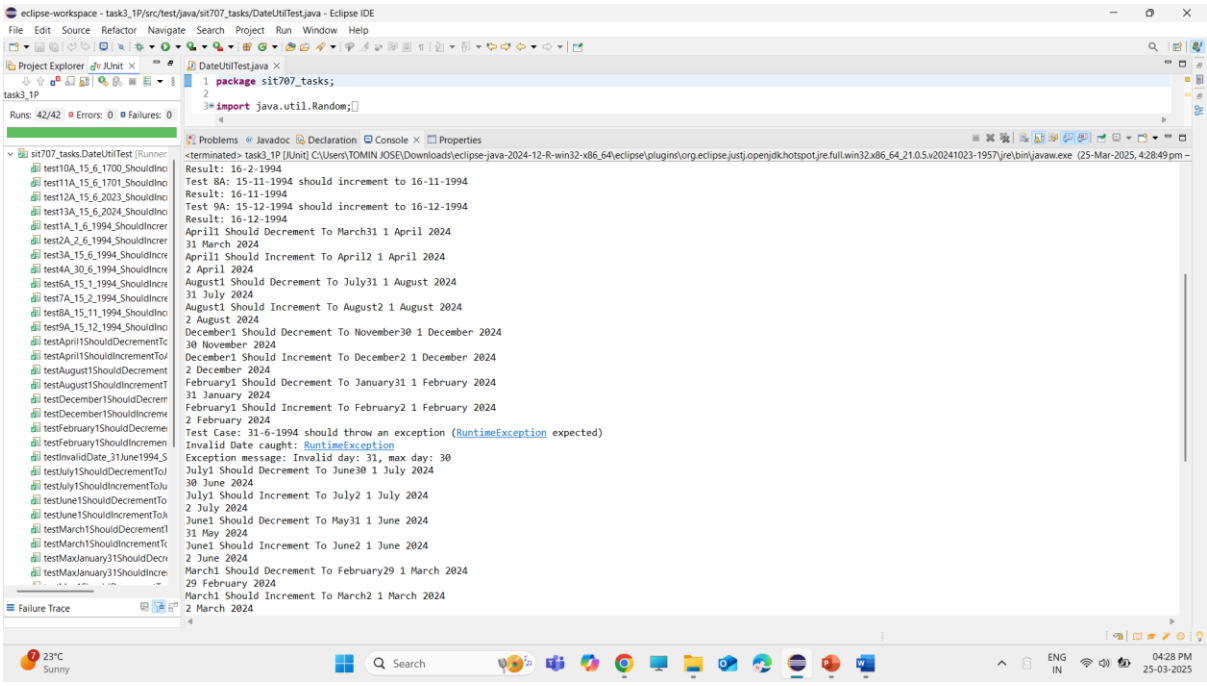
```
    @Test
    void testSubtractWithZero() {
        assertEquals(5, adder.subtract(5, 0));
        assertEquals(-5, adder.subtract(-5, 0));
    }

    @Test
    void testSubtractMixedSigns() {
        assertEquals(8, adder.subtract(5, -3));
        assertEquals(-8, adder.subtract(-5, 3));
    }
}
```

- **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)**

```
package sit707_tasks;

import java.util.Random;

import org.junit.Assert;
import org.junit.Test;
import org.junit.FixMethodOrder;
import org.junit.runners.MethodSorters;



@FixMethodOrder(MethodSorters.NAME_ASCENDING)
public class DateUtilTest {

	@Test
	public void testStudentIdentity() {
		String studentId = "224353043";
		Assert.assertNotNull("Student ID is null", studentId);
	}

	@Test
	public void testStudentName() {
		String studentName = "Tomin jose";
		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() {
            DateUtil date = new DateUtil(1, 1, 2024);
System.out.println("January1 Should Increment To January2 " + date);
date.increment();
```

```
    System.out.println(date);
    Assert.assertEquals(1, date.getMonth());
    Assert.assertEquals(2, date.getDay());

        }

        @Test
        public void testMinJanuary1ShouldDecrementToDecember31() {
                DateUtil date = new DateUtil(1, 1, 2024);
    System.out.println("January1 Should Decrement To December31 " + date);
    date.decrement();
    System.out.println(date);
    Assert.assertEquals(12, date.getMonth());
    Assert.assertEquals(31, date.getDay());
        }

        /*
         * Write tests for rest months of year 2024.
