**Test Cases**

| **Name** | **Class** | **Scenario** |
| --- | --- | --- |
| setUp1 | DirectGraphList | A graph with 5 vertex without edges. |
| setUp2 | DirectGraphList | A Graph without vertex and edges. |

| setUp1 | DirectedGraphMatrix | Initialize a graph with 5 vertices. |
| --- | --- | --- |
| setUp2 | DirectedGraphMatrix | A Graph without vertex and edges.Empty. |

| **Class** | **Method** | **Scenario** | **Input** | **Expected Result** |
| --- | --- | --- | --- | --- |
| Graph | AddVertex | setUp2 | insert a vertex to an empty graph | A new vertex is added to the size of the vertices. |
| Graph | AddVertex | setUp1 | Add Vertex To Graph that already has some vertices | A new vertex is added to the size of the vertices. |
| Graph | AddVertex | setUp1 | add a vertex with the same value that a vertex that already exists in the graph | I shouldn't be able to add it either because it doesn't allow it or it throws an exception message. |
| Graph | AddEdge | setUp1 | Add a edge  between two  existing vertex.  Graph with 5 vertex.  Initial vertex: 1  Final vertex: 2 | The edge is added successfully. The size of the edges for the source vertex should increase by 1 and the destination vertex should match the specified destination. |
| Graph | AddEdge | setup1 | Add a edge with weight  between two  existing vertex.  Graph with 5 vertex.  Initial vertex: 1  Final vertex: 2  weight: 10 | The weighted edge is added successfully. The size of the edges for the source vertex should increase by 1. The destination vertex should match the specified destination. The weight of the added edge should be 10. |
| Graph | AddEdge | setup1 | Add a edge with weight  between two  existing vertex.  Graph with 5 vertex.  Initial vertex: 1  Final vertex: 2  weight: 0 | The weighted edge is added successfully. The size of the edges for the source vertex should increase by 1. The destination vertex should match the specified destination. The weight of the added edge should be 0. |
| Graph | BFS | setup1 | add the value of the start vertex. | Each vertex reachable from the starting vertex should have a distance d assigned to it, which is not equal to Integer.MAX\_VALUE. |
| Graph | FloydWarshall | setup1 | add the value of the start vertex. | the minimum cost ways from the start vertex to other vertices. |

| **Class** | **Method** | **Scenario** | **Input** | **Expected Result** |
| --- | --- | --- | --- | --- |
| Graph | testAddVertexToEmptyGraph | Insert a vertex to an empty graph | Set up with setUp2() and then call emptyGraph.addVertex(1) | A new vertex is added. The size of the vertices should be 1. |
| Graph | AddVertexToGraphWithExistingVertices | Add a vertex to a graph that already has some vertices | Set up with setUp1() and then call graphWithVertices.addVertex(7) | A new vertex is added. The size of the vertices should be 6. |
| Graph | AddVertexWithSameID | Add a vertex with the same value as an existing vertex in the graph | Set up with setUp1() and then call graphWithVertices.addVertex(1) | An IllegalArgumentException is thrown, indicating that duplicate vertices are not allowed. |
| Graph | AddEdgeBetweenExistingVertices | Add an edge between two existing vertices | Set up with setUp1(), then call graphWithVertices.addEdge(1, 2, 2) | The edge is added successfully. graphWithVertices.hasEdge(1, 2) should return true. |
| Graph | AddWeightedEdge | Add a weighted edge between two existing vertices | Set up with setUp1(), then call graphWithVertices.addEdge(1, 2, 10) | The weighted edge is added successfully. The edge weight between vertices 1 and 2 should be 10. |
| Graph | AddEdgeBetweenExistingVerticesWithWeightZero | Add a weighted edge with weight 0 between two existing vertices | Set up with setUp1(), then call graphWithVertices.addEdge(1, 2, 0) | The weighted edge is added successfully. The edge weight between vertices 1 and 2 should be 0. |
| Graph | AddEdgeBetweenNonExistingVertices | Add an edge between non-existing vertices | Set up with setUp1(), then call graphWithVertices.addEdge(1, 10, 10) | An IllegalArgumentException is thrown, indicating that the vertices do not exist. |
| Graph | BFS | Perform BFS starting from a specified vertex | Set up with setUp1(), then add edges and call graphWithVertices.bfs(1) | Each vertex reachable from the starting vertex should have a distance d assigned to it, which is not equal to Integer.MAX\_VALUE. |
| Graph | FloydWarshall | Perform Floyd-Warshall algorithm to find shortest paths between all pairs of vertices | Set up with setUp1(), then add edges and call graphWithVertices.floydWarshallM() | The returned matrix should correctly represent the shortest path distances between all pairs of vertices, with the expected distances at specified positions. |

| **Class** | **Method** | **Scenario** | **Input** | **Expected Result** |
| --- | --- | --- | --- | --- |
| VentanaJuegoController | createGraph | VerifyVertices | Call createGraph() | Graph should have 50 vertices |
| VentanaJuegoController | createGraph | VerifyEdges | Call createGraph() | Graph should have more than 50 edges |
| VentanaJuegoController | createGraph | VerifyExtraEdges | Call createGraph() | Graph should have extra edges for vertices between 10 and 15 |
| VentanaJuegoController | drawGraph | VerifyBackground | Call drawGraph() after loadBackgroundImage() | Background image should be loaded |
| VentanaJuegoController | drawGraph | VerifyCharacterDrawn | Call drawGraph() after loadPlayerImage() | Player image should be loaded |
| VentanaJuegoController | drawGraph | VerifyEnemyDrawn | Call drawGraph() after loadEnemyImage() | Enemy image should be loaded |
| VentanaJuegoController | loadBackgroundImage | Success | Call loadBackgroundImage() | Background image should be loaded successfully |
| VentanaJuegoController | loadBackgroundImage | Error | Load an incorrect image path | Background image should remain null if there is an error |
| VentanaJuegoController | loadBackgroundImage | Path | Call loadBackgroundImage() | Background image path should be correct |
| VentanaJuegoController | handleCanvasClick | VerifyCharacterMovement | Simulate MouseEvent on canvas with energy 100 | Character position should change after clicking on a valid vertex |
| VentanaJuegoController | handleCanvasClick | NotEnoughEnergy | Simulate MouseEvent on canvas with energy 0 | Character position should not change due to insufficient energy |
| VentanaJuegoController | handleCanvasClick | Victory | Simulate MouseEvent on canvas with victory vertex and energy 100 | Character should move to the victory vertex |
| VentanaJuegoController | calculateShortestPaths | Initialization | Call calculateShortestPaths() | Distances and next arrays should be initialized |
| VentanaJuegoController | calculateShortestPaths | DistanceCalculation | Call calculateShortestPaths() | Distances should be correctly calculated |
| VentanaJuegoController | calculateShortestPaths | PathUpdate | Call calculateShortestPaths() | Next array should be updated to reflect the shortest path |
| VentanaJuegoController | drawCharacter | VerifyDraw | Call drawCharacter() after loadPlayerImage() | Player image should be loaded |
| VentanaJuegoController | drawCharacter | Positioning | Call drawCharacter() | Character should be drawn at the correct position |
| VentanaJuegoController | drawCharacter | NullImage | Call drawCharacter() with null image | Player image should be null |
| VentanaJuegoController | drawEnemy | VerifyDraw | Call drawEnemy() after loadEnemyImage() | Enemy image should be loaded |
| VentanaJuegoController | drawEnemy | Positioning | Call drawEnemy() | Enemy should be drawn at the correct position |
| VentanaJuegoController | drawEnemy | NullImage | Call drawEnemy() with null image | Enemy image should be null |