GENERAL INFORMATION			
Test Function Name :	testQuarterDefaultCtor	Test Case Number:	#1
Test Case Description:	The case tests the functionality of default constructor to be sure that the constructor retrieves the right year and the right quarter from current date/time		
Results:	PASS		
TEST			
Input Specifications:	No inputs required		
Procedural Steps:	Just calling the default constructor		
Expected Results of Case:	Due to The current date 27/4/2023 we are in Quarter 2 from year 2023 quarter=2 , year = 2023		
ACTUAL RESULTS			
Output Specifications and comments :	The actual results match the expected result as it makes q Quarter object that has the same data of the current date quarter=2 , year = 2023		

GENERAL INFORMATION			
Test Function Name :	testQuarterConstructor2	Test Case Number:	#2
Test Case Description:	Takes an illegal time 10/10/1899 as a parameter for the constructor and it throws an exception		
Results:	PASS		
TEST			
Input Specifications:	- Time object in milliseconds - Quarter object time as parameter		
Procedural Steps:	Take a time object 10/10/1899 in milliseconds as aparameter for the constructor and it is expected to throw an Illegal argument exception		
Expected Results of Case:	Expected to throw an illegal argument exception		
ACTUAL RESULTS			
Output Specifications and comments :	IlleagalArgumentException as the expected because 1899 is out of range		

GENERAL INFORMATION			
Test Function Name :	testQuarterConstructor2_2	Test Case Number:	#3
Test Case Description:	Provide a brief description of what functionality the case will test.		
Results:	☐Pass ☐Fail		
TEST			
Input Specifications:	Define each input required to execute the test case, and reference any required relationships between inputs.		
Procedural Steps:	Describe the sequences of actions necessary to prepare and execute the test case. Provide detailed test procedures for each test case; explain precisely how each test case will be executed.		
Expected Results of Case:	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 :	Define all of the outputs and features required of the test case and provide expected values. While executing the test, record and describe the visually observable outputs as they occur. Produce tangible evidence of the output such as a screen print. At the conclusion, describe the actual outcome. Indicate whether the test passed or failed, and identify any discrepancies between the expected results and the actual results.		

/////

GENERAL INFORMATION			
Test Function Name :	testQuarterConstructor2_3	Test Case Number:	#4
Test Case Description:	Takes an illegal time 10/10/10000 as a parameter for the constructor and it throws an exception		
Results:	PASS		
TEST			
Input Specifications:	- Time object in milliseconds - Quarter object time as parameter		
Procedural Steps:	Take a time object 10/10/10000 in milliseconds as aparameter for the constructor and it is expected to throw an Illegal argument exception		
Expected Results of Case:	expected to throw an Illegal argument exception		
ACTUAL RESULTS			
Output Specifications and comments :	IlleagalArgumentException as the expected because 1899 is out of range		

GENERAL INFORMATION			
Test Function Name :	testQuarterConstructor3	Test Case Number:	#5
Test Case Description:	Takes an illegal Year 10000 as a parameter for the constructor and it throws an exception		
Results:	PASS		
TEST			
Input Specifications:	- Year object - Integer Quarter		
Procedural Steps:	Take a year object 10000 as a parameter for the constructor and it is expected to throw an Illegal argument exception		
Expected Results of Case:	expected to throw an Illegal argument exception		
ACTUAL RESULTS			
Output Specifications and comments :	IlleagalArgumentException as the expected because 10000 is out of range		

GENERAL INFORMATION			
Test Function Name :	testQuarterConstructor3_2	Test Case Number:	#6
Test Case Description:	Tests the constructor if it takes valid quarter and invalid year (1899) as an object from class year		
Results:	PASS		
TEST			
Input Specifications:	- Int Quarter = 4 - Year year = 1899		
Procedural Steps:	Takes a quarter = 4 and year = 1899 as parameters to the constructor and it is expected to throw an Illegal argument exception		
Expected Results of Case:			
ACTUAL RESULTS			
Output Specifications and comments :	IlleagalArgumentException as the expected because 1899 is out of range		

