Method to be Tested: Constructor (Date)

**Description**: Constructor with Date param

**Test Case 1:**

* Input:
* Expected Output: todays year (2024)
* Actual Output: 2024
* Status: Passed

Method to be Tested: Constructor (Int, Int)

**Description**: Constructor with 2 integer parameters a quarter index and year

**Test Case 1:**

* Input: quarter index and year
* Expected Output: quarter index (3)
* Actual Output: 3
* Status: Passed

**Test Case 2:**

* Input: quarter index and year
* Expected Output: year (2022)
* Actual Output: 2022
* Status: Passed

**Test Case 3:**

* Input: quarter index 1 – 4, year 10,000
* Expected Output: Exception
* Actual Output: Exception
* Status: Passed

**Test Case 4:**

* Input: quarter index 1 – 4 and year < 1900
* Expected Output: Exception
* Actual Output: Exception
* Status: Passed

**Test Case 5:**

* Input: -1 and year (1900 – 9999)
* Expected Output: Exception
* Actual Output: -1
* Status: Failed

Notes: It failed because it allowed out of range numbers for quarter to be inputted

**Test Case 6:**

* Input: 5 and year (1900 – 9999)
* Expected Output: Exception
* Actual Output: 5
* Status: Failed

Notes: It failed because it allowed out of range numbers for quarter to be inputted

Method to be Tested: Constructor (Int, Year)

**Description**: Constructor with int param & Year param

**Test Case 1:**

* Input: int quarter index 1-4 & Year year 1900 - 9999
* Expected Output: quarter index (3)
* Actual Output: 3
* Status: Passed

**Test Case 2:**

* Input: int quarter index & Year year
* Expected Output: year (2024)
* Actual Output: 2024
* Status: Passed

**Test Case 3:**

* Input: -1 and year (1900 – 9999)
* Expected Output: Exception
* Actual Output: -1
* Status: Failed

Notes: It failed because it allowed out of range numbers for quarter to be inputted

**Test Case 4:**

* Input: quarter 5 and year (1900 – 9999)
* Expected Output: Exception
* Actual Output: 5
* Status: Failed

Notes: It failed because it allowed out of range numbers for quarter to be inputted

**Test Case 5:**

* Input: quarter index 1 – 4 and year > 9999
* Expected Output: Exception
* Actual Output: Exception
* Status: Passed

**Test Case 6:**

* Input: quarter index 1 - 4 and year < 1900
* Expected Output: Exception
* Actual Output: Exception
* Status: Passed

Method to be Tested: Get First Millisecond

**Description**: gets the first millisecond of a quarter

**Test Case 1:**

* Input: first millisecond from the calendar class
* Expected Output: first Millisecond
* Actual Output: first Millisecond
* Status: Passed

Method to be Tested: Get First Millisecond (Calendar)

**Description**: gets the first millisecond of a quarter but takes a parameter

**Test Case 1:**

* Input: first millisecond from the calendar class
* Expected Output: first Millisecond
* Actual Output: first Millisecond
* Status: Passed

Method to be Tested: Get Last Millisecond

**Description**: gets the last millisecond of a quarter

**Test Case 1:**

* Input: last millisecond from the calendar class
* Expected Output: last Millisecond
* Actual Output: last Millisecond
* Status: Passed

Method to be Tested: Get Last Millisecond (Calendar)

**Description**: gets the last millisecond of a quarter

**Test Case 1:**

* Input: last millisecond from the calendar class
* Expected Output: last Millisecond
* Actual Output: last Millisecond
* Status: Passed

Method to be Tested: Get Quarter

**Description**: returns an integer value representing the quarter of a year (1,2,3,4)

**Test Case 1:**

* Input: Date
* Expected Output: quarter (1,2,3,4)
* Actual Output: quarter (1,2,3,4)
* Status: Passed

Method to be Tested: Get Serial index

**Description**: returns a serial index number for the quarter

**Test Case 1:**

* Input: Date
* Expected Output: ~
* Actual Output: 8098
* Status: failed

Notes: doesn’t specify what the output should be and that outputted number clarifies nothing.

Method to be Tested: Get Year

**Description**: returns an object of type year

**Test Case 1:**

* Input: Date
* Expected Output: Year year
* Actual Output: Year year
* Status: Passed

Method to be Tested: Next

**Description**: gets the next quarter

**Test Case 1:**

* Input: ~
* Expected Output: this quarter + 1
* Actual Output: this quarter + 1
* Status: Passed

**Test Case 2:**

**Description**: Tested a corner case, the next of Q4

* Input: ~
* Expected Output: this quarter + 1
* Actual Output: this quarter + 1
* Status: Passed

Method to be Tested: Previous

**Description**: gets the previous quarter

**Test Case 1:**

* Input: ~
* Expected Output: this quarter - 1
* Actual Output: this quarter - 1
* Status: Passed

**Test Case 2:**

**Description**: Tested a corner case, the previous of Q1.

* Input: ~
* Expected Output: this quarter - 1
* Actual Output: this quarter - 1
* Status: Passed

Method to be Tested: Hash Code

**Description**: Returns a hash code for this object instance.

**Test Case 1:**

* Input: ~
* Expected Output: a string, not null
* Actual Output: not null
* Status: Passed

**Test Case 2:**

**Description**: Tested a corner case, compared 2 hash codes to make sure they’re different.

* Input: ~
* Expected Output: not equal
* Actual Output: not equal
* Status: Passed