No.	Clase	Metodo	Escenario	Valores Entrada	Resultado
1	StackTest	testStack	Se crea una Pila	Null	Se revisa que la pila creada no sea NULL
2	StackTest	testPush	Se crea una pila se agrega un elemento	“hola”	Se comprueba que la pila no esté vacía
3	StackTest	testPush	Se crea una pila se agrega un elemento	“hola”	Se comprueba que el método count sea igual a 1
4	StackTest	testPush	Se crea una pila se agrega un elemento	“hola”	Se utiliza el método peek y se comprueba que no sea NULL el resultado
5	StackTest	testPeek	Se crea una pila se agrega un elemento se utiliza el método peek	“hola”	Se comprueba que la pila no esté vacía además de que el elemento recuperado se igual al elemento que se agrego
6	StackTest	testPeek	Se crea una pila se agrega un elemento se utiliza el método peek	“hola”	Se comprueba que la pila no esté vacía además de que el elemento recuperado no sea NULL
7	StackTest	testPeek	Se crea una pila se agrega un elemento se utiliza el método peek	“hola”	Se comprueba que la pila no esté vacía además de que el tamaño de la pila después de utilizar el método peek sea el mismo uno
8	StackTest	testPop	Se crea una pila se agrega un elemento se utiliza el método pop	“hola”	Se comprueba que la pila esté vacía además de que el elemento recuperado se igual al elemento que se agrego
9	StackTest	testPop	Se crea una pila se agrega un elemento se utiliza el método pop	“hola”	Se comprueba que la pila esté vacía además de que el elemento recuperado por segunda vez sea NULL
10	StackTest	testPop	Se crea una pila se agrega un elemento se utiliza el método pop	“hola”	Se comprueba que la pila no esté vacía después de utilizar el método pop
11	StackTest	testIsEmpty	Se crea una pila y se agrega un elemento	“hola”	Se comprueba que la pila no esté vacía
12	StackTest	testIsEmpty	Se crea una pila y se agrega un elemento	“hola”	Se comprueba que el tamaño sea 1 y no esté vacía
13	StackTest	testIsEmpty	Se crea una pila y se agrega un elemento y se saca el elemento de la pila	“hola”	Se comprueba que la pila esta vacía
14	QueueTest	testEnqueue	Se crea una cola y se agrega un valor se utiliza el método enqueue	“hola”	Se revisa que la cola creada no sea NULL
15	QueueTest	testEnqueue	Se crea una cola y se agrega un valor se utiliza el método enqueue	“hola”	Se revisa que la cola el tamaño sea uno
16	QueueTest	testEnqueue	Se crea una cola y se agrega un valor se utiliza el método enqueue	“hola”	Se revisa que la cola no esté vacía