GENERAL INFORMATION			
Test Function Name :	testQuarterConstructor3_3	Test Case Number:	#7
Test Case Description:	Takes invalid quarter >4 and a valid year and expected to throw an exception		
Results:	FAIL		
TEST			
Input Specifications:	- Int Quarter = 5 - Year year = 1999		
Procedural Steps:	Create Quarter constructor with quarter = 5, and year = 1999 expected to throw an exception but instead of that it creates the constructor successfully and that is wrong		
Expected Results of Case:	Expected to throw an illegalArgumentException		
ACTUAL RESULTS			
Output Specifications and comments :	It creates the constructor successfully		

GENERAL INFORMATION			
Test Function Name :	testQuarterConstructor3_4	Test Case Number:	#8
Test Case Description:	Takes invalid quarter and a valid year and expected to throw an exception		
Results:	FAIL		
TEST			
Input Specifications:	- Int Quarter = 0 - Year year = 1999		
Procedural Steps:	Create Quarter constructor with quarter = 0, and year = 1999 expected to throw an exception but instead of that it creates the constructor successfully and that is wrong		
Expected Results of Case:	Expected to throw an illegalArgumentException		
ACTUAL RESULTS			
Output Specifications and comments :	It creates the constructor successfully		

GENERAL INFORMATION			
Test Function Name :	testQuarterConstructor3_5	Test Case Number:	#9
Test Case Description:	It tests the happy scenario as it takes quarter integer 3 which is <=4 & >=1 and a valid year 1990 which is >=1900 & <=9999		
Results:	PASS		
TEST			
Input Specifications:	- Int Quarter = 3 - Year year = 1990		
Procedural Steps:	Create Quarter constructor with quarter = 3, and year = 1990 expected to make a Quarter object with quarter = 3 and year = 1990		
Expected Results of Case:	make a Quarter object with quarter = 3 and year = 1990		
ACTUAL RESULTS			
Output Specifications and comments :	make a Quarter object with quarter = 3 and year = 1990 successfully		

GENERAL INFORMATION			
Test Function Name :	testQuarterConstructor4	Test Case Number:	#10
Test Case Description:	Tests the behavior of the constructor when it takes valid quarter and valid int year		
Results:	PASS		
TEST			
Input Specifications:	- Int Quarter = 4 - Int year = 1990		
Procedural Steps:	Creates a constructor with quarter = 4 and year = 1990 and check it is created with these values or not		
Expected Results of Case:	Create constructor successfully		
ACTUAL RESULTS			
Output Specifications and comments :	Create constructor successfully		

GENERAL INFORMATION			
Test Function Name :	estQuarterConstructor4_2	Test Case Number:	#11
Test Case Description:	Tests the behavior of the constructor when it takes invalid year that is smaller than 1900 and valid int quarter		
Results:	PASS		
TEST			
Input Specifications:	- Int Quarter = 4 - Int year = 1899		
Procedural Steps:	Creates a constructor with quarter = 4 and year = 1899 and check if it throws an invalid argument exception or not		
Expected Results of Case:	Throw an IllegalArgumentException		
ACTUAL RESULTS			
Output Specifications and comments :	IlleagalArgumentException as the expected because 1899 is out of range		

GENERAL INFORMATION			
Test Function Name :	testQuarterConstructor4_3	Test Case Number:	#12
Test Case Description:	Tests the behavior of the constructor when it takes invalid quarter that is greater than 4 and valid int year		
Results:	FAIL		
TEST			
Input Specifications:	- Int Quarter = 5 - Int year = 1900		
Procedural Steps:	Creates a constructor with quarter = 5 and year = 1990 and check if it throws an invalid argument exception or not		
Expected Results of Case:	Throw an IllegalArgumentException		
ACTUAL RESULTS			
Output Specifications and comments :	It creates the constructor successfully		

GENERAL INFORMATION			
Test Function Name :	testQuarterConstructor4_4	Test Case Number:	#13
Test Case Description:	Tests the behavior of the constructor when it takes invalid quarter that is smaller than 1 and valid int year		
Results:	FAIL		
TEST			
Input Specifications:	- Int Quarter = 0 - Int year = 1900		
Procedural Steps:	Creates a constructor with quarter = 0 and year = 1990 and check if it throws an invalid argument exception or not		
Expected Results of Case:	Throw an IllegalArgumentException		
ACTUAL RESULTS			
Output Specifications and comments :	It creates the constructor successfully		

