GENERAL INFORMATION			
Test Function Name :	testQuarterDefaultCtor	Test Case Number:	#1
Test Case Description:	This test case will verify the functionality of the Quarter class's default constructor by checking if the year and quarter values are set correctly.		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	No input is required to execute this test case.		
Procedural Steps:	1. Call the arrange() method to set up the test environment. 2. Call the default constructor of the Quarter class. 3. Retrieve the year and quarter values using the getYear() and getQuarter() methods of the Quarter class.		
Expected Results of Case:	The year value should be set to 2023 and the quarter value should be set to 2.		
ACTUAL RESULTS			
Output Specifications and comments :	● The test passed successfully. ● The year value was set to 2023 as expected. ● The quarter value was set to 2 as expected.		

GENERAL INFORMATION			
Test Function Name :	testQuarterDateCtor	Test Case Number:	#2
Test Case Description:	This test case will verify the functionality of the Quarter class's constructor with a Date parameter by checking if the year and quarter values are set correctly.		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	A Date object representing April 15, 2023.		
Procedural Steps:	1. Create a Date object representing April 15, 2023. 2. Call the arrange() method, passing the Date object as a parameter to set up the test environment. 3. Retrieve the year and quarter values using the getYear() and getQuarter() methods of the Quarter class.		
Expected Results of Case:	The year value should be set to 2023 and the quarter value should be set to 2.		
ACTUAL RESULTS			
Output Specifications and comments :	● The test passed successfully. ● The year value was set to 2023 as expected. ● The quarter value was set to 2 as expected.		

GENERAL INFORMATION			
Test Function Name :	testQuarterDateCtorBelow1900	Test Case Number:	#3
Test Case Description:	This test case will verify the functionality of the Quarter class's constructor with a Date parameter by checking if an IllegalArgumentException is thrown when a date before 1900 is passed as a parameter.		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	A Date object representing December 30, 1899.		
Procedural Steps:	1. Create a Date object representing December 30, 1899. 2. Call the arrange() method, passing the Date object as a parameter to set up the test environment. 3. Call the constructor of the Quarter class with the Date object as a parameter.		
Expected Results of Case:	An IllegalArgumentException should be thrown.		
ACTUAL RESULTS			
Output Specifications and comments :	● The test passed successfully. ● An IllegalArgumentException was thrown as expected. ● update the documentation to include any expected exceptions and the circumstances under which they may be thrown.		

GENERAL INFORMATION			
Test Function Name :	testQuarterDateAndTimeZoneCtor	Test Case Number:	#4
Test Case Description:	This test case will verify the functionality of the Quarter class's constructor with a Date and TimeZone parameter by checking if the year and quarter values are set correctly..		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	● A Date object representing October 1, 2022. ● A TimeZone object representing GMT.		
Procedural Steps:	1. Create a Date object representing October 1, 2022. 2. Create a TimeZone object representing GMT. 3. Call the arrange() method, passing the Date and TimeZone objects as parameters to set up the test environment. 4. Retrieve the year and quarter values using the getYear() and getQuarter() methods of the Quarter class.		
Expected Results of Case:	The year value should be set to 2022 and the quarter value should be set to 4.		
ACTUAL RESULTS			
Output Specifications and comments :	● The test passed successfully. ● The year value was set to 2022 as expected. ● The quarter value was set to 4 as expected		

GENERAL INFORMATION			
Test Function Name :	testQuarterDateAndTimeZoneCtorBelow1900	Test Case Number:	#5
Test Case Description:	This test case will verify the functionality of the Quarter class's constructor with a Date and TimeZone parameter when the Date object represents a date earlier than 1900, by checking if an IllegalArgumentException is thrown.		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	● A Date object representing December 30, 1899. ● A TimeZone object representing GMT.		
Procedural Steps:	1. Create a Date object representing December 30, 1899. 2. Create a TimeZone object representing GMT. 3. Call the arrange() method, passing the Date and TimeZone objects as parameters to set up the test environment. 4. Attempt to create a Quarter object using the constructor. 5. Expect an IllegalArgumentException to be thrown.		
Expected Results of Case:	An IllegalArgumentException should be thrown.		
ACTUAL RESULTS			
Output Specifications and comments :	● The test passed successfully. ● An IllegalArgumentException was thrown as expected. ● update the documentation to include any expected exceptions and the circumstances under which they may be thrown.		

GENERAL INFORMATION			
Test Function Name :	testQuarterCtor4	Test Case Number:	#6
Test Case Description:	Test whether the fourth constructor is working correctly.		
Results:	☑ Pass ☐Fail		
TEST			
Input Specifications:	Quartile Number (1) and Year(2023)		
Procedural Steps:	1. Call the arrange method to set up the test environment. 2. Call the fourth constructor with the given input. 3. Check whether the year and quartile number are the same.		
Expected Results of Case:	Same year and quartile number.		
ACTUAL RESULTS			
Output Specifications and comments :	● Test passed successfully. ● Same quartile number. ● Same year.		

GENERAL INFORMATION			
Test Function Name :	testQuarterCtor4LessThanOne	Test Case Number:	#7
Test Case Description:	Tests that the constructor shouldn't accept a quartile number less than 1.		
Results:	☐Pass ☑Fail		
TEST			
Input Specifications:	Quartile Number (0) and Year(1900)		
Procedural Steps:	1. Call the arrange method to set up the test environment. 2. Call the fourth constructor with the given input. 3. Check whether the quartile number is not the same.		
Expected Results of Case:	The quartile number shouldn't be the same..		
ACTUAL RESULTS			
Output Specifications and comments :	The constructor accepted the wrong quartile number.		

GENERAL INFORMATION			
Test Function Name :	testQuarterCtor4MoreThanFour	Test Case Number:	#8
Test Case Description:	Tests that the constructor shouldn't accept a quartile number more than 4.		
Results:	☐Pass ☑Fail		
TEST			
Input Specifications:	Quartile Number (5) and Year(1900)		
Procedural Steps:	1. Call the arrange method to set up the test environment. 2. Call the fourth constructor with the given input. 3. Check whether the quartile number is not the same.		
Expected Results of Case:	The quartile number shouldn't be the same..		
ACTUAL RESULTS			
Output Specifications and comments :	The constructor accepted the wrong quartile number.		

GENERAL INFORMATION			
Test Function Name :	testQuarterCtor4LessThan1900	Test Case Number:	#9
Test Case Description:	Tests that the constructor shouldn't accept a year less than 1900.		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	Quartile Number (2) and Year(1899)		
Procedural Steps:	1. Call the arrange method to set up the test environment. 2. Call the fourth constructor with the given input. 3. Check whether the year is not the same.		
Expected Results of Case:	The year shouldn't be the same..		
ACTUAL RESULTS			
Output Specifications and comments :	The constructor didn't accept the given year and it threw IllegalArgumentException which was not written in the documentation.		

GENERAL INFORMATION			
Test Function Name :	testQuarterCtor4MoreThan9999	Test Case Number:	#10
Test Case Description:	Tests that the constructor shouldn't accept a year more than 9999.		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	Quartile Number (2) and Year(10000)		
Procedural Steps:	1. Call the arrange method to set up the test environment. 2. Call the fourth constructor with the given input. 3. Check whether the year is not the same.		
Expected Results of Case:	The year shouldn't be the same..		
ACTUAL RESULTS			
Output Specifications and comments :	The constructor didn't accept the given year and it threw IllegalArgumentException which was not written in the documentation.		