         */

        @Test
        public void testFebruary1ShouldIncrementToFebruary2() {
          DateUtil date = new DateUtil(1, 2, 2024);
          System.out.println("February1 Should Increment To February2 " + date);
          date.increment();
          System.out.println(date);
          Assert.assertEquals(2, date.getMonth());
          Assert.assertEquals(2, date.getDay());
        }

        @Test
        public void testFebruary1ShouldDecrementToJanuary31() {
          DateUtil date = new DateUtil(1, 2, 2024);
          System.out.println("February1 Should Decrement To January31 " +
date);
          date.decrement();
          System.out.println(date);
          Assert.assertEquals(1, date.getMonth());
          Assert.assertEquals(31, date.getDay());
        }
```

```
    @Test
    public void testMarch1ShouldIncrementToMarch2() {
      DateUtil date = new DateUtil(1, 3, 2024);
      System.out.println("March1 Should Increment To March2 " + date);
      date.increment();
      System.out.println(date);
      Assert.assertEquals(3, date.getMonth());
      Assert.assertEquals(2, date.getDay());
    }

    @Test
    public void testMarch1ShouldDecrementToFebruary29() { // 2024 is a leap
year
      DateUtil date = new DateUtil(1, 3, 2024);
      System.out.println("March1 Should Decrement To February29 " + date);
      date.decrement();
      System.out.println(date);
      Assert.assertEquals(2, date.getMonth());
      Assert.assertEquals(29, date.getDay());
    }

    @Test
    public void testApril1ShouldIncrementToApril2() {
      DateUtil date = new DateUtil(1, 4, 2024);
      System.out.println("April1 Should Increment To April2 " + date);
      date.increment();
      System.out.println(date);
      Assert.assertEquals(4, date.getMonth());
      Assert.assertEquals(2, date.getDay());
    }

    @Test
    public void testApril1ShouldDecrementToMarch31() {
      DateUtil date = new DateUtil(1, 4, 2024);
      System.out.println("April1 Should Decrement To March31 " + date);
      date.decrement();
      System.out.println(date);
      Assert.assertEquals(3, date.getMonth());
      Assert.assertEquals(31, date.getDay());
    }
```

```
@Test
public void testMay1ShouldIncrementToMay2() {
  DateUtil date = new DateUtil(1, 5, 2024);
  System.out.println("May1 Should Increment To May2 " + date);
  date.increment();
  System.out.println(date);
  Assert.assertEquals(5, date.getMonth());
  Assert.assertEquals(2, date.getDay());
}

@Test
public void testMay1ShouldDecrementToApril30() {
  DateUtil date = new DateUtil(1, 5, 2024);
  System.out.println("May1 Should Decrement To April30 " + date);
  date.decrement();
  System.out.println(date);
  Assert.assertEquals(4, date.getMonth());
  Assert.assertEquals(30, date.getDay());
}

@Test
public void testJune1ShouldIncrementToJune2() {
  DateUtil date = new DateUtil(1, 6, 2024);
  System.out.println("June1 Should Increment To June2 " + date);
  date.increment();
  System.out.println(date);
  Assert.assertEquals(6, date.getMonth());
  Assert.assertEquals(2, date.getDay());
}

@Test
public void testJune1ShouldDecrementToMay31() {