GENERAL INFORMATION			
Test Function Name :	testQuarterConstructor4_5	Test Case Number:	#14
Test Case Description:	Tests the behavior of the constructor when it takes valid quarter and invalid int year which is greater than 9999 and expected to throw an exception		
Results:	PASS		
TEST			
Input Specifications:	- Int Quarter = 4 - Int year = 10000		
Procedural Steps:	Creates a constructor with quarter = 4 and year = 10000 and check if it throws an invalid argument exception or not		
Expected Results of Case:	Throw an IllegalArgumentException		
ACTUAL RESULTS			
Output Specifications and comments :	Throw an IllegalArgumentException		

GENERAL INFORMATION			
Test Function Name :	testQuarterConstructor5()	Test Case Number:	#15
Test Case Description:	Tests the happy scenario for the constructor that takes (Date , TimeZone) As the Date is >=1900 & <=9999		
Results:	PASS		
TEST			
Input Specifications:	- Date (1/MAY/2023) - Timezone with ID = UTC		
Procedural Steps:	Creates date object (1/MAY/2023) and time zone with ID = UTC and pass these parameters to the constructor it is expected to create Quarter object successfully		
Expected Results of Case:	create Quarter object successfully with quarter = 2 and year = 2023		
ACTUAL RESULTS			
Output Specifications and comments :	create Quarter object successfully with quarter = 2 and year = 2023		

GENERAL INFORMATION			
Test Function Name :	testQuarterConstructor5_2	Test Case Number:	#16
Test Case Description:	Expected to throw an exception when try to pass invalid time and valid timezone to the constructor		
Results:	PASS		
TEST			
Input Specifications:	- Date (1/MAY/1899) - Timezone with ID = UTC		
Procedural Steps:	Creates date object ((1/MAY/1899) and time zone with ID = UTC and pass these parameters to the constructor it is expected to throw an exception		
Expected Results of Case:	Throw an IllegalArgumentException		
ACTUAL RESULTS			
Output Specifications and comments :	Throw an IllegalArgumentException		

GENERAL INFORMATION			
Test Function Name :	testQuarterConstructor5_3	Test Case Number:	#17
Test Case Description:	Tests the boundary year for the constructor that takes (Date , TimeZone) As the Date is >9999		
Results:	PASS		
TEST			
Input Specifications:	- Date (1/MAY/10000) - Timezone with ID = UTC		
Procedural Steps:	Creates date object ((1/MAY/10000) and time zone with ID = UTC and pass these parameters to the constructor it is expected to throw an exception		
Expected Results of Case:	Throw an IllegalArgumentException		
ACTUAL RESULTS			
Output Specifications and comments :	Throw an IllegalArgumentException		

GENERAL INFORMATION			
Test Function Name :	testToString	Test Case Number:	#18
Test Case Description:	Test the valid format of toString function		
Results:	PASS		
TEST			
Input Specifications:	Quarter = 4 Year = 2000		
Procedural Steps:	Creates the constructor with quarter = 4 and year = 2000 and check if to string function equals (“Q4/2000”)		
Expected Results of Case:	“Q4/2000”		
ACTUAL RESULTS			
Output Specifications and comments :	“Q4/2000”		

GENERAL INFORMATION			
Test Function Name :		Test Case Number:	#19
Test Case Description:	Provide a brief description of what functionality the case will test.		
Results:	☐Pass ☐Fail		
TEST			
Input Specifications:	Define each input required to execute the test case, and reference any required relationships between inputs.		
Procedural Steps:	Describe the sequences of actions necessary to prepare and execute the test case. Provide detailed test procedures for each test case; explain precisely how each test case will be executed.		
Expected Results of Case:	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 :	Define all of the outputs and features required of the test case and provide expected values. While executing the test, record and describe the visually observable outputs as they occur. Produce tangible evidence of the output such as a screen print. At the conclusion, describe the actual outcome. Indicate whether the test passed or failed, and identify any discrepancies between the expected results and the actual results.		