GENERAL INFORMATION			
Test Function Name :	testQuarterCtor5	Test Case Number:	#11
Test Case Description:	Test whether the fifth constructor is working correctly.		
Results:	☑ Pass ☐Fail		
TEST			
Input Specifications:	Quartile Number (1) and Year Object(Current Year)		

Procedural Steps:	1. Call the arrange method to set up the test environment. 2. Call the fifth constructor with the given input. 3. Check whether the year and quartile number are the same.
Expected Results of Case:	Same year and quartile number.
ACTUAL RESULTS	
Output Specifications and comments :	● Test passed successfully. ● Same quartile number. ● Same year.

GENERAL INFORMATION			
Test Function Name :	testQuarterCtor5LessThanOne	Test Case Number:	#12
Test Case Description:	Tests that the constructor shouldn't accept a quartile number less than 1.		
Results:	☐Pass ☑Fail		
TEST			
Input Specifications:	Quartile Number (0) and Year Object(1900)		
Procedural Steps:	1. Call the arrange method to set up the test environment. 2. Call the fourth constructor with the given input. 3. Check whether the quartile number is not the same.		
Expected Results of Case:	The quartile number shouldn't be the same..		
ACTUAL RESULTS			
Output Specifications and comments :	The constructor accepted the wrong quartile number.		

GENERAL INFORMATION			
Test Function Name :	testQuarterCtor5MoreThanFour	Test Case Number:	#13
Test Case Description:	Tests that the constructor shouldn't accept a quartile number more than 4.		
Results:	☐Pass ☑Fail		
TEST			
Input Specifications:	Quartile Number (5) and Year Object(1900)		
Procedural Steps:	1. Call the arrange method to set up the test environment. 2. Call the fourth constructor with the given input. 3. Check whether the quartile number is not the same.		
Expected Results of Case:	The quartile number shouldn't be the same..		
ACTUAL RESULTS			

Output Specifications and comments :	The constructor accepted the wrong quartile number.

GENERAL INFORMATION			
Test Function Name :	testQuarterCtor5LessThan1900	Test Case Number:	#14
Test Case Description:	Tests that the constructor shouldn't accept a year less than 1900.		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	Quartile Number (2) and Year Object(1899)		
Procedural Steps:	1. Call the arrange method to set up the test environment. 2. Call the fourth constructor with the given input. 3. Check whether the year is not the same.		
Expected Results of Case:	The year shouldn't be the same..		
ACTUAL RESULTS			
Output Specifications and comments :	The constructor didn't accept the given year and it threw IllegalArgumentException which was not written in the documentation.		

GENERAL INFORMATION			
Test Function Name :	testQuarterCtor5MoreThan9999	Test Case Number:	#15
Test Case Description:	Tests that the constructor shouldn't accept a year more than 9999.		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	Quartile Number (2) and Year Object(10000)		
Procedural Steps:	1. Call the arrange method to set up the test environment. 2. Call the fourth constructor with the given input. 3. Check whether the year is not the same.		
Expected Results of Case:	The year shouldn't be the same..		
ACTUAL RESULTS			
Output Specifications and comments :	The constructor didn't accept the given year and it threw IllegalArgumentException which was not written in the documentation.		

GENERAL INFORMATION			
Test Function Name :	testQuarterCompareToLessThanEqualYears	Test Case Number:	#16
Test Case Description:	This test case will verify the functionality of the Quarter class's compareTo() method by testing whether a Quarter object representing the second quarter of 2022 is less than a Quarter object representing the third quarter of 2022.		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	● A Quarter object representing the second quarter of 2022. ● A Quarter object representing the third quarter of 2022.		
Procedural Steps:	1. Create a Quarter object representing the second quarter of 2022. 2. Create a Quarter object representing the third quarter of 2022. 3. Call the compareTo() method on the first Quarter object, passing the second Quarter object as a parameter. 4. Assert that the result of the compareTo() method call is less than 0.		
Expected Results of Case:	The Quarter object representing the second quarter of 2022 should be less than the Quarter object representing the third quarter of 2022, as the compareTo() method should return a negative integer.		
ACTUAL RESULTS			
Output Specifications and comments :	● The test passed successfully. ● The Quarter object representing the second quarter of 2022 is less than the Quarter object representing the third quarter of 2022, as expected.		

GENERAL INFORMATION			
Test Function Name :	testQuarterCompareToLessThanDifferentYears	Test Case Number:	#17
Test Case Description:	This test case will verify the functionality of the Quarter class's "compareTo" method when comparing two Quarter objects with different years, where the year of the first Quarter object is less than the year of the second Quarter object.		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	Two Quarter objects are created, one for the second quarter of 2022 and one for the third quarter of 2023.		
Procedural Steps:	1. Create a Quarter object for the second quarter of 2022. 2. Create a Quarter object for the third quarter of 2023. 3. Compare the two Quarter objects using the "compareTo" method.		
Expected Results of Case:	The test case is expected to pass if the "compareTo" method returns a negative integer, indicating that the first Quarter object is less than the second Quarter object.		
ACTUAL RESULTS			
Output Specifications and comments :	The test case passed without any issues, indicating that the "compareTo" method of the Quarter class works correctly.		

GENERAL INFORMATION			
Test Function Name :	testQuarterCompareToLessThanDifferentYears2	Test Case Number:	#18
Test Case Description:	This test case will verify the functionality of the Quarter class's "compareTo" method when comparing two Quarter objects with different years, where the year of the first Quarter object is greater than the year of the second Quarter object.		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	Two Quarter objects are created, one for the second quarter of 2022 and one for the first quarter of 2023.		
Procedural Steps:	1. Create a Quarter object for the second quarter of 2022. 2. Create a Quarter object for the first quarter of 2023. 3. Compare the two Quarter objects using the "compareTo" method.		
Expected Results of Case:	The test case is expected to pass if the "compareTo" method returns a negative integer, indicating that the 2022 object is less than the2023 object.		
ACTUAL RESULTS			
Output Specifications and comments :	The test case passed without any issues, indicating that the "compareTo" method of the Quarter class works correctly.		

GENERAL INFORMATION			
Test Function Name :	testQuarterCompareToGreaterThanEqualYears	Test Case Number:	#19
Test Case Description:	This test case verifies the functionality of the compareTo method when comparing two Quarter objects that are in the same year and have different quarters.		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	Two Quarter objects, q1 and q2, that are in the same year but have different quarters.		
Procedural Steps:	1. Create a Quarter object q1 for the second quarter of 2022. 2. Create a Quarter object q2 for the third quarter of 2022. 3. Call the compareTo method on q2, passing q1 as an argument. 4. Assert that the result of the compareTo method is greater than 0.		
Expected Results of Case:	The expected outcome of this test case is for the assertion to pass, indicating that the compareTo method correctly determined that q2 is greater than q1.		
ACTUAL RESULTS			
Output Specifications and comments :	The test case passed as expected.		

GENERAL INFORMATION			
Test Function Name :	testQuarterCompareToGreaterThanDifferentYears1	Test Case Number:	#20
Test Case Description:	This test case checks if the compareTo() method of the Quarter class works as expected for two quarters that belong to different years where the second quarter has a greater quarter number than the first quarter.		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	Two Quarter objects, q1 and q2, that are in different years and have different quarters.		
Procedural Steps:	1. Create a Quarter object q1 for the second quarter of 2022. 2. Create a Quarter object q2 for the third quarter of 2023. 3. Invoke the compareTo() method of q2 passing q1 as a parameter.		
Expected Results of Case:	The test case expects that the compareTo() method returns a positive integer, indicating that q2 is greater than q1.		
ACTUAL RESULTS			
Output Specifications and comments :	The test case passed successfully, and the actual result matched the expected result. The compareTo() method returned a positive integer indicating that q2 is greater than q1.		

