Test Report

The Method to be tested	Test cases			Status
Testing Constructors:	**Input**	**Expected**	**Actual**	
Quarter()	___	Current quarter = 2 Current year = 2023	quarter = 2 year = 2023	**Passed**
Quarter(int quarter, int year)	quarter = 2 year = 2023	quarter = 2 year = 2023	quarter = 2 year = 2023	**Passed**
	quarter = 0 year = 1900 => Quarter value < 1	Throwing an Exception	quarter = 0 year = 1900	**Failed**
	quarter = 5 year = 1900 => Quarter value > 4	Throwing an Exception	quarter = 5 year = 1900	**Failed**
	quarter = 1 year = 1800 => Year value < 1900	Throwing an Exception	Throwing an Exception	**Passed**
	quarter = 1 year = 10000 => Year value > 9999	Throwing an Exception	Throwing an Exception	**Passed**
Quarter(java.util.Date time)	date(1680300000000L) //Apr 1, 2023	quarter = 2 year = 2023	quarter = 2 year = 2023	**Passed**

Quarter(java.util.Date time, java.util.TimeZone zone)	date(1680300000000L)
	//Apr 1, 2023
	timezone("EET")
		quarter = 2
		year = 2023
			quarter = 2
			year = 2023
				Passed

Quarter(int quarter, Year year)	quart = 2 year(2023)	quarter = 2 year = 2023	quarter = 2 year = 2023	**Passed**
	quarter = 0 year(1900) => Quarter value < 1	Throwing an Exception	quarter = 0 year = 1900	**Failed**
	quarter = 5 year(1900) => Quarter value > 4	Throwing an Exception	quarter = 5 year = 1900	**Failed**
	quarter = 1 year(1800) => Year value < 1900	Throwing an Exception	Throwing an Exception	**Passed**
	quarter = 1 year(10000) => Year value > 9999	Throwing an Exception	Throwing an Exception	**Passed**

The Method to be tested	Test cases			Status
Testing Methods:	**Input**	**Expected**	**Actual**	
compareTo(java.lang.Object o1)	quarter1(2, 2023) quarter2(3,2023)	Result < 0	Result < 0	**Passed**
	quarter1(3, 2022) quarter2(2, 2023)	Result < 0	Result < 0	**Passed**
	quarter1(2, 2022) quarter2(2, 2022)	Result == 0	Result == 0	**Passed**
	quarter1(2, 2023) quarter2(1, 2023)	Result > 0	Result > 0	**Passed**
	quarter1(2, 2023) quarter2(1, 2022)	Result > 0	Result > 0	**Passed**
equals(java.lang.Object obj)	quarter1(2, 2023) quarter2(2, 2023)	Result = true	Result = true	**Passed**
	quarter1(2, 2023) quarter2(3, 2023)	Result = false	Result = false	**Passed**
	quarter1(2,2023) quarter2(2, 2022)	Result = false	Result = false	**Passed**
	quarter1(2,2023) quarter2(3, 2022)	Result = false	Result = false	**Passed**
getFirstMillisecond(java.util.Calendar calendar)	Calendar = April 1, 2023 00:00:00	1680300000000L	1680300000000L	**Passed**
getLastMillisecond(java.util.Calendar calendar)	Calendar = March 31, 2023 23:59:59	1680299999999L	1680299999999L	**Passed**
getQuarter()	quarter(1, 2023)	Quarter = 1	Quarter = 1	**Passed**
getSerialIndex()	object1 (2, 2023) object2 (3, 2023)	object1.serialIndex < object2.serialIndex = True	object1.serialIndex < object2.serialIndex = True	**Passed**
getYear()	quarter(1, 2023)	Year = 2023	Year = 2023	**Passed**
hashCode()	object1 (1, 2023) object2 (1, 2023) object3 (2, 2023)	object1.hashcode == object2.hashcode	object1.hashcode == object2.hashcode	**Passed**
next()	***=>Next quarter in the same year*** quarter(2, 2023)	Quarter(3, 2023)	Quarter(3, 2023)	**Passed**

	=>Next quarter in the next year quarter(4, 2022)	Quarter(1, 2023)	Quarter(1, 2023)	**Passed**
	=>Next quarter is null quarter(4, 9999)	NULL	NULL	**Passed**
parseQuarter(java.lang.String s)	Input string format: • "YYYY-QN" with dash = "2022-Q2" • "QN-YYYY" with dash = "Q2-2022" • "YYYY-QN" with space = "2022 Q2" • "QN-YYYY" with space = "Q2 2022" • "YYYY-QN" with comma = "2022,Q2" • "QN,YYYY" with comma = "Q2,2022" • "YYYY-QN" with forward-slash = "2022/Q2" • "QN-YYYY" with forward-slash = "Q2/2022"	• quarter = 2, year = 2022 • quarter = 2, year = 2022 • quarter = 2, year = 2022 • quarter = 2, year = 2022 • quarter = 2, year = 2022 • quarter = 2, year = 2022 • quarter = 2, year = 2022 • quarter = 2, year = 2022	• quarter = 2, year = 2022 • quarter = 2, year = 2022 • quarter = 2, year = 2022 • quarter = 2, year = 2022 • quarter = 2, year = 2022 • quarter = 2, year = 2022 • quarter = 2, year = 2022 • quarter = 2, year = 2022	**Passed**
	Input string (missing quarter) = "2022"	Throwing an Exception	Throwing an Exception	**Passed**
	Input string (missing year) = "Q2"	Throwing an Exception	Throwing an Exception	**Passed**
	Input string (invalid quarter) = "Q5-2022"	Throwing an Exception	quarter = 5 year = 2022	**Failed**
	Invalid input string (invalid year) = "ABC"	Throwing an Exception	Throwing an Exception	**Passed**
	Invalid input string (empty string) = ""	Throwing an Exception	Throwing an Exception	**Passed**
previous()	***=>Previous quarter in the same year*** quarter(2, 2023);	Quarter(1, 2023)	Quarter(1, 2023)	**Passed**
	=>Previous quarter in the previous Year quarter(1, 2023);	Quarter(4, 2022)	Quarter(4, 2022)	**Passed**
	=>Previous quarter is null quarter(1, 1900);	NULL	NULL	**Passed**
toString()	quarter(2, 2023)	"Q2/2023"	"Q2/2023"	**Passed**