17	QueueTest	testGetFirst	Se crea una cola se agrega un valor se utiliza el método getFirst	“hola”	Se revisa que el elemento obtenido que es el primero sea el que se agrego
18	QueueTest	testGetFirst	Se crea una cola se agrega un valor se utiliza el método getFirst	“hola”	Se revisa la cola no esté vacía
19	QueueTest	testGetFirst	Se crea una cola se agrega un valor se utiliza el método getFirst	“hola”	Se revisa que al obtener el primer elemento el tamaño de la cola no cambia
20	QueueTest	testConsult	Se creo una cola se agrega un valor se utiliza el método Consult	“hola”	Se revisa que el elemento consultado de la cola sea el agregado y que la cola no esté vacía
21	QueueTest	testConsult	Se creo una cola se agrega un valor se utiliza el método Consult	“hola”	Se revisa que el elemento consultado sea diferente al de la cola sea el agregado y que la cola no esté vacía
22	QueueTest	testConsult	Se creo una cola se agrega un valor se utiliza el método Consult	“hola”	Se revisa que al consultar el elemento el tamaño de la cola no varia
23	QueueTest	testDequeue	Se creo una cola se agrega un valor y se utiliza el método dequeue	“hola”	Se revisa que la cola no esté vacía
24	QueueTest	testDequeue	Se creo una cola se agrega un valor y se utiliza el método dequeue	“hola”	Se revisa que el elemento recuperado sea el agregado
25	QueueTest	testDequeue	Se creo una cola se agrega un valor y se utiliza el método dequeue	“hola”	Se revisa que el tamaño de la cola varíe
26	QueueTest	testIsEmpty	Se crea una cola se agrega un valor se utiliza el método isEmpty	“hola”	Se verifica que la cola no está vacía
27	QueueTest	testIsEmpty	Se crea una cola se agrega	“hola”	Se verifica que la cola esta vacía
28	QueueTest	testIsEmpty	Se crea una cola se agrega un valor se utiliza el método isEmpty	“hola”	Se verifica que la cola no está vacía
29	HashTableTest	testAdd	Se crea un hash table se agrega una tupla	“test, holaMundo”	Se verifica que el tamaño de la tabla sea diferente de 0
30	HashTableTest	testAdd	Se crea un hash table se agrega una tupla	“test, holaMundo”	Se verifica que al obtener la tupla esta sea la ingresada
31	HashTableTest	testAdd	Se crea un hash table se agrega una tupla	“test, holaMundo”	Se verifica que la tabla esta vacía
32	HashTableTest	testSize	Se crea un hash table y se agregan dos tuplas	“test,hola” “test2,mundo”	Se verifica que el tamaño de la tabla sea dos
33	HashTableTest	testSize	Se crea un hash table y se agregan dos tuplas	“test,hola” “test2,mundo”	Se verifica que el tamaño de la tabla no sea 1

34	HashTableTest	testSize	Se crea una hash table y se agregan dos tuplas se extrae una	“test,hola” “test2,mundo”	Se verifica que el tamaño de la tabla sea uno
35	HashTableTest	testGetList	Se crea un hash table y se agregan dos tuplas	“test,hola” “test2,mundo”	Se verifica que al obtener los valores con el método get list estos sean iguales a los valores ingresados en las tuplas
36	HashTableTest	testGetList	Se crea un hash table y se agregan dos tuplas	“test,hola” “test2,mundo”	Se verifica que al obtener los valores con el método get list estos sean diferentes a los valores ingresados en las tuplas
37	HashTableTest	testGetList	Se crea un hash table y se agregan dos tuplas	“test,hola” “test2,mundo”	Se verifica que al obtener los valores con el método get list la lista no esté vacía
38	HashTableTest	testIsEmpty	Se crea un hash table se agrega una tupla	“test,hola”	Se verifica que el hash table no este vacía
39	HashTableTest	testIsEmpty	Se crea un hash table	null	Se verifica que el hash table este vacía
40	HashTableTest	testIsEmpty	Se crea un hash table se agrega y se elimina una tupla	“test,hola”	Se verifica que el hash table este vacía
41	HashTableTest	testRemove	Se crea un hash table Se agregan dos tuplas y se elimina una	“test,hola” “test2,mundo”	Se verifica que el tamaño de la tabla sea igual a 1.
42	HashTableTest	testRemove	Se crea un hash table Se agregan dos tuplas y se eliminan dos	“test,hola” “test2,mundo”	Se verifica que la tabla esta vacía
43	HashTableTest	testRemove	Se crea un hash table Se agregan dos tuplas y se elimina una	“test,hola” “test2,mundo”	Se verifica que la tabla no esté vacía
44	HashTableTest	testResearch	Se crea un hash table se agregan dos tuplas y se recupera cada valor	“test,hola” “test2,mundo”	Se compara los dos elementos y se comparan con los valores agregados
45	HashTableTest	testResearch	Se crea un hash table se agregan dos tuplas y se recupera cada valor	“test,hola” “test2,mundo”	Se compara los dos elementos y se comparan con los valores agregados se asegura que sean diferentes
46	HashTableTest	testResearch	Se crea un hash table se agregan dos tuplas y se recupera cada valor	“test,hola” “test2,mundo”	Se compara los dos elementos y se comparan con los valores agregados
47	PriorityQueueTest	testEnqueue	Se crea una cola y se agrega un valor se utiliza el método enqueue	“hola”	Se revisa que la cola creada no sea NULL
48	PriorityQueueTest	testEnqueue	Se crea una cola y se agrega un valor se utiliza el método enqueue	“hola”	Se revisa que la cola el tamaño sea uno