GENERAL INFORMATION			
Test Function Name :	testQuarterCompareToGreaterThanDifferentYears2	Test Case Number:	#21
Test Case Description:	This test case is to verify if the Quarter.compareTo() method is correctly comparing two Quarter objects based on their year and quarter.		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	Two Quarter objects with different years and quarters.		
Procedural Steps:	1. Create two Quarter objects, q1 and q2, with different years and quarters. 2. Call q2.compareTo(q1). 3. Assert that the result of the comparison is greater than 0.		
Expected Results of Case:	The expected result is that the comparison of q2 and q1 should return a value greater than 0, indicating that q2 is greater than q1.		
ACTUAL RESULTS			
Output Specifications and comments :	The test passed successfully, and the actual result matched the expected result. The comparison of q2 and q1 returned a value greater than 0, indicating that q2 is greater than q1.		

GENERAL INFORMATION			
Test Function Name :	testQuarterCompareToEqualToEqualYears	Test Case Number:	#22
Test Case Description:	This test case checks if the compareTo method of the Quarter class returns the expected result when comparing two Quarter objects with equal quarters and equal years.		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	Two Quarter objects are created with the same quarter and year.		
Procedural Steps:	1. Create two Quarter objects with the same quarter and year. 2. Call the compareTo method of one Quarter object and pass the other Quarter object as an argument.		
Expected Results of Case:	The expected result is that the compareTo method should return 0, indicating that both Quarter objects are equal.		
ACTUAL RESULTS			
Output Specifications and comments :	The test passed without any issues.		

GENERAL INFORMATION			
Test Function Name :	testQuarterCompareToEqualToDifferentYears	Test Case Number:	#23
Test Case Description:	This test case is designed to verify the functionality of the compareTo method in the Quarter class when comparing quarters with equal and different years.		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	Two Quarter objects are created, one with a specific quarter and year, and the other with a different quarter and year.		
Procedural Steps:	1. Create a Quarter object for a specific quarter and year. 2. Create another Quarter object for a different quarter and year. 3. Call the compareTo method on the first Quarter object, passing the second Quarter object as an argument. 4. Verify the result of the compareTo method call.		
Expected Results of Case:	When comparing two Quarter objects with different years, the expected result is a non-zero integer value, indicating that the two objects are not equal. When comparing two Quarter objects with the same year, the expected result is zero, indicating that the two objects are equal.		
ACTUAL RESULTS			
Output Specifications and comments :	The test passed without any issues.		

GENERAL INFORMATION			
Test Function Name :	testQuarterEqualsSame	Test Case Number:	#24
Test Case Description:	This test case aims to check whether the equals() method returns true for two Quarter objects that are equal.		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	Two Quarter objects with the same quarter and year.		
Procedural Steps:	1. Create a Quarter object for the second quarter of 2023. 2. Create another Quarter object for the second quarter of 2023. 3. Call equals() method on q1 with q2 as an argument.		
Expected Results of Case:	The test is expected to pass, and the equals() method should return true.		
ACTUAL RESULTS			
Output Specifications and comments :	The test has passed successfully, and the equals() method has returned true for q1 and q2.		

GENERAL INFORMATION			
Test Function Name :	testQuarterEqualsDifferent	Test Case Number:	#25
Test Case Description:	The purpose of this test case is to verify the functionality of the Quarter equals() method when comparing two Quarter objects.		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	Two Quarter objects with different or same quarter and year values.		
Procedural Steps:	1. Create a Quarter object with a specified quarter and year. 2. Create another Quarter object with different or same quarter and year values. 3. Invoke equals() method on the first Quarter object, passing the second Quarter object as an argument. 4. Assert that the returned value is true if the two Quarter objects have the same quarter and year values, otherwise false.		
Expected Results of Case:	The equals() method should return true when two Quarter objects have the same quarter and year values, and false otherwise.		
ACTUAL RESULTS			
Output Specifications and comments :	The testQuarterEqualsDifferent() test case passed, indicating that the equals() method correctly identifies two Quarter objects with different quarter and year values as not equal.		

GENERAL INFORMATION			
Test Function Name :	testQuarterEqualsSameQDiffObj	Test Case Number:	#26
Test Case Description:	This test case verifies that two Quarter objects with the same quarter and year attributes but different object references are considered equal. Additionally, it ensures that Quarter objects are not equal to null.		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	The inputs for this test case are two Quarter objects, q1 and q2, with the same quarter and year attributes.		
Procedural Steps:	1. Create a Quarter object q1 for the second quarter of 2023. 2. Create another Quarter object q2 for the second quarter of 2023. 3. Assert that q1 is equal to q2 using the equals() method. 4. Assert that q1 is not equal to null.		
Expected Results of Case:	The expected result of this test case is that q1 is equal to q2 and q1 is not equal to null.		
ACTUAL RESULTS			
Output Specifications and comments :	● The equals() method returns true when comparing q1 and q2, indicating that they are considered equal. ● The assertFalse() method returns false when checking if q1 is equal to null, as expected. Therefore, this test case passes		

GENERAL INFORMATION			
Test Function Name :	testQuarterEqualsDiffQDiffObj	Test Case Number:	#27
Test Case Description:	This test case verifies that two Quarter objects with the different quarter and year attributes and different object references are considered equal. Additionally, it ensures that Quarter objects are equal.		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	The inputs for this test case are two Quarter objects, q1 and q2, with the same quarter and year attributes.		
Procedural Steps:	1. Create a Quarter object q1 for the second quarter of 2023. 2. Create another Quarter object q2 for the forth quarter of 2024. 3. Assert that q1 is not equal to q2 using the equals() method. 4. Assert that q1 is not equal to string.		
Expected Results of Case:	The expected result of this test case is that q1 is not equal to q2 and q1 is not equal to string.		
ACTUAL RESULTS			
Output Specifications and comments :	● The equals() method returns false when comparing q1 and q2, indicating that they are considered not equal. ● The assertFalse() method returns false when checking if q1 is equal to string, as expected. Therefore, this test case passes		

GENERAL INFORMATION			
Test Function Name :	testGetFirstMillisecondYear	Test Case Number:	#28
Test Case Description:	This test case will test the functionality of the getFirstMillisecond() method of the Quarter class.		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	● Two integer values representing the quarter and year to be passed to the arrange() method ● An instance of the Calendar class initialized to the "GMT" time zone		
Procedural Steps:	1. Call the arrange() method to create a Quarter object representing the specified quarter and year. 2. Create an instance of the Calendar class set to the "GMT" time zone. 3. Call the getFirstMillisecond() method of the Quarter object passing in the Calendar instance as a parameter. 4. Set the Calendar object to the time represented by the value returned by the getFirstMillisecond() method. 5. Check that the year of the Calendar object is equal to the year of the Quarter object and that it is set to April 1st at 00:00:00.000 in the "GMT" time zone.		
Expected Results of Case:	The test case is expected to pass if the Calendar object is set to April 1st, 2023 at 00:00:00.000 in the "GMT" time zone, and if the year of the Calendar object is equal to the year of the Quarter object.		
ACTUAL RESULTS			
Output Specifications and comments :	The test case passed as expected. The year of the Calendar object was equal to the year of the Quarter object.		

