Bug Report

Class Year

1. Method to be Tested: 1. Class Year "public Year ()"

Description: Creates a new Year, based on the current system date/time.

2. Test Cases and Status

Test Case ID	Input Parameters	Expected Output	Actual Output	Status (Passed/Failed)
TC-01	[ ]	2025	2025	Passed / Failed

3. Additional Notes

- **Note**: Returns the year at the time of execution

- **Steps to Reproduce**: Call new Year() and retrieve the value using getYear()

---

1. Method to be Tested: 2. Class Year "public Year (int year)"

Description: Creates a time period representing a single year.

Parameters: year - the year.

2. Test Cases and Status

Test Case ID	Input Parameters	Expected Output	Actual Output	Status (Passed/Failed)
TC-01	[9999]	9999	9999	Passed / Failed

TC-02	[1900]	1900	1900	Passed / Failed
TC-03	[2024]	2024	2024	Passed / Failed
TC-04	[1899]	Exception	1899	Passed / Failed
TC-05	Null	Exception	Actual: No exception thrown	Passed / Failed

3. Additional Notes

- For testing above and below the bound (e.g 1899 & 10000) it should fail and through an exception and it does not it fails to through exceptions

- Testing a Null value It should through "`IllegalArgumentException`" but it does not so it fails

---

1. Method to be Tested: 3. Class Year "Year(java.util.Date time)"

Description: Creates a new Year, based on a particular instant in time, using the default time zone.

Parameters:

time - the time.

2. Test Cases and Status

Test Case ID	Input Parameters	Expected Output	Actual Output	Status (Passed/Failed)
TC-01	[1/7/2005]	2005	2005	Passed / Failed
TC-02	[1/1/2025 0:0:0]	2025	2025	Passed / Failed
TC-03	[12/1/2024 23:59:59]	2024	2024	Passed / Failed

3. Additional Notes

- Tested Boundary cases beginning of the Year and the end to make sure it works with boundaries.

---

1. Method to be Tested: 4. Class Year "Year(java.util.Date time,java.util.TimeZone zone)"

Description: Constructs a year, based on a particular instant in time and a time zone.

Parameters:

time - the time.

zone - the time zone.

2. Test Cases and Status

Test Case ID	Input Parameters	Expected Output	Actual Output	Status (Passed/Failed)
TC-01	[time=12/31/2021, zone="ASIA/TOKYO"]	2021	2021	Passed / Failed
TC-02	[time=12/31/2021, zone="America/New_York"]	2024	2024	Passed / Failed
TC-03	[time=12/31/2021, zone="ASIA/TOKYO"]	2025	2025	Passed / Failed
TC-04	[null, zone=TimeZone.*getDefault*()]	Exception	Actual: No exception thrown	Passed / Failed

3. Additional Notes

- for setting time = null an exception should be raised else the function test should fail, and it does as there is no exception raised.

-

---

1. Method to be Tested: 5. Class Year "RegularTimePeriod previous()"

Description: Returns the year preceding this one.

Specified by:

previous in class RegularTimePeriod

Returns:

The year preceding this one (or null if the current year is 1900).

2. Test Cases and Status

Test Case ID	Input Parameters	Expected Output	Actual Output	Status (Passed/Failed)
TC-01	[2024]	2023	2023	Passed / Failed
TC-02	[1899]	Exception	Actual: No exception thrown	Passed / Failed
TC-03	[-2000]	Exception	Actual: No exception thrown	Passed / Failed
TC-04	[1900]	Null	Actual: No exception thrown	Passed / Failed

3. Additional Notes

- [1899] the function returns 1898 and it should not accept the input as it is less than the min an Exception should be raised

- [-2000] the function returns -1999 and logically it should be -2001 but anyway it should not accept the input as it is less than the min Exception should be raised

- [1900] the function should return null and it doesn't.

---

1. Method to be Tested: 6. Class Year "RegularTimePeriod next()"

Description: public RegularTimePeriod **next**()

Returns the year following this one.

Specified by:

next in class RegularTimePeriod

Returns:

The year following this one (or null if the current year is 9999).

2. Test Cases and Status

Test Case ID	Input Parameters	Expected Output	Actual Output	Status (Passed/Failed)
TC-01	[2025]	2026	2026	Passed / Failed
TC-02	[9999]	10000	10000	Passed / Failed
TC-03	[1899]	Exception	Actual: No exception thrown	Passed / Failed
TC-04	[-2025]	Exception	Actual: No exception thrown	Passed / Failed

3. Additional Notes

-

---

1. Method to be Tested: 7. Class Year "long getSerialIndex()"

Description: Returns a serial index number for the year.

The implementation simply returns the year number (e.g. 2002).

Specified by:

getSerialIndex in class RegularTimePeriod

Returns:

The serial index number.

2. Test Cases and Status

Test Case ID	Input Parameters	Expected Output	Actual Output	Status (Passed/Failed)
TC-01	[2022]	2022L	2022L	Passed / Failed
TC-02	[1899]	Exception	Actual: No exception thrown	Passed / Failed
TC-03	[9999]	9999L	9999L	Passed / Failed
TC-04	[-2000]	Exception	Actual: No exception thrown	Passed / Failed
TC-05	[1900]	1900L	1900L	Passed / Failed
TC-06	[9998]	9998L	9998L	Passed / Failed

3. Additional Notes

- [1899] should through exception as it is an edge case below bounds.