49	PriorityQueueTest	testEnqueue	Se crea una cola y se agrega un valor se utiliza el método enqueue	“hola”	Se revisa que la cola no esté vacía
50	PriorityQueueTest	testGetFirst	Se crea una cola se agrega un valor se utiliza el método getFirst	“hola”	Se revisa que el elemento obtenido que es el primero sea el que se agrego
51	PriorityQueueTest	testGetFirst	Se crea una cola se agrega un valor se utiliza el método getFirst	“hola”	Se revisa la cola no esté vacía
52	PriorityQueueTest	testGetFirst	Se crea una cola se agrega un valor se utiliza el método getFirst	“hola”	Se revisa que al obtener el primer elemento el tamaño de la cola no cambia
53	PriorityQueueTest	testConsult	Se creo una cola se agrega un valor se utiliza el método Consult	“hola”	Se revisa que el elemento consultado de la cola sea el agregado y que la cola no esté vacía
54	PriorityQueueTest	testConsult	Se creo una cola se agrega un valor se utiliza el método Consult	“hola”	Se revisa que el elemento consultado sea diferente al de la cola sea el agregado y que la cola no esté vacía
55	PriorityQueueTest	testConsult	Se creo una cola se agrega un valor se utiliza el método Consult	“hola”	Se revisa que al consultar el elemento el tamaño de la cola no varia
56	PriorityQueueTest	testDequeue	Se creo una cola se agrega un valor y se utiliza el método dequeue	“hola”	Se revisa que la cola no esté vacía
57	PriorityQueueTest	testDequeue	Se creo una cola se agrega un valor y se utiliza el método dequeue	“hola”	Se revisa que el elemento recuperado sea el agregado
58	PriorityQueueTest	testDequeue	Se creo una cola se agrega un valor y se utiliza el método dequeue	“hola”	Se revisa que el tamaño de la cola varíe
59	PriorityQueueTest	testIsEmpty	Se crea una cola se agrega un valor se utiliza el método isEmpty	“hola”	Se verifica que la cola no está vacía
60	PriorityQueueTest	testIsEmpty	Se crea una cola se agrega	“hola”	Se verifica que la cola está vacía
61	PriorityQueueTest	testIsEmpty	Se crea una cola se agrega un valor se utiliza el método isEmpty	“hola”	Se verifica que la cola no está vacía
63	ListSortsTest	testMergeSort	Escenario 1	"Cristian", "Morales", "1101", Tarjet.AHORROS, "123456", 20000, LocalDate.now() Santiago", "Hurtado", "4010", Tarjet.AHORROS, "123456", 10000, LocalDate.now());	Se verifica que se ordenen los clientes por cantidad de dinero en la cuenta se ordenen ascendentemente

				"Juan", "Morales", "3210", Tarjet.AHORROS, "123456", 30000, LocalDate.now());	
64	ListSortsTest	testMergeSort	Escenario 1	"Cristian", "Morales", "1101", Tarjet.AHORROS, "123456", 20000, LocalDate.now() Santiago", "Hurtado", "4010", Tarjet.AHORROS, "123456", 10000, LocalDate.now()); "Juan", "Morales", "3210", Tarjet.AHORROS, "123456", 30000, LocalDate.now());	Se verifica que se ordenen los clientes por cantidad de dinero en la cuenta se ordene descendentemente
65	ListSortsTest	testMergeSort	Escenario 1	"Cristian", "Morales", "1101", Tarjet.AHORROS, "123456", 20000, LocalDate.now() Santiago", "Hurtado", "4010", Tarjet.AHORROS, "123456", 10000, LocalDate.now()); "Juan", "Morales", "3210", Tarjet.AHORROS,	Se verifica con otro arreglado no ordenado y que estos difieran