GENERAL INFORMATION			
Test Function Name :	testGetFirstMillisecondMonth	Test Case Number:	#29
Test Case Description:	This test case will test the functionality of the getFirstMillisecond() method of the Quarter class.		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	● Two integer values representing the quarter and year to be passed to the arrange() method ● An instance of the Calendar class initialized to the "GMT" time zone		
Procedural Steps:	1. Call the arrange() method to create a Quarter object representing the specified quarter and year. 2. Create an instance of the Calendar class set to the "GMT" time zone. 3. Call the getFirstMillisecond() method of the Quarter object passing in the Calendar instance as a parameter. 4. Set the Calendar object to the time represented by the value returned by the getFirstMillisecond() method.		

	5. Check that the month of the Calendar object is equal to the month of the Quarter object and that it is set to April 1st at 00:00:00.000 in the "GMT" time zone.
Expected Results of Case:	The test case is expected to pass if the Calendar object is set to April 1st, 2023 at 00:00:00.000 in the "GMT" time zone, and if the month of the Calendar object is equal to the month of the Quarter object.
ACTUAL RESULTS	
Output Specifications and comments :	The test case passed as expected. The month of the Calendar object was equal to the month of the Quarter object.

GENERAL INFORMATION			
Test Function Name :	testGetFirstMillisecondDay	Test Case Number:	#30
Test Case Description:	This test case will test the functionality of the getFirstMillisecond() method of the Quarter class.		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	● Two integer values representing the quarter and year to be passed to the arrange() method ● An instance of the Calendar class initialized to the "GMT" time zone		
Procedural Steps:	1. Call the arrange() method to create a Quarter object representing the specified quarter and year. 2. Create an instance of the Calendar class set to the "GMT" time zone. 3. Call the getFirstMillisecond() method of the Quarter object passing in the Calendar instance as a parameter. 4. Set the Calendar object to the time represented by the value returned by the getFirstMillisecond() method. 5. Check that the day of the Calendar object is equal to the day of the Quarter object and that it is set to April 1st at 00:00:00.000 in the "GMT" time zone.		
Expected Results of Case:	The test case is expected to pass if the Calendar object is set to April 1st, 2023 at 00:00:00.000 in the "GMT" time zone, and if the day of the Calendar object is equal to the day of the Quarter object.		
ACTUAL RESULTS			
Output Specifications and comments :	The test case passed as expected. The day of the Calendar object was equal to the day of the Quarter object.		

GENERAL INFORMATION			
Test Function Name :	testGetFirstMillisecondHour	Test Case Number:	#31
Test Case Description:	This test case will test the functionality of the getFirstMillisecond() method of the Quarter class.		
Results:	☑Pass ☐Fail		
TEST			

Input Specifications:	● Two integer values representing the quarter and year to be passed to the arrange() method ● An instance of the Calendar class initialized to the "GMT" time zone
Procedural Steps:	1. Call the arrange() method to create a Quarter object representing the specified quarter and year. 2. Create an instance of the Calendar class set to the "GMT" time zone. 3. Call the getFirstMillisecond() method of the Quarter object passing in the Calendar instance as a parameter. 4. Set the Calendar object to the time represented by the value returned by the getFirstMillisecond() method. 5. Check that the hour of the Calendar object is equal to the hour of the Quarter object and that it is set to April 1st at 00:00:00.000 in the "GMT" time zone.
Expected Results of Case:	The test case is expected to pass if the Calendar object is set to April 1st, 2023 at 00:00:00.000 in the "GMT" time zone, and if the hour of the Calendar object is equal to the hour of the Quarter object.
ACTUAL RESULTS	
Output Specifications and comments :	The test case passed as expected. The hour of the Calendar object was equal to the hour of the Quarter object.

GENERAL INFORMATION			
Test Function Name :	testGetFirstMillisecondMinute	Test Case Number:	#32
Test Case Description:	This test case will test the functionality of the getFirstMillisecond() method of the Quarter class.		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	● Two integer values representing the quarter and year to be passed to the arrange() method ● An instance of the Calendar class initialized to the "GMT" time zone		
Procedural Steps:	1. Call the arrange() method to create a Quarter object representing the specified quarter and year. 2. Create an instance of the Calendar class set to the "GMT" time zone. 3. Call the getFirstMillisecond() method of the Quarter object passing in the Calendar instance as a parameter. 4. Set the Calendar object to the time represented by the value returned by the getFirstMillisecond() method. 5. Check that the minute of the Calendar object is equal to the minute of the Quarter object and that it is set to April 1st at 00:00:00.000 in the "GMT" time zone.		
Expected Results of Case:	The test case is expected to pass if the Calendar object is set to April 1st, 2023 at 00:00:00.000 in the "GMT" time zone, and if the minute of the Calendar object is equal to the minute of the Quarter object.		
ACTUAL RESULTS			
Output Specifications and comments :	The test case passed as expected. The minute of the Calendar object was equal to the hour of the Quarter object.		

GENERAL INFORMATION			
Test Function Name :	testGetFirstMillisecondSecond	Test Case Number:	#33
Test Case Description:	This test case will test the functionality of the getFirstMillisecond() method of the Quarter class.		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	● Two integer values representing the quarter and year to be passed to the arrange() method ● An instance of the Calendar class initialized to the "GMT" time zone		
Procedural Steps:	1. Call the arrange() method to create a Quarter object representing the specified quarter and year. 2. Create an instance of the Calendar class set to the "GMT" time zone. 3. Call the getFirstMillisecond() method of the Quarter object passing in the Calendar instance as a parameter. 4. Set the Calendar object to the time represented by the value returned by the getFirstMillisecond() method. 5. Check that the seconds of the Calendar object is equal to the seconds of the Quarter object and that it is set to April 1st at 00:00:00.000 in the "GMT" time zone.		
Expected Results of Case:	The test case is expected to pass if the Calendar object is set to April 1st, 2023 at 00:00:00.000 in the "GMT" time zone, and if the seconds of the Calendar object is equal to the seconds of the Quarter object.		
ACTUAL RESULTS			
Output Specifications and comments :	The test case passed as expected. The seconds of the Calendar object was equal to the seconds of the Quarter object.		

GENERAL INFORMATION			
Test Function Name :	testGetFirstMillisecond	Test Case Number:	#34
Test Case Description:	This test case will test the functionality of the getFirstMillisecond() method of the Quarter class.		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	● Two integer values representing the quarter and year to be passed to the arrange() method ● An instance of the Calendar class initialized to the "GMT" time zone		
Procedural Steps:	1. Call the arrange() method to create a Quarter object representing the specified quarter and year. 2. Create an instance of the Calendar class set to the "GMT" time zone. 3. Call the getFirstMillisecond() method of the Quarter object passing in the Calendar instance as a parameter. 4. Set the Calendar object to the time represented by the value returned by the getFirstMillisecond() method. 5. Check that the milliseconds of the Calendar object is equal to the milliseconds of the Quarter object and that it is set to April 1st at 00:00:00.000 in the "GMT" time zone.		