  DateUtil date = new DateUtil(1, 6, 2024);
  System.out.println("June1 Should Decrement To May31 " + date);
  date.decrement();
  System.out.println(date);
  Assert.assertEquals(5, date.getMonth());
  Assert.assertEquals(31, date.getDay());
}

@Test
```

```
public void testJuly1ShouldIncrementToJuly2() {
  DateUtil date = new DateUtil(1, 7, 2024);
  System.out.println("July1 Should Increment To July2 " + date);
  date.increment();
  System.out.println(date);
  Assert.assertEquals(7, date.getMonth());
  Assert.assertEquals(2, date.getDay());
}

@Test
public void testJuly1ShouldDecrementToJune30() {
  DateUtil date = new DateUtil(1, 7, 2024);
  System.out.println("July1 Should Decrement To June30 " + date);
  date.decrement();
  System.out.println(date);
  Assert.assertEquals(6, date.getMonth());
  Assert.assertEquals(30, date.getDay());
}

@Test
public void testAugust1ShouldIncrementToAugust2() {
  DateUtil date = new DateUtil(1, 8, 2024);
  System.out.println("August1 Should Increment To August2 " + date);
  date.increment();
  System.out.println(date);
  Assert.assertEquals(8, date.getMonth());
  Assert.assertEquals(2, date.getDay());
}

@Test
public void testAugust1ShouldDecrementToJuly31() {
  DateUtil date = new DateUtil(1, 8, 2024);
  System.out.println("August1 Should Decrement To July31 " + date);
  date.decrement();
  System.out.println(date);
  Assert.assertEquals(7, date.getMonth());
  Assert.assertEquals(31, date.getDay());
}

@Test
public void testSeptember1ShouldIncrementToSeptember2() {
```

```
        DateUtil date = new DateUtil(1, 9, 2024);
        System.out.println("September1 Should Increment To September2 " +
date);
        date.increment();
        System.out.println(date);
        Assert.assertEquals(9, date.getMonth());
        Assert.assertEquals(2, date.getDay());
    }

    @Test
    public void testSeptember1ShouldDecrementToAugust31() {
        DateUtil date = new DateUtil(1, 9, 2024);
        System.out.println("September1 Should Decrement To August31 " +
date);
        date.decrement();
        System.out.println(date);
        Assert.assertEquals(8, date.getMonth());
        Assert.assertEquals(31, date.getDay());
    }

    @Test
    public void testOctober1ShouldIncrementToOctober2() {
        DateUtil date = new DateUtil(1, 10, 2024);
        System.out.println("October1 Should Increment To October2 " + date);
        date.increment();
        System.out.println(date);
        Assert.assertEquals(10, date.getMonth());
        Assert.assertEquals(2, date.getDay());
    }

    @Test
    public void testOctober1ShouldDecrementToSeptember30() {
        DateUtil date = new DateUtil(1, 10, 2024);
        System.out.println("October1 Should Decrement To September30 " +
date);
        date.decrement();
        System.out.println(date);
        Assert.assertEquals(9, date.getMonth());
        Assert.assertEquals(30, date.getDay());
    }
```

```
    @Test
    public void testNovember1ShouldIncrementToNovember2() {
      DateUtil date = new DateUtil(1, 11, 2024);
      System.out.println("November1 Should Increment To November2 " +