GENERAL INFORMATION			
Test Function Name :	testParseQuarterBackSlash	Test Case Number:	#20
Test Case Description:	Tests parseQuarter function when it takes the format “YYYY\\QN”		
Results:	PASS		
TEST			
Input Specifications:	- Parameter to parseQuarter 2004\\Q4		
Procedural Steps:	Pass 2004\\Q4 to parseQuarter function and check if it throws an exception		
Expected Results of Case:	Throws TimePeriodFormatException		
ACTUAL RESULTS			
Output Specifications and comments :	Throws TimePeriodFormatException		

GENERAL INFORMATION			
Test Function Name :		Test Case Number:	#21
Test Case Description:	Provide a brief description of what functionality the case will test.		
Results:	☐Pass ☐Fail		
TEST			
Input Specifications:	Define each input required to execute the test case, and reference any required relationships between inputs.		
Procedural Steps:	Describe the sequences of actions necessary to prepare and execute the test case. Provide detailed test procedures for each test case; explain precisely how each test case will be executed.		
Expected Results of Case:	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 :	Define all of the outputs and features required of the test case and provide expected values. While executing the test, record and describe the visually observable outputs as they occur. Produce tangible evidence of the output such as a screen print. At the conclusion, describe the actual outcome. Indicate whether the test passed or failed, and identify any discrepancies between the expected results and the actual results.		

GENERAL INFORMATION			
Test Function Name :	testParseQuarterDoubleDash	Test Case Number:	#22
Test Case Description:	Tests parseQuarter function when it takes the format “YYYY- - QN”		
Results:	FAIL		
TEST			
Input Specifications:	Parameter to parseQuarter "2004--Q4"		
Procedural Steps:	Pass 2004- -Q4 to parseQuarter function and check if it throws an exception		
Expected Results of Case:	Throws TimePeriodFormatException		
ACTUAL RESULTS			
Output Specifications and comments :	It accepts the format Comment : poor instructions if the function accepts more than one dash or not		

GENERAL INFORMATION			
Test Function Name :		Test Case Number:	#23
Test Case Description:	Provide a brief description of what functionality the case will test.		
Results:	☐Pass ☐Fail		
TEST			
Input Specifications:	Define each input required to execute the test case, and reference any required relationships between inputs.		
Procedural Steps:	Describe the sequences of actions necessary to prepare and execute the test case. Provide detailed test procedures for each test case; explain precisely how each test case will be executed.		
Expected Results of Case:	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 :	Define all of the outputs and features required of the test case and provide expected values. While executing the test, record and describe the visually observable outputs as they occur. Produce tangible evidence of the output such as a screen print. At the conclusion, describe the actual outcome. Indicate whether the test passed or failed, and identify any discrepancies between the expected results and the actual results.		

GENERAL INFORMATION			
Test Function Name :	testParseQuarterDoubleSlash	Test Case Number:	#24
Test Case Description:	Tests parseQuarter function when it takes the format “ QN//YYYY”		
Results:	FAIL		
TEST			
Input Specifications:	Parameter to parseQuarter "Q4//2004"		
Procedural Steps:	Pass "Q4//2004" to parseQuarter function and check if it throws an exception		
Expected Results of Case:	Throws TimePeriodFormatException		
ACTUAL RESULTS			
Output Specifications and comments :	It accepts the format Comment : poor instructions if the function accepts more than one dash or not		

GENERAL INFORMATION			
Test Function Name :		Test Case Number:	#25
Test Case Description:	Provide a brief description of what functionality the case will test.		
Results:	☐Pass ☐Fail		
TEST			
Input Specifications:	Define each input required to execute the test case, and reference any required relationships between inputs.		
Procedural Steps:	Describe the sequences of actions necessary to prepare and execute the test case. Provide detailed test procedures for each test case; explain precisely how each test case will be executed.		
Expected Results of Case:	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 :	Define all of the outputs and features required of the test case and provide expected values. While executing the test, record and describe the visually observable outputs as they occur. Produce tangible evidence of the output such as a screen print. At the conclusion, describe the actual outcome. Indicate whether the test passed or failed, and identify any discrepancies between the expected results and the actual results.		

GENERAL INFORMATION			
Test Function Name :	testParseQuarterDoubleSlash	Test Case Number:	#26
Test Case Description:	Tests parseQuarter function when it takes the format “ QN YYYY”		
Results:	FAIL		
TEST			
Input Specifications:	Parameter to parseQuarter "Q4 2004"		
Procedural Steps:	Pass "Q4 2004" to parseQuarter function and check if it throws an exception		
Expected Results of Case:	Throws TimePeriodFormatException		
ACTUAL RESULTS			
Output Specifications and comments :	It accepts the format Comment : poor instructions if the function accepts more than one dash or not		

GENERAL INFORMATION			
Test Function Name :		Test Case Number:	#27
Test Case Description:	Provide a brief description of what functionality the case will test.		
Results:	☐Pass ☐Fail		
TEST			
Input Specifications:	Define each input required to execute the test case, and reference any required relationships between inputs.		
Procedural Steps:	Describe the sequences of actions necessary to prepare and execute the test case. Provide detailed test procedures for each test case; explain precisely how each test case will be executed.		
Expected Results of Case:	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 :	Define all of the outputs and features required of the test case and provide expected values. While executing the test, record and describe the visually observable outputs as they occur. Produce tangible evidence of the output such as a screen print. At the conclusion, describe the actual outcome. Indicate whether the test passed or failed, and identify any discrepancies between the expected results and the actual results.		

