Unit Name		Sqrt		
Summary (e.g. Interface spec)		Return the square root of the given integer		
Test Case Number	Input Type(s)	Input Value(s)	Expected Result	Note
1	Integer	Integer.MAX_VALUE	46340.95	boundary
2	Integer	0	0	boundary
3	Integer	-1	Exception	Error case
4	Integer	1	1	
5	Integer	2	1.414	
6	Integer	100	10	some small int
7	Integer	16584033	4072.34	Some large int

Explanation for Sqrt:

Test cases 1, 2 and 3 covers the boundary, since we know that only 0 and positive integers have square root. So 0 and Integer.MAX_VALUE is the boundary, that if input is 0, its sqrt root will itself, and if it is max integer, its square root could be get as shown in the form, if the input is negative, it will throw an exception. Then test case 4 and 5 will check 1 and 2, test cases 6 and 7 will test some small integer and large integer.

Unit Name		Sqr		
Summary (e.g. Interface spec)		Return the square of the given integer		
Test Case Number	Input Type(s)	Input Value(s)	Expected Result	Note
1	Integer	Floor Sqrt of Integer.MAX_VALUE	Integer.MAX_VALUE	boundary
2	Integer	Floor Sqrt of Integer.MAX_VALUE + 1	Exception	Error case
3	Integer	- Floor Sqrt of Integer.MAX_VALUE	Integer.MAX_VALUE	boundary
4	Integer	- Floor Sqrt of Integer.MIN_VALUE - 1	Exception	Error case
5	Integer	-1	1	
6	Integer	0	0	
7	Integer	1	1	
8	Integer	2	4	
9	Integer	-14560	211993600	Some small int
10	Integer	14560	211993600	Some large int

Explanation for Sqr:

Test cases 1, 2, 3 and 4 covers the boundary, we know that both negative integers, 0 and positive integers have square, and the largest and smallest input should make sure that its square will smaller than or equal to Integer.MAX_VALUE, So we upper boundary will be the floor of square root of Integer.MAX_VALUE, and the lower boundary will be the negative value of the upper boundary. And if the input is greater than upper boundary or smaller than lower boundary will throws exception. Then test cases 5, 6, 7, 8 will test -1, 0, 1, 2 to check if the method works well for them. Test cases 9 and 10 will test some small integer and large integer.

Unit Name		Factorial		
Summary (e.g. Interface spec)		Return the factorial of the given integer		
Test Case Number	Input Type(s)	Input Value(s)	Expected Result	Note
1	Integer	12	479001600	boundary
2	Integer	13	Exception	Error case
3	Integer	0	1	boundary
4	Integer	-1	Exception	Error case
5	Integer	1	1	
6	Integer	2	2	
7	Integer	5	120	Some small int
8	Integer	10	3628800	Some large int

Explanation for Factorial:

Test cases 1, 2, 3 and 4 covers the boundary, we know that only 0 and positive integers have factorial, and the largest and smallest input should make sure that its factorial will smaller than or equal to Integer.MAX_VALUE, So we upper boundary will be 12, and the factorial of 13 will exceed the Integer.MAX_VALUE and it will throw exception. And the lower boundary will be 0 and its factorial should be 1. If input is negative integers will cause error, method will throw exception. Then test cases 5, 6 will test 1, 2 to check if the method works well for them. Test cases 7 and 8 will test some small integer and large integer.