- [-2000] should through exception as it is negative value and below bounds.

---

1. Method to be Tested: 8. Class Year "long getFirstMillisecond(java.util.Calendar calendar)"

Description: Returns the first millisecond of the year, evaluated using the supplied calendar (which determines the time zone).

Specified by:

getFirstMillisecond in class RegularTimePeriod

Parameters:

calendar - the calendar.

Returns:

The first millisecond of the year.

2. Test Cases and Status

Test Case ID	Input Parameters	Expected Output	Actual Output	Status (Passed/Failed)
TC-01	[1/1/2021]	Jan 1, 2021 00:00:00.000	Returned value from method	Passed / Failed
TC-02	[1/1/1900]	Jan 1, 1900 00:00:00.000	Returned value from method	Passed / Failed
TC-03	[1/1/9999]	Jan 1, 9999 00:00:00.000	Returned value from method	Passed / Failed
TC-04	[1/1/-2025]	Exception	Actual: No exception thrown	Passed / Failed
TC-05	[1/1/10000]	Exception	Actual: No exception thrown	Passed / Failed
TC-06	[1/1/2021] in UTC			Passed / Failed

3. Additional Notes

- exceptions are not raised

---

1. Method to be Tested: **9. Class Year "long getLastMillisecond(java.util.Calendar calendar)"**

Description: [Brief description of the method's purpose and functionality.]

2. Test Cases and Status

Test Case ID	Input Parameters	Expected Output	Actual Output	Status (Passed/Failed)
TC-01	[2021]	Dec 31, 2021 23:59:59.999	Value returned from method	Passed / Failed
TC-02	[1900]	Dec 31, 1900 23:59:59.999	Value returned from method	Passed / Failed
TC-03	[9999]	Dec 31, 9999 23:59:59.999	Value returned from method	Passed / Failed
TC-04	[-2025]	Expected: Exception thrown	If method returns a value: **BUG**	Passed / Failed
TC-05	[10000]	Expected: Exception thrown	If method returns a value: **BUG**	Passed / Failed
TC-06	[2022]	Dec 31, 2022 23:59:59.999 UTC	Value returned from method	Passed / Failed

3. Additional Notes

- Exception should be raised on edge cases more that 9999 and less than 1900

---

1. Method to be Tested: **10. Class Year "Boolean equals(java.lang.Object object)"**

Description: Tests the equality of this Year object to an arbitrary object. Returns true if the target is a Year instance representing the same year as this object. In all other cases, returns false.

Overrides:

equals in class java.lang.Object

Parameters:

object - the object.

Returns:

true if the year of this and the object are the same.

2. Test Cases and Status

Test Case ID	Input Parameters	Expected Output	Actual Output	Status (Passed/Failed)
TC-01	[2024, 2024]	Expected: True	True	Passed / Failed
TC-02	[1899, 1899]	Expected: Exception thrown	True	Passed / Failed
TC-03	[10000, 10000]	Expected: Exception thrown	True	Passed / Failed
TC-04	[2023, 2023]	Expected: True	True	Passed / Failed
TC-07	[2024, 2024]	Expected: True	True	Passed / Failed

3. Additional Notes

- no exceptions raised

1. Method to be Tested: 11. Class Year “int hashCode()”

Description: Returns a hash code for this object instance. The approach described by Joshua Bloch in "Effective Java" has been used here:

http://developer.java.sun.com/developer/Books/effectivejava /Chapter3.pdf

Overrides:

hashCode in class java.lang.Object

Returns:

A hash code.

2. Test Cases and Status

Test Case ID	Input Parameters	Expected Output	Actual Output	Status (Passed/Failed)
TC-01	[2024]	2551	2653	Passed / Failed

3. Additional Notes

- Failed

---

1. Method to be Tested: 12. Class Year “int compareTo(java.lang.Object o1)”

Description: Returns an integer indicating the order of this Year object relative to the specified object: negative == before, zero == same, positive == after.

Specified by:

compareTo in interface java.lang.Comparable

Parameters:

o1 - the object to compare.

Returns:

negative == before, zero == same, positive == after.

2. Test Cases and Status

Test Case ID	Input Parameters	Expected Output	Actual Output	Status (Passed/Failed)
TC-01	[2024, 2024]	0	0	Passed / Failed
TC-02	[2023 , 2024]	>0	>0	Passed / Failed
TC-03	[2025 , 2024]	<0	<0	Passed / Failed
TC-04	[2024, ”2025”]	Exception	_	Passed / Failed
TC-05	[1900, 2025]	<0	<0	Passed / Failed
TC-06	[9999, 2025]	>0	>0	Passed / Failed
TC-07	[2024, null]	Exception	_	Passed / Failed

3. Additional Notes

- the null exception is not handled and no year type validation or exception raised

---

1. Method to be Tested: 13. Class Year “java.lang.String toString()”

Description: Returns a string representing the year..