Expected Results of Case:	The test case is expected to pass if the Calendar object is set to April 1st, 2023 at 00:00:00.000 in the "GMT" time zone, and if the milliseconds of the Calendar object is equal to the milliseconds of the Quarter object.
ACTUAL RESULTS	
Output Specifications and comments :	The test case passed as expected. The milliseconds of the Calendar object was equal to the milliseconds of the Quarter object.

GENERAL INFORMATION			
Test Function Name :	testGetLastMillisecondYear	Test Case Number:	#35
Test Case Description:	This test case will test the functionality of the getLastMillisecond() method of the Quarter class.		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	• Two integer values representing the quarter and year to be passed to the arrange() method • An instance of the Calendar class initialized to the "GMT" time zone		
Procedural Steps:	1. Call the arrange() method to create a Quarter object representing the specified quarter and year. 2. Create an instance of the Calendar class set to the "GMT" time zone. 3. Call the getLastMillisecond() method of the Quarter object passing in the Calendar instance as a parameter. 4. Set the Calendar object to the time represented by the value returned by the getLastMillisecond() method. 5. Check that the year of the Calendar object is equal to the year of the Quarter object and that it is set to June 30th, 2023 at 23:59:59.999 in the "GMT" time zone.		
Expected Results of Case:	The test case is expected to pass if the Calendar object is set to June 30th, 2023 at 23:59:59.999 in the "GMT" time zone, and if the year of the Calendar object is equal to the year of the Quarter object.		
ACTUAL RESULTS			
Output Specifications and comments :	The test case passed as expected. The year of the Calendar object was equal to the year of the Quarter object.		

GENERAL INFORMATION			
Test Function Name :	testGetLastMillisecondMonth	Test Case Number:	#36
Test Case Description:	This test case will test the functionality of the getLastMillisecond() method of the Quarter class.		
Results:	☑Pass ☐Fail		
TEST			

Input Specifications:	● Two integer values representing the quarter and year to be passed to the arrange() method ● An instance of the Calendar class initialized to the "GMT" time zone
Procedural Steps:	1. Call the arrange() method to create a Quarter object representing the specified quarter and year. 2. Create an instance of the Calendar class set to the "GMT" time zone. 3. Call the getLastMillisecond() method of the Quarter object passing in the Calendar instance as a parameter. 4. Set the Calendar object to the time represented by the value returned by the getLastMillisecond() method. 5. Check that the month of the Calendar object is equal to the month of the Quarter object and that it is set to June 30th, 2023 at 23:59:59.999 in the "GMT" time zone.
Expected Results of Case:	The test case is expected to pass if the Calendar object is set to June 30th, 2023 at 23:59:59.999 in the "GMT" time zone, and if the month of the Calendar object is equal to the month of the Quarter object.
ACTUAL RESULTS	
Output Specifications and comments :	The test case passed as expected. The month of the Calendar object was equal to the month of the Quarter object.

GENERAL INFORMATION			
Test Function Name :	testGetLastMillisecondDay	Test Case Number:	#37
Test Case Description:	This test case will test the functionality of the getLastMillisecond() method of the Quarter class.		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	● Two integer values representing the quarter and year to be passed to the arrange() method ● An instance of the Calendar class initialized to the "GMT" time zone		
Procedural Steps:	1. Call the arrange() method to create a Quarter object representing the specified quarter and year. 2. Create an instance of the Calendar class set to the "GMT" time zone. 3. Call the getLastMillisecond() method of the Quarter object passing in the Calendar instance as a parameter. 4. Set the Calendar object to the time represented by the value returned by the getLastMillisecond() method. 5. Check that the day of the Calendar object is equal to the day of the Quarter object and that it is set to June 30th, 2023 at 23:59:59.999 in the "GMT" time zone.		
Expected Results of Case:	The test case is expected to pass if the Calendar object is set to June 30th, 2023 at 23:59:59.999 in the "GMT" time zone, and if the day of the Calendar object is equal to the day of the Quarter object.		
ACTUAL RESULTS			
Output Specifications and comments :	The test case passed as expected. The day of the Calendar object was equal to the day of the Quarter object.		

GENERAL INFORMATION			
Test Function Name :	testGetLastMillisecondHour	Test Case Number:	#38
Test Case Description:	This test case will test the functionality of the getLastMillisecond() method of the Quarter class.		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	● Two integer values representing the quarter and year to be passed to the arrange() method ● An instance of the Calendar class initialized to the "GMT" time zone		
Procedural Steps:	1. Call the arrange() method to create a Quarter object representing the specified quarter and year. 2. Create an instance of the Calendar class set to the "GMT" time zone. 3. Call the getLastMillisecond() method of the Quarter object passing in the Calendar instance as a parameter. 4. Set the Calendar object to the time represented by the value returned by the getLastMillisecond() method. 5. Check that the hour of the Calendar object is equal to the hour of the Quarter object and that it is set to June 30th, 2023 at 23:59:59.999 in the "GMT" time zone.		
Expected Results of Case:	The test case is expected to pass if the Calendar object is set to June 30th, 2023 at 23:59:59.999 in the "GMT" time zone, and if the hour of the Calendar object is equal to the hour of the Quarter object.		
ACTUAL RESULTS			
Output Specifications and comments :	The test case passed as expected. The hour of the Calendar object was equal to the hour of the Quarter object.		

GENERAL INFORMATION			
Test Function Name :	testGetLastMillisecondMinutes	Test Case Number:	#39
Test Case Description:	This test case will test the functionality of the getLastMillisecond() method of the Quarter class.		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	● Two integer values representing the quarter and year to be passed to the arrange() method ● An instance of the Calendar class initialized to the "GMT" time zone		
Procedural Steps:	1. Call the arrange() method to create a Quarter object representing the specified quarter and year. 2. Create an instance of the Calendar class set to the "GMT" time zone. 3. Call the getLastMillisecond() method of the Quarter object passing in the Calendar instance as a parameter. 4. Set the Calendar object to the time represented by the value returned by the getLastMillisecond() method.		

	5. Check that the minutes of the Calendar object is equal to the minutes of the Quarter object and that it is set to June 30th, 2023 at 23:59:59.999 in the "GMT" time zone.
Expected Results of Case:	The test case is expected to pass if the Calendar object is set to June 30th, 2023 at 23:59:59.999 in the "GMT" time zone, and if the minutes of the Calendar object is equal to the minutes of the Quarter object.
ACTUAL RESULTS	
Output Specifications and comments :	The test case passed as expected. The minutes of the Calendar object was equal to the minutes of the Quarter object.

GENERAL INFORMATION			
Test Function Name :	testGetLastMillisecondSeconds	Test Case Number:	#40
Test Case Description:	This test case will test the functionality of the getLastMillisecond() method of the Quarter class.		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	● Two integer values representing the quarter and year to be passed to the arrange() method ● An instance of the Calendar class initialized to the "GMT" time zone		
Procedural Steps:	1. Call the arrange() method to create a Quarter object representing the specified quarter and year. 2. Create an instance of the Calendar class set to the "GMT" time zone. 3. Call the getLastMillisecond() method of the Quarter object passing in the Calendar instance as a parameter. 4. Set the Calendar object to the time represented by the value returned by the getLastMillisecond() method. 5. Check that the seconds of the Calendar object is equal to the seconds of the Quarter object and that it is set to June 30th, 2023 at 23:59:59.999 in the "GMT" time zone.		
Expected Results of Case:	The test case is expected to pass if the Calendar object is set to June 30th, 2023 at 23:59:59.999 in the "GMT" time zone, and if the seconds of the Calendar object is equal to the seconds of the Quarter object.		
ACTUAL RESULTS			
Output Specifications and comments :	The test case passed as expected. The seconds of the Calendar object was equal to the seconds of the Quarter object.		