date);
      date.increment();
      System.out.println(date);
      Assert.assertEquals(11, date.getMonth());
      Assert.assertEquals(2, date.getDay());
    }

    @Test
    public void testNovember1ShouldDecrementToOctober31() {
      DateUtil date = new DateUtil(1, 11, 2024);
      System.out.println("November1 Should Decrement To October31 " +
date);
      date.decrement();
      System.out.println(date);
      Assert.assertEquals(10, date.getMonth());
      Assert.assertEquals(31, date.getDay());
    }

    @Test
    public void testDecember1ShouldIncrementToDecember2() {
      DateUtil date = new DateUtil(1, 12, 2024);
      System.out.println("December1 Should Increment To December2 " +
date);
      date.increment();
      System.out.println(date);
      Assert.assertEquals(12, date.getMonth());
      Assert.assertEquals(2, date.getDay());
    }

    @Test
    public void testDecember1ShouldDecrementToNovember30() {
      DateUtil date = new DateUtil(1, 12, 2024);
      System.out.println("December1 Should Decrement To November30 " +
date);
      date.decrement();
      System.out.println(date);
      Assert.assertEquals(11, date.getMonth());
```

```
        Assert.assertEquals(30, date.getDay());
    }


        @Test
        public void test1A_1_6_1994_ShouldIncrementTo_2_6_1994() {
            DateUtil date = new DateUtil(1, 6, 1994);
            System.out.println("Test 1A: 1-6-1994 should increment to 2-6-1994");
            date.increment();
            System.out.println("Result: " + date.getDay() + "-" + date.getMonth() +
"-" + date.getYear());
            Assert.assertEquals(2, date.getDay());
            Assert.assertEquals(6, date.getMonth());
            Assert.assertEquals(1994, date.getYear());
        }

        @Test
        public void test2A_2_6_1994_ShouldIncrementTo_3_6_1994() {
            DateUtil date = new DateUtil(2, 6, 1994);
            System.out.println("Test 2A: 2-6-1994 should increment to 3-6-1994");
            date.increment();
            System.out.println("Result: " + date.getDay() + "-" + date.getMonth() +
"-" + date.getYear());
            Assert.assertEquals(3, date.getDay());
            Assert.assertEquals(6, date.getMonth());
            Assert.assertEquals(1994, date.getYear());
        }

        @Test
        public void test3A_15_6_1994_ShouldIncrementTo_16_6_1994() {
            DateUtil date = new DateUtil(15, 6, 1994);
            System.out.println("Test 3A: 15-6-1994 should increment to 16-6-
1994");
            date.increment();
            System.out.println("Result: " + date.getDay() + "-" + date.getMonth() +
"-" + date.getYear());
            Assert.assertEquals(16, date.getDay());
            Assert.assertEquals(6, date.getMonth());
            Assert.assertEquals(1994, date.getYear());
        }
```

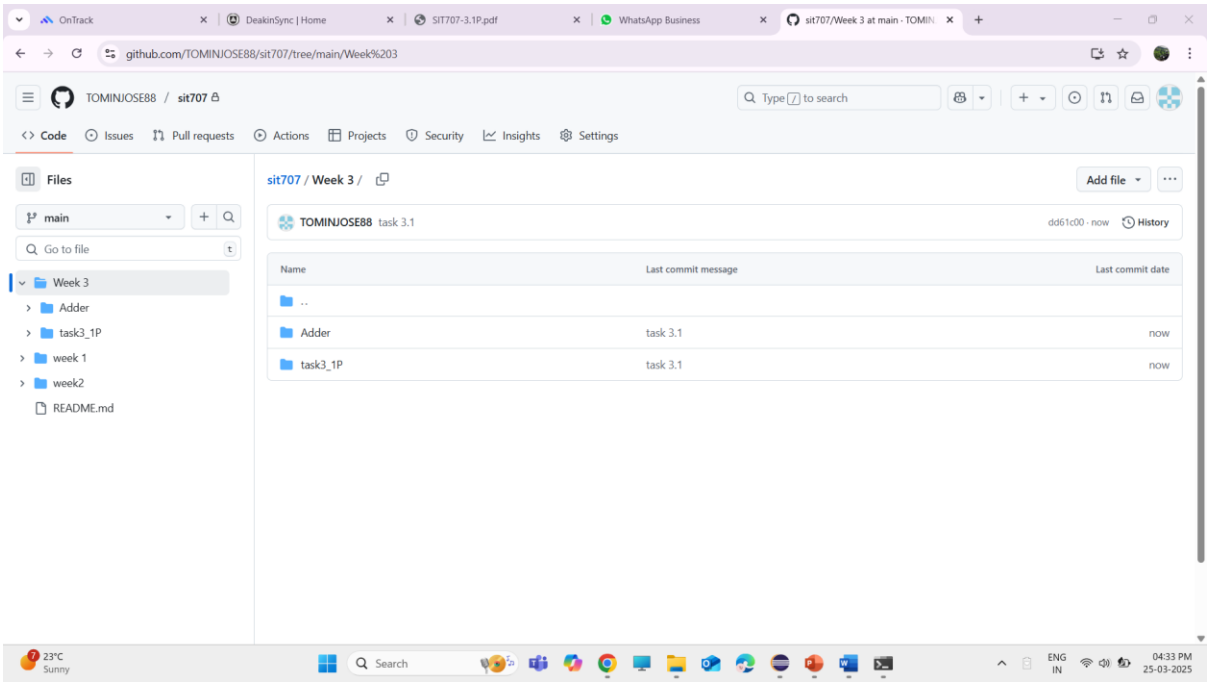
```
    @Test
    public void test4A_30_6_1994_ShouldIncrementTo_1_7_1994() {
      DateUtil date = new DateUtil(30, 6, 1994);
      System.out.println("Test 4A: 30-6-1994 should increment to 1-7-
1994");
      date.increment();