GENERAL INFORMATION			
Test Function Name :	testParseQuarterDoubleSpace	Test Case Number:	#28
Test Case Description:	Tests parseQuarter function when it takes the format “ QN,,YYYY”		
Results:	FAIL		
TEST			
Input Specifications:	Parameter to parseQuarter “ QN,,YYYY”		
Procedural Steps:	Pass “QN,,YYYY” to parseQuarter function and check if it throws an exception		
Expected Results of Case:	Throws TimePeriodFormatException		
ACTUAL RESULTS			
Output Specifications and comments :	It accepts the format Comment : poor instructions if the function accepts more than one dash or not		

GENERAL INFORMATION			
Test Function Name :		Test Case Number:	#29
Test Case Description:	Provide a brief description of what functionality the case will test.		
Results:	☐Pass ☐Fail		
TEST			
Input Specifications:	Define each input required to execute the test case, and reference any required relationships between inputs.		
Procedural Steps:	Describe the sequences of actions necessary to prepare and execute the test case. Provide detailed test procedures for each test case; explain precisely how each test case will be executed.		
Expected Results of Case:	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 :	Define all of the outputs and features required of the test case and provide expected values. While executing the test, record and describe the visually observable outputs as they occur. Produce tangible evidence of the output such as a screen print. At the conclusion, describe the actual outcome. Indicate whether the test passed or failed, and identify any discrepancies between the expected results and the actual results.		

GENERAL INFORMATION			
Test Function Name :	testHashCode	Test Case Number:	#30
Test Case Description:	Tests hashCode function if the two Quarter instances with the same quarter and the year, have the same hashCode or not		
Results:	PASS		
TEST			
Input Specifications:	Two Quarter instances with the quarter = 1 and year = 2023		
Procedural Steps:	Creates two Quarter instances and test quality of hashcode for each of them		
Expected Results of Case:	Same hashCode value		
ACTUAL RESULTS			
Output Specifications and comments :	Same hashCode value		

GENERAL INFORMATION			
Test Function Name :	testHashCode2	Test Case Number:	#31
Test Case Description:	Tests hashCode function if the two Quarter instances have different quarter value and have same year, don't have the same hashCode		
Results:	PASS		
TEST			
Input Specifications:	Quarter_1 (2,2023) Quarter_2 (1,2023)		
Procedural Steps:	Creates two Quarter instances with different values and test if they have not same hashCode		
Expected Results of Case:	Don't have the same hashCode value		
ACTUAL RESULTS			
Output Specifications and comments :	Don't have the same hashCode value		

GENERAL INFORMATION			
Test Function Name :	testHashCode3	Test Case Number:	#32
Test Case Description:	Tests hashCode function if the two Quarter instances have different quarter value and have different year, don't have the same hashCode		
Results:	PASS		
TEST			
Input Specifications:	Quarter_1 (2,2021) Quarter_2 (1,2023)		
Procedural Steps:	Creates two Quarter instances with different values and test if they have not same hashCode		
Expected Results of Case:	Don't have the same hashCode value		
ACTUAL RESULTS			
Output Specifications and comments :	Don't have the same hashCode value		

GENERAL INFORMATION			
Test Function Name :	testGetFirstMillisecond	Test Case Number:	#33
Test Case Description:	Tests the first millisecond in the first quarter of the year		
Results:	PASS		
TEST			
Input Specifications:	Quarter (1, 2022) Calender (01/01/2022 00:00:00:0000)		
Procedural Steps:	Set calender with (01/01/2022 00:00:00) and creates Quarter object with quarter 1 and year = 2022 and check if the milliseconds in the calender is the same as first millisecond in the quarter		
Expected Results of Case:	Same milliseconds		
ACTUAL RESULTS			
Output Specifications and comments :	Same milliseconds		