GENERAL INFORMATION			
Test Function Name :	testGetLastMillisecond	Test Case Number:	#41
Test Case Description:	This test case will test the functionality of the getLastMillisecond() method of the Quarter class.		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	• Two integer values representing the quarter and year to be passed to the arrange() method • An instance of the Calendar class initialized to the "GMT" time zone		
Procedural Steps:	1. Call the arrange() method to create a Quarter object representing the specified quarter and year. 2. Create an instance of the Calendar class set to the "GMT" time zone. 3. Call the getLastMillisecond() method of the Quarter object passing in the Calendar instance as a parameter. 4. Set the Calendar object to the time represented by the value returned by the getLastMillisecond() method. 5. Check that the year of the Calendar object is equal to the year of the Quarter object and that it is set to June 30th, 2023 at 23:59:59.999 in the "GMT" time zone.		
Expected Results of Case:	The test case is expected to pass if the Calendar object is set to June 30th, 2023 at 23:59:59.999 in the "GMT" time zone, and if the milliseconds of the Calendar object is equal to the milliseconds of the Quarter object.		
ACTUAL RESULTS			
Output Specifications and comments :	The test case passed as expected. The milliseconds of the Calendar object was equal to the milliseconds of the Quarter object.		

GENERAL INFORMATION			
Test Function Name :	testGetQuarterBelow1900	Test Case Number:	#42
Test Case Description:	Tests the Quarter.getQuarter method for a date with year below 1900		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	Date instance with year value below 1900		
Procedural Steps:	1- Create a new Date instance with year value below 1900 2- Instantiate a new Quarter given this Date object		
Expected Results of Case:	A valid Quarter object is instantiated with an integer quarter field		
ACTUAL RESULTS			
Output Specifications and comments :	A new Quarter object is instantiated with the appropriate quarter number		

GENERAL INFORMATION			
Test Function Name :	testGetQuarterFirst	Test Case Number:	#43
Test Case Description:	Tests Quarter.getQuarter method that returns the quarter number of the given date for a date in the first quarter		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	Date object of value lying in first quarter		
Procedural Steps:	1- Create a new Date instance for an arbitrary first-quarter date 2- Instantiate a new Quarter with that date 3- invoke getQuarter method of that instance		
Expected Results of Case:	The invoked method should return 1		
ACTUAL RESULTS			
Output Specifications and comments :	Returned quarter number is 1		

GENERAL INFORMATION			
Test Function Name :	testGetQuarterSecond	Test Case Number:	#44
Test Case Description:	Tests Quarter.getQuarter method that returns the quarter number of the given date for a date in the second quarter		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	Date object of value lying in second quarter		
Procedural Steps:	1- Create a new Date instance for an arbitrary second-quarter date 2- Instantiate a new Quarter with that date 3- invoke getQuarter method of thatinstance		
Expected Results of Case:	The invoked method should return 2		
ACTUAL RESULTS			
Output Specifications and comments :	Returned quarter number is 2		

GENERAL INFORMATION

Test Function Name :	testGetQuarterThird	Test Case Number:	#45
Test Case Description:	Tests Quarter.getQuarter method that returns the quarter number of the given date for a date in the third quarter		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	Date object of value lying in third quarter		
Procedural Steps:	1- Create a new Date instance for an arbitrary third-quarter date 2- Instantiate a new Quarter with that date 3- invoke getQuarter method of that instance		
Expected Results of Case:	The invoked method should return 3		
ACTUAL RESULTS			
Output Specifications and comments :	Returned quarter number is 3		

GENERAL INFORMATION			
Test Function Name :	testGetQuarterForth	Test Case Number:	#46
Test Case Description:	Tests Quarter.getQuarter method that returns the quarter number of the given date for a date in the fourth quarter		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	Date object of value lying in fourth quarter		
Procedural Steps:	1- Create a new Date instance for an arbitrary fourth-quarter date 2- Instantiate a new Quarter with that date 3- invoke getQuarter method of that instance		
Expected Results of Case:	The invoked method should return 4		
ACTUAL RESULTS			
Output Specifications and comments :	Returned quarter number is 4		

GENERAL INFORMATION			
Test Function Name :	testGetSerialIndexQ1	Test Case Number:	#47
Test Case Description:	Tests Quarter.getSerialIndex method with a first-quarter Quarter instance		

Results:	☑Pass ☐Fail
TEST	
Input Specifications:	Quarter number of value 1, and year of any value
Procedural Steps:	1- Create a new instance of Quarter with any year value and quarter of value 1 2- invoke getSerialIndex for that instance
Expected Results of Case:	The invoked method should return a value of (year*4 + 1)
ACTUAL RESULTS	
Output Specifications and comments :	The returned value is the expected value (year*4+1)

GENERAL INFORMATION			
Test Function Name :	testGetSerialIndexQ2	Test Case Number:	#48
Test Case Description:	Tests Quarter.getSerialIndex method with a second-quarter Quarter instance		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	Quarter number of value 2, and year of any value		
Procedural Steps:	1- Create a new instance of Quarter with any year value and quarter of value 2 2- invoke getSerialIndex for that instance		
Expected Results of Case:	The invoked method should return a value of (year*4 + 2)		
ACTUAL RESULTS			
Output Specifications and comments :	The returned value is the expected value (year*4+2)		

GENERAL INFORMATION			
Test Function Name :	testGetSerialIndexQ3	Test Case Number:	#49
Test Case Description:	Tests Quarter.getSerialIndex method with a third-quarter Quarter instance		
Results:	☑ass ☐Fail		
TEST			
Input Specifications:	Quarter number of value 3, and year of any value		

Procedural Steps:	1- Create a new instance of Quarter with any year value and quarter of value 3 2- invoke getSerialIndex for that instance
Expected Results of Case:	The invoked method should return a value of (year*4 + 3)
ACTUAL RESULTS	
Output Specifications and comments :	The returned value is the expected value (year*4+3)

GENERAL INFORMATION			
Test Function Name :	testGetSerialIndexQ4	Test Case Number:	#50
Test Case Description:	Tests Quarter.getSerialIndex method with a fourth-quarter Quarter instance		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	Quarter number of value 4, and year of any value		
Procedural Steps:	1- Create a new instance of Quarter with any year value and quarter of value 4 2- invoke getSerialIndex for that instance		
Expected Results of Case:	The invoked method should return a value of (year*4 + 4)		
ACTUAL RESULTS			
Output Specifications and comments :	The returned value is the expected value (year*4+4)		

GENERAL INFORMATION			
Test Function Name :	testGetSerialIndexBelowMinValue	Test Case Number:	#51
Test Case Description:	Tests Quarter.getSerialIndex method with a Quarter instance of year value below 1900		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	Year value below 1900 and quarter number of any known value		
Procedural Steps:	1- create a new Quarter instance of value 1899, and quarter number of value 1 2- invoket getSerialIndex of that instance		
Expected Results of Case:	Returned value of the invoked instance should be (year*4 + quarterNumber)		
ACTUAL RESULTS			

Output Specifications and comments :	Returned value was the expected value (1899*4 + 1)