      System.out.println("Result: " + date.getDay() + "-" + date.getMonth() +
"-" + date.getYear());
      Assert.assertEquals(1, date.getDay());
      Assert.assertEquals(7, date.getMonth());
      Assert.assertEquals(1994, date.getYear());
    }

    @Test
    public void testInvalidDate_31June1994_ShouldThrowException() {
      System.out.println("Test Case: 31-6-1994 should throw an exception
(RuntimeException expected)");
      try {
        DateUtil date = new DateUtil(31, 6, 1994);
        // If execution reaches here, no exception was thrown, so the test
should fail
        Assert.fail("Expected RuntimeException for invalid date 31-6-1994");
      } catch (RuntimeException e) {
        System.out.println("Invalid Date caught: " +
e.getClass().getSimpleName());
        System.out.println("Exception message: " + e.getMessage());
        // Optionally verify part of the exception message
        Assert.assertTrue(e.getMessage().contains("Invalid"));
      }
    }


    @Test
    public void test6A_15_1_1994_ShouldIncrementTo_16_1_1994() {
      DateUtil date = new DateUtil(15, 1, 1994);
      System.out.println("Test 6A: 15-1-1994 should increment to 16-1-
1994");
      date.increment();
      System.out.println("Result: " + date.getDay() + "-" + date.getMonth() +
"-" + date.getYear());
      Assert.assertEquals(16, date.getDay());
```

```
        Assert.assertEquals(1, date.getMonth());
        Assert.assertEquals(1994, date.getYear());
    }

    @Test
    public void test7A_15_2_1994_ShouldIncrementTo_16_2_1994() {
        DateUtil date = new DateUtil(15, 2, 1994);
        System.out.println("Test 7A: 15-2-1994 should increment to 16-2-
1994");
        date.increment();
        System.out.println("Result: " + date.getDay() + "-" + date.getMonth() +
"-" + date.getYear());
        Assert.assertEquals(16, date.getDay());
        Assert.assertEquals(2, date.getMonth());
        Assert.assertEquals(1994, date.getYear());
    }

    @Test
    public void test8A_15_11_1994_ShouldIncrementTo_16_11_1994() {
        DateUtil date = new DateUtil(15, 11, 1994);
        System.out.println("Test 8A: 15-11-1994 should increment to 16-11-
1994");
        date.increment();
        System.out.println("Result: " + date.getDay() + "-" + date.getMonth() +
"-" + date.getYear());
        Assert.assertEquals(16, date.getDay());
        Assert.assertEquals(11, date.getMonth());
        Assert.assertEquals(1994, date.getYear());
    }

    @Test
    public void test9A_15_12_1994_ShouldIncrementTo_16_12_1994() {
        DateUtil date = new DateUtil(15, 12, 1994);
        System.out.println("Test 9A: 15-12-1994 should increment to 16-12-
1994");
        date.increment();
        System.out.println("Result: " + date.getDay() + "-" + date.getMonth() +
"-" + date.getYear());
        Assert.assertEquals(16, date.getDay());
        Assert.assertEquals(12, date.getMonth());
        Assert.assertEquals(1994, date.getYear());
```

```
    }

    @Test
    public void test10A_15_6_1700_ShouldIncrementTo_16_6_1700() {
      DateUtil date = new DateUtil(15, 6, 1700);
