TESTS:

Configuration of stages

Name	Class	Stage
setup1	StackTest	Creates an empty String Stack.
setup1	QueueTest	Creates an empty Integer queue.
setup1	PriorityQueueTest	Creates an empty Integer priority queue.
setupEmpty	HashTableTest	Create an empty hash table with key = Integer and value = String.
setupFull	HashTableTest	Creates a hash table with 60 filled positions, has key = Integer and value = String
setUpNormal	GameControllerTest	The gameController class is initialized and a player name list is created to which the following is added: 1. "Player 1" 2. "Player 2" 3. "Player 3". Finally the game is initialized with the starGame method that takes as parameter the list that was created.
setUpExtreme	GameControllerTest	The gameController class is initialized and a list of player names is created to which five players are added. In addition, the game is initialized with the starGame method that takes as parameter the list that was created.
setUpSpecial	GameControllerTest	IInvolve two players. Player 1's hand is laid out with the following red cards in order: one REVERSE card, one DRAW TWO card, and two regular NUMBER cards. Player 2's hand contains a red NUMBER card followed by a red SKIP card. The setup method prepares the game by setting up the game deck to start with these red cards and sets the initial state of the game to simulate the game involving special actions and card sequences to try.

Design of test cases:

Objective of the test: Verify that the push, pop, peek and isEmpty methods work correctly for the operability of the Stack class.				
Class	**Method**	**Stage**	**Input**	**Output**
Stack	push	setup1	Add: “Blue 3”	The stack is not empty, and when we request its last element, it returns 'Blue 3'.
Stack	push	setup1	Add: null	The stack is not empty, and when we request its last element, it returns 'null'.
Stack	push	setup1	Add: “Blue 3”, “Red Skip”	When requesting the last element of the stack, it returns 'Red Skip'.
Stack	pop	setup1		It is expected to throw an exception since the stack is empty.
Stack	pop	setup1	Add: “Green Reverse”	The stack should be empty, and the value 'Green Reverse' should be returned.
Stack	pop	setup1	Add: “Yellow 2”, “Red Draw Two”	It is expected that when the last element of the stack is requested, it returns 'Yellow 2'.
Stack	peek	setup1		It is expected to throw an exception since the stack is empty.
Stack	peek	setup1	Add: “Red 7”	The stack should not be empty, and when requesting its last element, it should return 'Red 7'.
Stack	peek	setup1	Add: “Blue Skip”, “Green 4”	When requesting the last element of the stack, it should return 'Green 4'.
Stack	isEmpty	setup1		It should return true, indicating that the stack is empty.
Stack	isEmpty	setup1	Add: “Red 0”	It should return false, indicating that the stack has elements.

Objective of the test:
Verify that the *enqueue, dequeue and isEmpty* methods work correctly for the operability of the Queue class.

Class	Method	Stage	Input	Output
Queue	enqueue	setup1	Add: 5	The queue should not be empty.
Queue	enqueue	setup1	Add: Integer.MAX	The element is dequeued, and Integer.MAX should be obtained.
Queue	enqueue	setup1	Add: 10, 20	The last element, 10, is dequeued, followed by the next element, 20.
Queue	dequeue	setup1	Add: 1, 2	1 and 2 are dequeued, then it is checked if the queue is empty.
Queue	dequeue	setup1		Se espera que lance una excepción pues no existen elementos en la cola.
Queue	dequeue	setup1	Add: 5, 10, 15	The first element is dequeued, and it is expected to return the number 10.
Queue	isEmpty	setup1	Add: 1	Se espera que envíe un false ya que la cola no está vacía.
Queue	isEmpty	setup1	Add: 5	The element is dequeued, and it is expected to return true since the queue is empty.
Queue	isEmpty	setup1	Add: enqueue 1000 elements	The same elements are dequeued, and it is expected to return true since the queue is empty.

Objective of the test:
Verify that the methods enqueue, dequeue, peek, isEmpty, size, increasePriority and prioritizeLowest work correctly for the operability of the PriorityQueue class.