GENERAL INFORMATION			
Test Function Name :	testGetSerialIndexMinValue	Test Case Number:	#52
Test Case Description:	Tests Quarter.getSerialIndex method with a Quarter instance of year value equal to 1900		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	Year value equals 1900 and quarter number of any known value		
Procedural Steps:	1- create a new Quarter instance of value 1900, and quarter number of value 1 2- invoket getSerialIndex of that instance		
Expected Results of Case:	Returned value of the invoked instance should be (year*4 + quarterNumber)		
ACTUAL RESULTS			
Output Specifications and comments :	Returned value was the expected value (1900*4 + 1)		

GENERAL INFORMATION			
Test Function Name :	testGetSerialIndexMaxValue	Test Case Number:	#53
Test Case Description:	Tests Quarter.getSerialIndex method with a Quarter instance of year maximum value (9999) and max quarter number		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	Year value equals 9999and quarter number value 4		
Procedural Steps:	1- create a new Quarter instance of value 9999, and quarter number of value 4 2- invoket getSerialIndex of that instance		
Expected Results of Case:	Returned value of the invoked instance should be (year*4 + quarterNumber)		
ACTUAL RESULTS			
Output Specifications and comments :	Returned value was the expected value (1900*4 + 4)		

GENERAL INFORMATION			
Test Function Name :	testGetYear	Test Case Number:	#54

Test Case Description:	Tests the Quarter.getYear method with a Quarter instance of known year value
Results:	☑Pass ☐Fail
TEST	
Input Specifications:	Year instance of known year, and quarter value of any value
Procedural Steps:	1- Create a new Year instance of value 2023 2- Create a new Quarter instance using any quarter number and the forementioned Year instance 3- Invoke th getYear method
Expected Results of Case:	Returned value of the getYear method should be the same as that of the Year instance
ACTUAL RESULTS	
Output Specifications and comments :	The returned value of the invoked function is the expected value: Year(2023)

GENERAL INFORMATION			
Test Function Name :	testParseQuarterFormat1WithDash	Test Case Number:	#54
Test Case Description:	Testing the first format with dash.		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	String of the first format ("Q2-2023")		
Procedural Steps:	1. Creating parsedQuarter object using the function with the input string. 2. Using the arrange method to make Quarter of the same quarter number and year... 3. Checking whether the parsedQuarter is the same as the Quarter		
Expected Results of Case:	The parsed quarter should be the same as the quarter.		
ACTUAL RESULTS			
Output Specifications and comments :	Test passed succesfully and the parsedQuarter is the same as the quarter.		

GENERAL INFORMATION			
Test Function Name :	testHashCode	Test Case Number:	#57
Test Case Description:	Tests Quarter.hashCode function for 2 instances of Quarter having same field values		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	Year and quarter number equal values over 2 instances of Quarter		
Procedural Steps:	1- Create a Quareter instance of year value 2023, and quarter number 1 2- Create another Quareter instance of year value 2023, and quarter number 2 3- invoke getHashcode for both of them		
Expected Results of Case:	The two hash codes of the 2 instances are equal		
ACTUAL RESULTS			
Output Specifications and comments :	Two hash codes of the 2 instances are of the same value		

GENERAL INFORMATION			
Test Function Name :	testHashCodeDifferentQuarters	Test Case Number:	#58
Test Case Description:	Tests Quarter.hashCode function for 2 instances of Quarter having different quarter number values		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	2 years of same value and 2 quarter numbers of different values over 2 instances of Quarter		
Procedural Steps:	1- Create a Quareter instance of year value 2000, and quarter number 1 2- Create another Quareter instance of year value 2023, and quarter number 1 3- invoke getHashcode for both of them		
Expected Results of Case:	The two hash codes of the 2 instances are not equal		
ACTUAL RESULTS			
Output Specifications and comments :	Two hash codes of the 2 instances are of different values		

GENERAL INFORMATION			
Test Function Name :	testHashCodeDifferentYears	Test Case Number:	#59
Test Case Description:	Tests Quarter.hashCode function for 2 instances of Quarter having different year values		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	2 years of different values and 2 quarter numbers of the same value over 2 instances of Quarter		
Procedural Steps:	1- Create a Quareter instance of year value 2023, and quarter number 1 2- Create another Quareter instance of year value 2023, and quarter number 1 3- invoke getHashcode for both of them		
Expected Results of Case:	The two hash codes of the 2 instances are not equal		
ACTUAL RESULTS			
Output Specifications and comments :	Two hash codes of the 2 instances are of different values		

GENERAL INFORMATION			
Test Function Name :	testNext	Test Case Number:	#65
Test Case Description:	Testing whether the testNext function is working correctly.		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	Two Quarter Objects: Quarter Number(1) and Year(2023) Quarter Number(2) and Year(2023)		
Procedural Steps:	1. Using the arrange method to make the first Quarter. 2. Capturing the previous of it. . 3. Using the arrange method again to make the second Quarter.. 4. Checking whether the captured next is the same as the second Quarter		
Expected Results of Case:	The captured previous is the same as the second Quarter.		
ACTUAL RESULTS			
Output Specifications and comments :	Test passed succesfully and the previous of (2, 2023) is (1, 2023).		

GENERAL INFORMATION			
Test Function Name :	testNextInFourthQuarter	Test Case Number:	#66
Test Case Description:	Checking that the next of the fourth quartile of a year should also increment the year and the quartile number becomes one..		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	Two Quarter Objects: Quartile Number(4) and Year(2023) Quartile Number(1) and Year(2024)		
Procedural Steps:	1. second Quarter.Using the arrange method with the given input. 2. Defining the expected string. 3. Checking whether the expected string is the same as the string coming from toString function.		
Expected Results of Case:	The captured previous is the same as the second Quarter.Describe the outcome anticipated from the test case. Specify the criteria to be used to determine whether the item has passed or failed.		
ACTUAL RESULTS			
Output Specifications and comments :	Test passed succesfully and the previous of (1, 2024) is (4, 2023)..		

GENERAL INFORMATION			
Test Function Name :	testNextInFourthQuarterMaxYear	Test Case Number:	#67
Test Case Description:	Testing that the previous quarter of the maximum year (9999) is null.		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	Quartile Number(4) and Year(9999)		
Procedural Steps:	1. Using the arrange method with the given input. 2. Checking whether the previous is null		
Expected Results of Case:	The next quarter should be null.		
ACTUAL RESULTS			
Output Specifications and comments :	Test passed successfully and the next quarter is null.		

GENERAL INFORMATION			
Test Function Name :	testParseQuarterFormat2WithDash	Test Case Number:	#68
Test Case Description:	Testing the second format with dash.		

Results:	☑Pass ☐Fail
TEST	
Input Specifications:	String of the first format ("Q2-2023")
Procedural Steps:	1. Creating parsedQuarter object using the function with the input string. 2. Using the arrange method to make Quarter of the same quarter number and year... 3. Checking whether the parsedQuarter is the same as the Quarter
Expected Results of Case:	The parsed quarter should be the same as the quarter.
ACTUAL RESULTS	
Output Specifications and comments :	Test passed succesfully and the parsedQuarter is the same as the quarter.

GENERAL INFORMATION			
Test Function Name :	testParseQuarterFormat1WithSlash	Test Case Number:	#69
Test Case Description:	Testing the first format with slash.		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	String of the first format (""2023/Q1"")		
Procedural Steps:	1. Creating parsedQuarter object using the function with the input string. 2. Using the arrange method to make Quarter of the same quarter number and year... 3. Checking whether the parsedQuarter is the same as the Quarter		
Expected Results of Case:	The parsed quarter should be the same as the quarter.		
ACTUAL RESULTS			
Output Specifications and comments :	Test passed succesfully and the parsedQuarter is the same as the quarter.		