      System.out.println("Test 10A: 15-6-1700 should increment to 16-6-
1700");
      date.increment();
      System.out.println("Result: " + date.getDay() + "-" + date.getMonth() +
"-" + date.getYear());
      Assert.assertEquals(16, date.getDay());
      Assert.assertEquals(6, date.getMonth());
      Assert.assertEquals(1700, date.getYear());
    }

    @Test
    public void test11A_15_6_1701_ShouldIncrementTo_16_6_1701() {
      DateUtil date = new DateUtil(15, 6, 1701);
      System.out.println("Test 11A: 15-6-1701 should increment to 16-6-
1701");
      date.increment();
      System.out.println("Result: " + date.getDay() + "-" + date.getMonth() +
"-" + date.getYear());
      Assert.assertEquals(16, date.getDay());
      Assert.assertEquals(6, date.getMonth());
      Assert.assertEquals(1701, date.getYear());
    }

    @Test
    public void test12A_15_6_2023_ShouldIncrementTo_16_6_2023() {
      DateUtil date = new DateUtil(15, 6, 2023);
      System.out.println("Test 12A: 15-6-2023 should increment to 16-6-
2023");
      date.increment();
      System.out.println("Result: " + date.getDay() + "-" + date.getMonth() +
"-" + date.getYear());
      Assert.assertEquals(16, date.getDay());
      Assert.assertEquals(6, date.getMonth());
      Assert.assertEquals(2023, date.getYear());
    }
```

```
    @Test
    public void test13A_15_6_2024_ShouldIncrementTo_16_6_2024() {
      DateUtil date = new DateUtil(15, 6, 2024);
      System.out.println("Test 13A: 15-6-2024 should increment to 16-6-
2024");
      date.increment();
      System.out.println("Result: " + date.getDay() + "-" + date.getMonth() +
"-" + date.getYear());
      Assert.assertEquals(16, date.getDay());
      Assert.assertEquals(6, date.getMonth());
      Assert.assertEquals(2024, date.getYear());
    }

}
```

- **A screenshot of your GitHub page where your latest project folder is pushed.**



Orange table:

13 Test Cases to increment the date code:

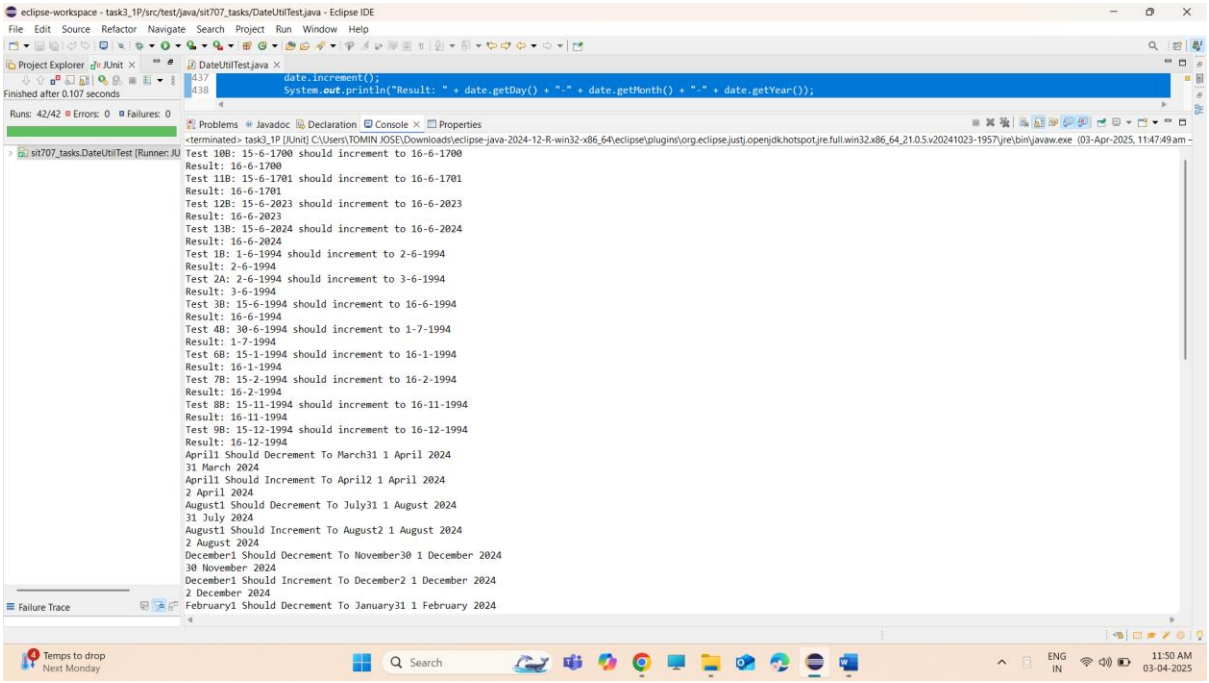
```
@Test
    public void test1B_1_6_1994_ShouldIncrementTo_2_6_1994() {
      DateUtil date = new DateUtil(1, 6, 1994);
      System.out.println("Test 1B: 1-6-1994 should increment to 2-6-1994");
      date.increment();
      System.out.println("Result: " + date.getDay() + "-" + date.getMonth() + "-" + date.getYear());
```

```
    Assert.assertEquals(2, date.getDay());
    Assert.assertEquals(6, date.getMonth());
    Assert.assertEquals(1994, date.getYear());
}

@Test
public void test2B_2_6_1994_ShouldIncrementTo_3_6_1994() {
    DateUtil date = new DateUtil(2, 6, 1994);
    System.out.println("Test 2A: 2-6-1994 should increment to 3-6-1994");
    date.increment();
    System.out.println("Result: " + date.getDay() + "-" + date.getMonth() + "-" + date.getYear());
    Assert.assertEquals(3, date.getDay());
    Assert.assertEquals(6, date.getMonth());
    Assert.assertEquals(1994, date.getYear());
}

@Test
public void test3B_15_6_1994_ShouldIncrementTo_16_6_1994() {
    DateUtil date = new DateUtil(15, 6, 1994);
    System.out.println("Test 3B: 15-6-1994 should increment to 16-6-1994");
    date.increment();