Unit Name		Sum up		
Summary (e.g. Interface spec)		Return the sum from 0 to the given integer		
Test Case Number	Input Type(s)	Input Value(s)	Expected Result	Note
1	Integer	65536	2147516417	boundary
2	Integer	65537	Exception	Error case

3	Integer	0	0	boundary
4	Integer	-1	Exception	Error case
5	Integer	1	1	
6	Integer	2	3	
7	Integer	10	56	Some small int
8	Integer	50000	1250025001	Some large int

Explanation for SumUp

Test cases 1, 2, 3 and 4 covers the boundary, we know that only 0 and positive integers can sum up, and the largest and smallest input should make sure that the sum from 0 to the integer will smaller than or equal to Integer.MAX_VALUE, So we upper boundary will be 65536, and the factorial of 65537 will exceed the Integer.MAX_VALUE and it will throw exception. And the lower boundary will be 0 and its factorial should be 0. If input is negative integers will cause error, it will throw exception. Then test cases 5, 6 will test 1, 2 to check if the method works well for them. Test cases 7 and 8 will test some small integer and large integer.

Unit Name		X plus Y		
Summary (e.g. Interface spec)		Return the sum of given X and Y		
Test Case Number	Input Type(s)	Input Value(s)	Expected Result	Note
1	Integer	X = Integer.MAX_VALUE, Y = 0	Integer.MAX_VALUE	boundary
2	Integer	X = Integer.MAX_VALUE, Y = 1	Exception	Error case
3	Integer	X = 0, Y = Integer.MAX_VALUE	Integer.MAX_VALUE	boundary
4	Integer	X = 1, Y = Integer.MAX_VALUE	Exception	Error case
5	Integer	X = Integer.MIN_VALUE, Y = 0	Integer.MIN_VALUE	boundary
6	Integer	X = Integer.MIN_VALUE, Y = -1	Exception	Error case
7	Integer	X = 0, Y = Integer.MIN_VALUE	Integer.MIN_VALUE	boundary
8	Integer	X = -1, Y = Integer.MIN_VALUE	Exception	Error case
9	Integer	X = 666666666, Y = 1666666666	Exception	Error case
10	Integer	X = -666666666, Y = -1666666666	Exception	Error case
11	Integer	X = 0, Y = 0	0	
12	Integer	X = 0, Y = 1	1	
13	Integer	X = 0, Y = -2	-2	

14	Integer	X = 1, Y = -1	0	
15	Integer	X = -1, Y = 1	0	
16	Integer	X = 1, Y = 1	2	
17	Integer	X = -1, Y = -1	-2	
18	Integer	X = -1346575100, Y = 510	1346575590	
19	Integer	X =510, Y = 1346575100	1346575610	

Explanation for X Plus Y:

Test cases 1 to 10 covers the boundary, we know that the sum of the two integer should be in the range of Integer.MIM_VALUE and Integer.MAX_VALUE. These test cases cover that one of X and Y is zero and the other is Integer.MAX_VALUE or Integer.MIN_VALUE, and both X and Y are positive and both X and Y are negative. In a word, either X or Y can not be too large or too small when the other one is set, or it will throw exception. Then test cases 11 to 17 covers the 0, 1 and -1 cases, and also covers the exchange of the values of X and Y, check if the method works well for all these cases. Finally 18 and 19 test some large positive integer and smaller negative integers.

Unit Name		Despacer		
Summary (e.g. Interface spec)		Replace multiple continuous spaces with only one space in a given string		
Test Case Number	Input Type(s)	Input Value(s)	Expected Result	Note
1	String	“”	“”	boundary
2	String	“a”	“a”	boundary
3	String	“ ”	“ ”	boundary
4	String	“a ”	“a ”	boundary
5	String	“ a”	“ a”	boundary
6	String	“a big apple”	“a big apple”	boundary
7	String	“a ”	“a ”	
8	String	“ a”	“ a“	
9	String	“an apple”	“an apple”	
10	String	“it’s really a big apple”	“it’s really a big apple”	

Explanation for Despacer:

Test cases 1 to 6 covers the boundary, we know that an empty string or a pure string without appearance of any space is one of the boundary case, it should the input string itself. And a string only contains no continuous space is also a boundary, there are two kind of cases, one is that the string only contains space, the other is the string contains both space and other characters, it should return the input string itself. While for string that contains continuous space, it should keep one space instead for each continuous spaces, test cases 7 to 10 cover this case.