GENERAL INFORMATION			
Test Function Name :	testGetFirstMillisecond2	Test Case Number:	#34
Test Case Description:	Tests the first millisecond in the last quarter of the year		
Results:	PASS		
TEST			
Input Specifications:	Quarter (4, 2022) Calender (01/10/2022 00:00:00:0000)		
Procedural Steps:	Set calender with (01/10/2022 00:00:00) and creates Quarter object with quarter 4 and year = 2022 and check if the milliseconds in the calender is the same as first millisecond in the quarter		
Expected Results of Case:	Same milliseconds		
ACTUAL RESULTS			
Output Specifications and comments :	Same milliseconds		

GENERAL INFORMATION			
Test Function Name :	testGetLastMillisecond	Test Case Number:	#35
Test Case Description:	Tests the last millisecond in the first quarter of the year		
Results:	PASS		
TEST			
Input Specifications:	Quarter (1, 2022) Calender (30/3/2022 23:59:59:9999)		
Procedural Steps:	Set calender with (30/3/2022 23:59:59:9999) and creates Quarter object with quarter 1 and year = 2022 and check if the milliseconds in the calender is the same as last millisecond in the quarter		
Expected Results of Case:	Same milliseconds		
ACTUAL RESULTS			
Output Specifications and comments :	Same milliseconds		

GENERAL INFORMATION			
Test Function Name :	testGetFirstMillisecond2	Test Case Number:	#36
Test Case Description:	Tests the last millisecond in the last quarter of the year		
Results:	PASS		
TEST			
Input Specifications:	Quarter (4, 2022) Calender (30/12/2022 23:59:59:9999)		
Procedural Steps:	Set calender with (30/12/2022 23:59:59:9999) and creates Quarter object with quarter 4 and year = 2022 and check if the milliseconds in the calender is the same as last millisecond in the quarter		
Expected Results of Case:	Same milliseconds		
ACTUAL RESULTS			
Output Specifications and comments :	Same milliseconds		

GENERAL INFORMATION			
Test Function Name :	testGetQuarter	Test Case Number:	#37
Test Case Description:	Tests if the function returns the correct quarter of the object		
Results:	PASS		
TEST			
Input Specifications:	Quarter (4,2500)		
Procedural Steps:	Creates quarter object with quarter = 4 and check if getQuarter returns 4 or not		
Expected Results of Case:	4		
ACTUAL RESULTS			
Output Specifications and comments :	4		

GENERAL INFORMATION			
Test Function Name :	testGetYear	Test Case Number:	#38
Test Case Description:	Tests if the function returns the correct year of the object		
Results:	PASS		
TEST			
Input Specifications:	Quarter (2,2500)		
Procedural Steps:	Creates quarter object with year = 2500 and check if GetYear returns 2500 or not		
Expected Results of Case:	2500		
ACTUAL RESULTS			
Output Specifications and comments :	2500		

GENERAL INFORMATION			
Test Function Name :	testPrevious	Test Case Number:	#39
Test Case Description:	Tests if the function previous() returns the previous Quarter when the quarter have the normal value = 2 and normal year = 2023		
Results:	PASS		
TEST			
Input Specifications:	Quarter (2,2023)		
Procedural Steps:	Creates quarter with quarter value = 2 and year =2023, and check if the previous function returns (1,2023) or not		
Expected Results of Case:	Q2/2023		
ACTUAL RESULTS			
Output Specifications and comments :	Q2/2023		

GENERAL INFORMATION			
Test Function Name :	testPrevious2	Test Case Number:	#40
Test Case Description:	Tests if the function previous() returns the previous Quarter when the quarter have the minimum quarter value which is 1 and minimum year = 1900		
Results:	PASS		
TEST			
Input Specifications:	Quarter (1,1900)		
Procedural Steps:	Creates quarter with quarter value = 1 and year =1900, and check if the previous function returns null or not		
Expected Results of Case:	NULL		
ACTUAL RESULTS			
Output Specifications and comments :	NULL		

GENERAL INFORMATION			
Test Function Name :	testPrevious3	Test Case Number:	#41
Test Case Description:	Tests if the function previous() returns the previous Quarter when the quarter have the minimum quarter value which is 1 and normal year = 1901		
Results:	PASS		
TEST			
Input Specifications:	Quarter (1,1901)		
Procedural Steps:	Creates quarter with quarter value = 1 and year =1901, and check if the previous function returns (4, 1900) or not		
Expected Results of Case:	Q4/1901		
ACTUAL RESULTS			
Output Specifications and comments :	Q4/1901		