    System.out.println("Result: " + date.getDay() + "-" + date.getMonth() + "-" + date.getYear());
    Assert.assertEquals(16, date.getDay());
    Assert.assertEquals(6, date.getMonth());
    Assert.assertEquals(1994, date.getYear());
}

@Test
public void test4B_30_6_1994_ShouldIncrementTo_1_7_1994() {
    DateUtil date = new DateUtil(30, 6, 1994);
    System.out.println("Test 4B: 30-6-1994 should increment to 1-7-1994");
    date.increment();
    System.out.println("Result: " + date.getDay() + "-" + date.getMonth() + "-" + date.getYear());
    Assert.assertEquals(1, date.getDay());
    Assert.assertEquals(7, date.getMonth());
    Assert.assertEquals(1994, date.getYear());
}

@Test
public void testInvalidDate_31June1994_ShouldThrowException() {
    System.out.println("Test 5B Case: 31-6-1994 should throw an exception (RuntimeException expected)");
    try {
        DateUtil date = new DateUtil(31, 6, 1994);
        // If execution reaches here, no exception was thrown, so the test should fail
        Assert.fail("Expected RuntimeException for invalid date 31-6-1994");
    } catch (RuntimeException e) {
        System.out.println("Invalid Date caught: " + e.getClass().getSimpleName());
        System.out.println("Exception message: " + e.getMessage());
        // Optionally verify part of the exception message
        Assert.assertTrue(e.getMessage().contains("Invalid"));
    }
}


@Test
public void test6B_15_1_1994_ShouldIncrementTo_16_1_1994() {
    DateUtil date = new DateUtil(15, 1, 1994);
    System.out.println("Test 6B: 15-1-1994 should increment to 16-1-1994");
    date.increment();
    System.out.println("Result: " + date.getDay() + "-" + date.getMonth() + "-" + date.getYear());
    Assert.assertEquals(16, date.getDay());
```

```
    Assert.assertEquals(1, date.getMonth());
    Assert.assertEquals(1994, date.getYear());
}

@Test
public void test7B_15_2_1994_ShouldIncrementTo_16_2_1994() {
    DateUtil date = new DateUtil(15, 2, 1994);
    System.out.println("Test 7B: 15-2-1994 should increment to 16-2-1994");
    date.increment();
    System.out.println("Result: " + date.getDay() + "-" + date.getMonth() + "-" + date.getYear());
    Assert.assertEquals(16, date.getDay());
    Assert.assertEquals(2, date.getMonth());
    Assert.assertEquals(1994, date.getYear());
}

@Test
public void test8B_15_11_1994_ShouldIncrementTo_16_11_1994() {
    DateUtil date = new DateUtil(15, 11, 1994);
    System.out.println("Test 8B: 15-11-1994 should increment to 16-11-1994");
    date.increment();
    System.out.println("Result: " + date.getDay() + "-" + date.getMonth() + "-" + date.getYear());
    Assert.assertEquals(16, date.getDay());
    Assert.assertEquals(11, date.getMonth());
    Assert.assertEquals(1994, date.getYear());
}

@Test
public void test9B_15_12_1994_ShouldIncrementTo_16_12_1994() {
    DateUtil date = new DateUtil(15, 12, 1994);
    System.out.println("Test 9B: 15-12-1994 should increment to 16-12-1994");
    date.increment();
    System.out.println("Result: " + date.getDay() + "-" + date.getMonth() + "-" + date.getYear());
    Assert.assertEquals(16, date.getDay());