				"123456", 30000, LocalDate.now());	
66	ListSortsTest	testSelectionSort	Escenario 1	"Cristian", "Morales", "1101", Tarjet.AHORROS, 24"123456", 20000, LocalDate.now() Santiago", "Hurtado", "4010", Tarjet.AHORROS, "123456", 10000, LocalDate.now()); "Juan", "Morales", "3210", Tarjet.AHORROS, "123456", 30000, LocalDate.now());	Se verifica que se ordenen los clientes por nombre ascendentemente
67	ListSortsTest	testSelectionSort	Escenario 1	"Cristian", "Morales", "1101", Tarjet.AHORROS, 24"123456", 20000, LocalDate.now() Santiago", "Hurtado", "4010", Tarjet.AHORROS, "123456", 10000, LocalDate.now()); "Juan", "Morales", "3210", Tarjet.AHORROS, "123456", 30000, LocalDate.now());	Se verifica que se ordenen los clientes por nombre se ordenen descendentemente

68	ListSortsTest	testSelectionSort	Escenario 1	"Cristian", "Morales", "1101", Tarjet.AHORROS, 24"123456", 20000, LocalDate.now() Santiago", "Hurtado", "4010", Tarjet.AHORROS, "123456", 10000, LocalDate.now()); "Juan", "Morales", "3210", Tarjet.AHORROS, "123456", 30000, LocalDate.now());	Se verifica con otro arreglado no ordenado y que estos difieran
69	ListSortsTest	testHeapSort	Escenario 1	"Cristian", "Morales", "1101", Tarjet.AHORROS, "123456", 20000, LocalDate.now() Santiago", "Hurtado", "4010", Tarjet.AHORROS, "123456", 10000, LocalDate.now()); "Juan", "Morales", "3210", Tarjet.AHORROS, "123456", 30000, LocalDate.now());	Se verifica que se ordenen los clientes por ID ascendentemente

70	ListSortsTest	testHeapSort	Escenario 1	"Cristian", "Morales", "1101", Tarjet.AHORROS, "123456", 20000, LocalDate.now() Santiago", "Hurtado", "4010", Tarjet.AHORROS, "123456", 10000, LocalDate.now()); "Juan", "Morales", "3210", Tarjet.AHORROS, "123456", 30000, LocalDate.now());	Se verifica que se ordenen los clientes por ID descendentemente
71	ListSortsTest	testHeapSort	Escenario 1	"Cristian", "Morales", "1101", Tarjet.AHORROS, "123456", 20000, LocalDate.now() Santiago", "Hurtado", "4010", Tarjet.AHORROS, "123456", 10000, LocalDate.now()); "Juan", "Morales", "3210", Tarjet.AHORROS, "123456", 30000, LocalDate.now());	Se verifica con otro arreglado no ordenado y que estos difieran
72	ListSortsTest	testQuickSort	Escenario 1	"Cristian", "Morales", "1101", Tarjet.AHORROS,	Se verifica que se ordenen los clientes por fecha de ingreso al banco se haga ascendentemente

				"123456", 20000, LocalDate.now() Santiago", "Hurtado", "4010", Tarjet.AHORROS, "123456", 10000, LocalDate.now()); "Juan", "Morales", "3210", Tarjet.AHORROS, "123456", 30000, LocalDate.now());	
73	ListSortsTest	testQuickSort	Escenario 1	"Cristian", "Morales", "1101", Tarjet.AHORROS, "123456", 20000, LocalDate.now() Santiago", "Hurtado", "4010", Tarjet.AHORROS, "123456", 10000, LocalDate.now()); "Juan", "Morales", "3210", Tarjet.AHORROS, "123456", 30000, LocalDate.now());	Se verifica que se ordenen los clientes por fecha de ingreso al banco se haga descendentemente
74	ListSortsTest	testQuickSort	Escenario 1	"Cristian", "Morales", "1101", Tarjet.AHORROS, "123456", 20000, LocalDate.now()	Se verifica con otro arreglado no ordenado y que estos difieran

				Santiago", "Hurtado", "4010", Tarjet.AHORROS, "123456", 10000, LocalDate.now()); "Juan", "Morales", "3210", Tarjet.AHORROS, "123456", 30000, LocalDate.now());	