GENERAL INFORMATION			
Test Function Name :	testParseQuarterFormat2WithSlash	Test Case Number:	#70
Test Case Description:	Testing the second format with slash.		
Results:	☑Pass ☐Fail		
TEST			

Input Specifications:	String of the first format ("Q2/2023")
Procedural Steps:	1. Creating parsedQuarter object using the function with the input string. 2. Using the arrange method to make Quarter of the same quarter number and year... 3. Checking whether the parsedQuarter is the same as the Quarter
Expected Results of Case:	The parsed quarter should be the same as the quarter.
ACTUAL RESULTS	
Output Specifications and comments :	Test passed succesfully and the parsedQuarter is the same as the quarter.

GENERAL INFORMATION			
Test Function Name :	testParseQuarterFormat1WithComma	Test Case Number:	#71
Test Case Description:	Testing the first format with comma.		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	String of the first format ("2023,Q1")		
Procedural Steps:	1. Creating parsedQuarter object using the function with the input string. 2. Using the arrange method to make Quarter of the same quarter number and year... 3. Checking whether the parsedQuarter is the same as the Quarter		
Expected Results of Case:	The parsed quarter should be the same as the quarter.		
ACTUAL RESULTS			
Output Specifications and comments :	Test passed succesfully and the parsedQuarter is the same as the quarter.		

GENERAL INFORMATION			
Test Function Name :	testParseQuarterFormat2WithComma	Test Case Number:	#72
Test Case Description:	Testing the second format with comma.		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	String of the first format ("Q2,2023")		
Procedural Steps:	1. Creating parsedQuarter object using the function with the input string. 2. Using the arrange method to make Quarter of the same quarter number and year...		

	3. Checking whether the parsedQuarter is the same as the Quarter
Expected Results of Case:	The parsed quarter should be the same as the quarter.
ACTUAL RESULTS	
Output Specifications and comments :	Test passed succesfully and the parsedQuarter is the same as the quarter.

GENERAL INFORMATION			
Test Function Name :	testParseQuarterFormat1WithSpace	Test Case Number:	#73
Test Case Description:	Testing the first format with space.		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	String of the first format ("2023 Q1")		
Procedural Steps:	1. Creating parsedQuarter object using the function with the input string. 2. Using the arrange method to make Quarter of the same quarter number and year... 3. Checking whether the parsedQuarter is the same as the Quarter		
Expected Results of Case:	The parsed quarter should be the same as the quarter.		
ACTUAL RESULTS			
Output Specifications and comments :	Test passed succesfully and the parsedQuarter is the same as the quarter.		

GENERAL INFORMATION			
Test Function Name :	testParseQuarterFormat2WithSpace	Test Case Number:	#74
Test Case Description:	Testing the second format with space.		
Results:	☑☑Pass ☐Fail		
TEST			
Input Specifications:	String of the second format ("Q2 2023")		
Procedural Steps:	1. Creating parsedQuarter object using the function with the input string. 2. Using the arrange method to make Quarter of the same quarter number and year... 3. Checking whether the parsedQuarter is the same as the Quarter		
Expected Results of Case:	The parsed quarter should be the same as the quarter.		
ACTUAL RESULTS			

Output Specifications and comments :	Test passed succesfully and the parsedQuarter is the same as the quarter.

GENERAL INFORMATION			
Test Function Name :	testParseQuarterWithoutQ	Test Case Number:	#75
Test Case Description:	Testing when the string doesn't contain 'Q'.		
Results:	☑☑Pass ☐Fail		
TEST			
Input Specifications:	String without 'Q' ("2-2023")		
Procedural Steps:	Using .parseQuarterFunction with this input.		
Expected Results of Case:	The function shouldn't accept the given string.		
ACTUAL RESULTS			
Output Specifications and comments :	Test passed and the function threw TimePeriodFormatException but this wasn't mentioned int the documentation..		

GENERAL INFORMATION			
Test Function Name :	testParseQuarterNoNumbers	Test Case Number:	#76
Test Case Description:	Testing when the string doesn't contain numbers.		
Results:	☑☑Pass ☐Fail		
TEST			
Input Specifications:	String with no number("QN-YYYY")		
Procedural Steps:	Using .parseQuarterFunction with this input.		
Expected Results of Case:	The function shouldn't accept the given string.		
ACTUAL RESULTS			
Output Specifications and comments :	Test passed and the function threw java.lang.Exception but this wasn't mentioned int the documentation..		

GENERAL INFORMATION			
Test Function Name :	testPrevious	Test Case Number:	#77
Test Case Description:	Testing whether the testPrevious function is working correctly.		

Results:	☑☑Pass ☐Fail
TEST	
Input Specifications:	Two Quarter Objects: Quarter Number(2) and Year(2023) Quarter Number(1) and Year(2023)
Procedural Steps:	1. Using the arrange method to make the first Quarter. 2. Capturing the previous of it. . 3. Using the arrange method again to make the second Quarter.. 4. Checking whether the captured previous is the same as the second Quarter
Expected Results of Case:	The captured previous is the same as the second Quarter.
ACTUAL RESULTS	
Output Specifications and comments :	Test passed succesfully and the previous of (2, 2023) is (1, 2023).

GENERAL INFORMATION			
Test Function Name :	testPreviousInFirstQuarter	Test Case Number:	#78
Test Case Description:	Checking that the previous of the first quartile of a year should also decrement the year and the quartile number becomes four..		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	Two Quarter Objects: Quartile Number(1) and Year(2024) Quartile Number(4) and Year(2023)		
Procedural Steps:	1. second Quarter.Using the arrange method with the given input. 2. Defining the expected string. 3. Checking whether the expected string is the same as the string coming from toString function.		
Expected Results of Case:	The captured previous is the same as the second Quarter.Describe the outcome anticipated from the test case. Specify the criteria to be used to determine whether the item has passed or failed.		
ACTUAL RESULTS			
Output Specifications and comments :	Test passed succesfully and the previous of (1, 2024) is (4, 2023)..		

GENERAL INFORMATION			
Test Function Name :	testPreviousInFirstQuarterMinYear	Test Case Number:	#79
Test Case Description:	Testing that the previous quarter of the minimum year (1900) is null.		
Results:	☑Pass ☐Fail		
TEST			

Input Specifications:	Quartile Number(1) and Year(1900)
Procedural Steps:	1. Using the arrange method with the given input. 2. Checking whether the previous is null
Expected Results of Case:	The previous quarter should be null.
ACTUAL RESULTS	
Output Specifications and comments :	Test passed successfully and the previous quarter is null.

GENERAL INFORMATION			
Test Function Name :	testToString	Test Case Number:	#80
Test Case Description:	Testing whether toString function is working correctly.		
Results:	☑Pass ☐Fail		
TEST			
Input Specifications:	Quartile Number(3) and Year(2023)		
Procedural Steps:	1. Using the arrange method with the given input. 2. Defining the expected string. 3. Checking whether the expected string is the same as the string coming from toString function.		
Expected Results of Case:	The expected string should be the same as the string of toString function.		
ACTUAL RESULTS			
Output Specifications and comments :	Test passed successfully and the toString returned “Q3/2023”.		