    Assert.assertEquals(12, date.getMonth());
    Assert.assertEquals(1994, date.getYear());
}

@Test
public void test10B_15_6_1700_ShouldIncrementTo_16_6_1700() {
    DateUtil date = new DateUtil(15, 6, 1700);
    System.out.println("Test 10B: 15-6-1700 should increment to 16-6-1700");
    date.increment();
    System.out.println("Result: " + date.getDay() + "-" + date.getMonth() + "-" + date.getYear());
    Assert.assertEquals(16, date.getDay());
    Assert.assertEquals(6, date.getMonth());
    Assert.assertEquals(1700, date.getYear());
}

@Test
public void test11B_15_6_1701_ShouldIncrementTo_16_6_1701() {
    DateUtil date = new DateUtil(15, 6, 1701);
    System.out.println("Test 11B: 15-6-1701 should increment to 16-6-1701");
    date.increment();
    System.out.println("Result: " + date.getDay() + "-" + date.getMonth() + "-" + date.getYear());
    Assert.assertEquals(16, date.getDay());
    Assert.assertEquals(6, date.getMonth());
    Assert.assertEquals(1701, date.getYear());
}

@Test
public void test12B_15_6_2023_ShouldIncrementTo_16_6_2023() {
```

```
        DateUtil date = new DateUtil(15, 6, 2023);
        System.out.println("Test 12B: 15-6-2023 should increment to 16-6-2023");
        date.increment();
        System.out.println("Result: " + date.getDay() + "-" + date.getMonth() + "-" + date.getYear());
        Assert.assertEquals(16, date.getDay());
        Assert.assertEquals(6, date.getMonth());
        Assert.assertEquals(2023, date.getYear());
    }

    @Test
    public void test13B_15_6_2024_ShouldIncrementTo_16_6_2024() {
        DateUtil date = new DateUtil(15, 6, 2024);
        System.out.println("Test 13B: 15-6-2024 should increment to 16-6-2024");
        date.increment();
        System.out.println("Result: " + date.getDay() + "-" + date.getMonth() + "-" + date.getYear());
        Assert.assertEquals(16, date.getDay());
        Assert.assertEquals(6, date.getMonth());
        Assert.assertEquals(2024, date.getYear());
    }

}
```

Output in the console and junit for orange table:



Output for console is a mixture of results it doesn't have a order

output of the orange table:

est 10B: 15-6-1700 should increment to 16-6-1700
Result: 16-6-1700
Test 11B: 15-6-1701 should increment to 16-6-1701
Result: 16-6-1701
Test 12B: 15-6-2023 should increment to 16-6-2023

Result: 16-6-2023
Test 13B: 15-6-2024 should increment to 16-6-2024
Result: 16-6-2024
Test 1B: 1-6-1994 should increment to 2-6-1994
Result: 2-6-1994
Test 2A: 2-6-1994 should increment to 3-6-1994
Result: 3-6-1994
Test 3B: 15-6-1994 should increment to 16-6-1994
Result: 16-6-1994
Test 4B: 30-6-1994 should increment to 1-7-1994
Result: 1-7-1994
Test 6B: 15-1-1994 should increment to 16-1-1994
Result: 16-1-1994
Test 7B: 15-2-1994 should increment to 16-2-1994
Result: 16-2-1994
Test 8B: 15-11-1994 should increment to 16-11-1994
Result: 16-11-1994
Test 9B: 15-12-1994 should increment to 16-12-1994
Result: 16-12-1994
Test 5B Case: 31-6-1994 should throw an exception (RuntimeException expected)
Invalid Date caught: RuntimeException
Exception message: Invalid day: 31, max day: 30