Overrides:

toString in class RegularTimePeriod

Returns:

A string representing the year.

2. Test Cases and Status

Test Case ID	Input Parameters	Expected Output	Actual Output	Status (Passed/Failed)
TC-01	[2024]	“2024”	“2024”	Passed / Failed
TC-02	[-1000]	“-1000”	“-1000”	Passed / Failed
TC-03	[1900]	“1900”	“1900”	Passed / Failed
TC-04	[9999]	“9999”	“9999”	Passed / Failed
TC-05	[0]	“0”	“0”	Passed / Failed
TC-06	[5]	“5”	“5”	Passed / Failed

3. Additional Notes

- The method should reliably convert both positive and negative year values to their corresponding string representations.

- Edge cases such as year 0, very small numbers (e.g., 5), and very large numbers (e.g., 9999) are included in the test cases.

- No exception handling is expected as toString() should always return a valid string representation of the year.

---

1. Method to be Tested: 14. Class Year “parseYear(String s)”

Description: Parses the string argument as a year.

The string format is YYYY.

Parameters:

s - a string representing the year.

Returns:

null if the string is not parseable, the year otherwise.

2. Test Cases and Status

Test Case ID	Input Parameters	Expected Output	Actual Output	Status (Passed/Failed)
TC-01	[“2024”]	2024	2024	Passed / Failed
TC-02	[“202a”]	Null	_	Passed / Failed
TC-03	[“ 2022“]	2022	2022	Passed / Failed

3. Additional Notes

- the function is provided in the documentation to return Null incase of invalid input but it does not as it failed the “invalid input returns null” check.

Class DiscountCalculatorTest

1. Method to be Tested: 1. Class Year “isTheSpecialWeek”

Description: Checks if the current week is the special promotional week (week 26).

2. Test Cases and Status

Test Case ID	Input Parameters	Expected Output	Actual Output	Status (Passed/Failed)
TC-01	[June 10, 2025]	False	Value returned from method ”False”	Passed / Failed
TC-02	[June 23, 2025]	True	Value returned from method ”True”	Passed / Failed

3. Additional Notes

- June 10, 2025 value is expected to return false as it is not week 26

- June 23, 2025 is week 26 so it returns true as expected

---

1. Method to be Tested: 2. Class Year “getDiscountPercentage”

Description: Calculates a discount based on whether the week number is even or odd.
If even: returns 7% discount.
If odd: returns 5% discount.

2. Test Cases and Status

Test Case ID	Input Parameters	Expected Output	Actual Output	Status (Passed/Failed)
TC-01	[Week(15, 2023)]	5	5	Passed / Failed

TC-02	[Week(10, 2023)]	7	7	Passed / Failed
TC-03	[Week(1, 2023)]	5	5	Passed / Failed
TC-04	[Week(52, 2023)]	7	7	Passed / Failed
TC-05	[Week(53, 2020)]	5	5	Passed / Failed
TC-06	[Week(-50, 2020)]	Expected: Exception thrown	Illegal Argument Exception	Passed / Failed
TC-07	[(Week) null]	Expected: Exception thrown	No exception	Passed / Failed

3. Additional Notes

- tested normal cases like week 15 and week 10
- test boundary cases like week 1 at the beginning of the year, and week 52, 53 some years contain 53 weeks and all passed

- negative week raises an exception was raised successfully!

```
IllegalArgumentException.class
```

- BUGS: no null week object handling!

Class DiscountManagerTest

1. Method to be Tested: 1. Class Year "DiscountManager"

Description: Constructor for the DiscountManager class that initializes the isDiscountsSeason flag and assigns a discount calculator strategy object.

2. Test Cases and Status

Test Case ID	Input Parameters	Expected Output	Actual Output	Status (Passed/Failed)
TC-01	isDiscountsSeason = true, mockCalculator	An instance of DiscountManager with fields correctly initialized	An instance of DiscountManager with fields correctly initialized	Passed / Failed

3. Additional Notes

- This test uses JMock to create a mock object of the IDiscountCalculator interface.

---

1. Method to be Tested: 2. Class Year "calculatePriceAfterDiscount"

Description: [Brief description of the method's purpose and functionality.]

2. Test Cases and Status

Test Case ID	Input Parameters	Expected Output	Actual Output	Status (Passed/Failed)
TC-01	[False, mockedDependency]	100.0	100.0	Passed / Failed

TC-02	[True, mocked Dependency]	80.0	80.0	Passed / Failed
TC-03	[100.0]	95.0	500.0	Passed / Failed
TC-04	[100.0]	93.0	700.0	Passed / Failed

3. Additional Notes

- TC-01: not discount season input 100 expected output 100
created a mocking object to make sure dependency are not called like getDiscountPercentage, and isTheSpecialWeek returns False

- TC-02: discount season = true, and special week discount should be 20%
created a mocking object to make sure dependency funtions will return expected values

- TC-03: discount season + even week , and made sure dependency returns the expected outputs by using mocking object.

- TC-04: discount season + odd week , and made sure dependency returns the expected outputs by using mocking object.