GENERAL INFORMATION			
Test Function Name :	testPrevious4	Test Case Number:	#42
Test Case Description:	Tests if the function previous() returns the previous Quarter when the quarter have the maximum quarter value which is 4		
Results:	PASS		
TEST			
Input Specifications:	Quarter (4,1990)		
Procedural Steps:	Creates quarter with quarter value = 1 and year =1900, and check if the previous function returns null or not		
Expected Results of Case:	Expected result for the quarter is 3		
ACTUAL RESULTS			
Output Specifications and comments :	The actual result for the quarter is 3		

GENERAL INFORMATION			
Test Function Name :	testNext	Test Case Number:	#43
Test Case Description:	Tests the behavior of the function next() if the maximum quarter value = 4 and the year is maximum value = 9999		
Results:	PASS		
TEST			
Input Specifications:	Quarte (4,9999)		
Procedural Steps:	Creates quarter with quarter value = 4 and year =9999, and check if the next function returns null or not		
Expected Results of Case:	NULL		
ACTUAL RESULTS			
Output Specifications and comments :	NULL		

GENERAL INFORMATION			
Test Function Name :	testNext2	Test Case Number:	#44
Test Case Description:	Tests the behavior of the function next() if the quarter value = 3 and the year is maximum value = 9999		
Results:	PASS		
TEST			
Input Specifications:	Quarte (3,9999)		
Procedural Steps:	Creates quarter with quarter value = 3 and year =9999, and check if the next function returns (4,9999)		
Expected Results of Case:	Q4/9999		
ACTUAL RESULTS			
Output Specifications and comments :	Q4/9999		

GENERAL INFORMATION			
Test Function Name :	testGetSerialIndex	Test Case Number:	#45
Test Case Description:	Check if serialIndex calculate correctly or not		
Results:	PASS		
TEST			
Input Specifications:	Quarter (1,1990)		
Procedural Steps:	Creates quarter (1,1990) and check if getSerialIndex returns 8093		
Expected Results of Case:	8093		
ACTUAL RESULTS			
Output Specifications and comments :	8093 Comment: Poor documentation as it takes me too long to discover the calculations of the function		

GENERAL INFORMATION			
Test Function Name :	testEquals	Test Case Number:	#46
Test Case Description:	Tests equals() function when the two quarters instances have the same quarter and the same year		
Results:	PASS		
TEST			
Input Specifications:	quarter _1 (2,2023) quarter _2 (2,2023)		
Procedural Steps:	Creates two quarter instances with the same quarter and year and check if equals function returns true or not		
Expected Results of Case:	TRUE		
ACTUAL RESULTS			
Output Specifications and comments :	TRUE		

GENERAL INFORMATION			
Test Function Name :	testEquals2	Test Case Number:	#47
Test Case Description:	Tests equals() function when the two quarters instances have the different quarter and the different year		
Results:	PASS		
TEST			
Input Specifications:	quarter _1 (3,1990) quarter _2 (2,2023)		
Procedural Steps:	Creates two quarter instances with the different quarter and year and check if equals function returns false or not		
Expected Results of Case:	FALSE		
ACTUAL RESULTS			
Output Specifications and comments :	FALSE		

GENERAL INFORMATION			
Test Function Name :	testCompareTo	Test Case Number:	#48
Test Case Description:	Tests compareTo() function when compare the quarter object to other quarter objects		
Results:	PASS		
TEST			
Input Specifications:	quarter (2,2023) Quarter_1 (3,2023) Quarter_2 (1,2023) Quarter_3 (2,2023)		
Procedural Steps:	Crates the four quarter instances and check compareTo function when compare (quarter,Quarter_1) (quarter,Quarter_2) (quarter,Quarter_3)		
Expected Results of Case:	Expected less than zero In case (quarter,Quarter_1) Expected grater than zero In case (quarter,Quarter_2) Expected zero In case (quarter,Quarter_3)		
ACTUAL RESULTS			
Output Specifications and comments :	less than zero In case (quarter,Quarter_1) grater than zero In case (quarter,Quarter_2) zero In case (quarter,Quarter_3)		