Class	Method	Stage	Input	Output
Priority Queue	enqueue	setup1	5, 1	The priority queue must not be empty so it is expected to return the first and only element in the queue.
Priority Queue	enqueue	setup1	5,Integer.MAX_VALUE	The priority queue is not empty so it is expected to return the only element by taking the first one in the queue.
Priority Queue	enqueue	setup1	1. 5, 2 2. 10, 1 3. 15, 3	The queue has three elements, it is expected that taking the first element of the queue will return the one with the highest priority, i.e. 15.
Priority Queue	dequeue	setup1	Add: 1. 5, 1 2. 10, 2	The queue has two elements and the one with the highest priority is removed, it is expected to return and remove 10.
Priority Queue	dequeue	setup1	5, 1	There is a single element in the priority queue, the queue is expected to be empty when it is removed.
Priority Queue	dequeue	setup1	Add: 1. 5, 2 2. 10, 1 3. 15, 3	The first element in the queue is removed, at the time of obtaining the first element in the queue must be the one that had the second highest priority, i.e. 5.
Priority Queue	peek	setup1	Add: 5, 1	The first element is obtained, in this case being the only element it must return 5.
Priority Queue	peek	setup1		It is expected to throw an exception since the priority queue is empty.
Priority Queue	peek	setup1	Add: 1. 5, 2 2. 10, 1	It is expected to return 5, the element with the highest priority.
Priority Queue	isEmpty	setup1	Add: 5, 1	An element is deleted, the queue must be empty.
Priority Queue	isEmpty	setup1	Add: 1. 5, 1 2. 10, 2	Two elements are removed, the queue must be empty.
Priority Queue	size	setup1	Add: 5, 1	Returns a 1 since there is only one element in the queue.
Priority	size	setup1	Add 1000	Returns 1000, as this is the number

Queue			elements	of elements in the queue.
Priority Queue	size	setup1	Add: 1. 5, 1 2. 10, 2	An element is removed, it is expected to return 1, the number of elements in the queue.
Priority Queue	increase Priority	setup1	Add: 1. 5, 1 2. 10, 2	When calling the method, in the first position with the highest priority the 10 must be found.
Priority Queue	increase Priority	setup1	5,Integer.MAX_VALUE -1	When calling the method, in the first position with the highest priority the 5 must be found.
Priority Queue	increase Priority	setup1	Add: 1. 5, 1 2. 10, 1	When calling the method, it must be true that when the first element is eliminated, 5 comes out and at the same time when the first element is called, 10 comes out.
Priority Queue	prioritizeLowest	setup1	Add: 1. 5, 3 2. 10, 1 3. 15, 2	When calling the method and asking for the first element in the priority queue, it should return 10.
Priority Queue	prioritizeLowest	setup1	Add: 5, 1	Once the method is called, the first element of the queue must be 5.
Priority Queue	prioritizeLowest	setup1	Add: 1. 5, 1 2. 10, 2 3. 15, 1	When calling the method, it must be true that the first position is either 5 or 15.

Objective of the test:
Verify that the methods put, get, remove, isEmpty and size work correctly for the operability of the Hash Table class.

Class	Method	Stage	Input	Output
HashTable	put	setupEmpty	1, Blue 3	The first position of the table is called and must return "Blue 3".
HashTable	put	setupEmpty	1. 1, “Red Reverse” 2. 38, “Green Draw Two”	In position 1 must be the value "Red Reverse" and in position 38 the value "Green Draw Two", also the size must be set to 2.
HashTable	put	setupEmpty	1. 1, “Red Skip” 2. 1, “Yellow 7”	It must be fulfilled that the value of the key that already exists is replaced, so the value of position 1 must be "Yellow 7".
HashTable	get	setupFull	Add: 100, "Wild Draw Four"	It is expected that when the 100 position is obtained, the value will be "Wild Draw Four".
HashTable	get	setupFull		When getting the value at position -1, it must return a null.
HashTable	get	setupEmpty		When getting the value at position 1, it must return a null.
HashTable	remove	setupFull	10	When getting the value at position 10, it must return a null.
HashTable	remove	setupFull	null	Returns a null when trying to get a null position.
HashTable	remove	setupFull	Add: 1, null Remove: 1	When asking for the first position it must show a null.
HashTable	isEmpty	setupEmpty		It must throw a true, meaning that the table is empty.
HashTable	isEmpty	setupEmpty	Add: 1, “Blue 7” Remove: 1	It must throw a true, meaning that the table is empty.
HashTable	size	setupFull		It should return 60 which is the number of elements in the table.
HashTable	size	setupEmpty	Add: 1, "Green Reverse" Remove: 1	The element is added and the size must be 1, the element is removed and the size must be 0.

Objective of the test: To verify that the 's methods correctly handle game states and player interactions in a controlled scenario where cards are present.				
Class	**Method**	**Stage**	**Input**	**Output**
GameController	isGameOver	setUpNormal		should return false
GameController	getPlayerQueue size	setUpNormal	3	should return 3
GameController	currentPlayer	setUpNormal		should return "Player 3".
GameController	handSizePlayer	setUpNormal		should return 7.
GameController	currentCard	setUpNormal		should not return null.
GameController	currentPlayerCardList	setUpNormal		should not return an empty string.
GameController	isActiveSpecialcard	setUpNormal		should return false.
GameController	getAuxiliaryCard	setUpNormal	Card.Color.BLUE	should return .Card.Color.BLUE
GameController	handleSpecialCardEffect	setUpNormal		should not return a null message.
GameController	currentPlayer nextTurn currentPlayer	setUpNormal		After , should return a different player's name than before.
GameController	handSizePlayer drawCard	setUpNormal	initialHandSize + 1	After , should return initial hand size plus one.
GameController	checkGameOver	setUpNormal		should return false.
GameController	getPlayerQueue size	setUpExtreme	5	should return 5, indicating the maximum number of players is set.
GameController	drawCard	setUpExtreme	1 (attempt to draw one card)	A try-catch block should catch an Exception, indicating there are no cards to draw.
GameCo	handleSpecial	setUpE	"A card with	should return a message about the

ntroller	CardEffect	xtreme	skip effect was used against you. You lost your turn."	SKIP effect.
GameController	getDeck} getPlayDeck peek	setUpExtreme	cardId	will give equality between both cards
GameController	playCard	setUpExtreme	Index of the draw two card (0)	should return true
GameController	isGameOver	setUpExtreme		should return true, indicating the player has won by playing their last card.
GameController	currentPlayer	setUpSpecial		Name of the player expected to start, "Player 2".
GameController	playCard	setUpSpecial	Index of the reverse card (0)	True, indicating the card was played successfully.
GameController	currentPlayer	setUpSpecial		Name of the next player in turn, "Player 1".
GameController	playCard	setUpSpecial	Index of the draw two card (0)	True, indicating the card was played successfully.
GameController	playCard	setUpSpecial	Index of the draw two card (0)	True, indicating a special card effect (DRAW TWO) is active.
GameController	isActiveSpecialcard	setUpSpecial		String message detailing the DRAW TWO card effect.
GameController	handleSpecialCardEffect	setUpSpecial	Index of next card after handling special effect	True, indicating the card was played successfully.
GameController	playCard	setUpSpecial	Index of the draw two card (0)	Integer, the count of cards in the player's hand (3).
GameController	playerQueue peek getHand size	setUpSpecial	Index of card in player's hand	True, indicating the SKIP card was played successfully.
GameController	playCard	setUpSpecial	Index of the draw two card (0)	True, if the game is over, indicating a player has won.

GameController	playCard	setUpSpecial	Index of the draw two card (0)	Name of the player expected to start, "Player 2".
GameController	isGameOver	setUpSpecial	Index of the reverse card (0)	True, indicating the card was played successfully.