**Test de los Grafos**

|  |  |  |
| --- | --- | --- |
| Nombre | Clase | Escenario |
| setUpEscenary1 | Graph | Int[3][3] |
| setUpEscenary2 | Graph | Vertex<Island>(new Island(0, 0, 0)));  Vertex<Island>(new Island(0, 0, 1)));  Vertex<Island>(new Island(0, 0, 2))); |
| setUpEscenary3 | Graph | {0,3,4},  {3,0,1},  {4,1,0} |
| setUpScenary2 | GraphArray | VertexArray <Island>(new Island(50, 50, 0))  ArrayList< VertexArray <Island>> connections = {null}  Int[][] weigths = []  VertexArray <Island>(new Island(50, 50, 0))  ArrayList< VertexArray <Island>> connections = {vertex1}  Int[][] weigths = [1]  VertexArray <Island>(new Island(50, 50, 2))  ArrayList< VertexArray <Island>> connections = {Vertex1, Vertex2}  Int[][] weigths = [2, 3] |
| setUpScenary1 | GraphArray | new GraphArray<Island>() |
| setUpScenary4 | Graph y Graph Array | Int [][] mapa |
| SetUpScenary5 | Graph y GraphArray | Int[][] mapa |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Clase | Metodo | Escenario | Valores de entrada | Resultado |
| Graph | testGraph() | setUpEscenary1 | Graph<Integer> graph = new Graph<>(); | El método constructor genera correctamente el grafo y lo retorna un nulo. |
| Graph | testConstructor2() | setUpEscenary1 | ArrayList<Vertex<Island>> vertexes = setupScenary2();    Graph<Island> graph = new Graph<>(matrix,vertexes); | EL grafo y el arraylist de los vértices inicializado y que no retorne un nulo. |
| Graph | testAddVertexes() |  | int[] edges = {0,1};  int[] weights = {2,3};  Island island = Island(0,0,3); | Matriz de adyacencia {0,0,0,2},  {0,0,0,3},  {0,0,0,0},  {2,3,0,0} |
| Graph | testEnlargeMatrix() | setupScenary2() |  | El tamaño de la matriz es el esperado por su método constructor. |
| Graph | getWeight() | setupScenary2() |  | Retorna el tamaño de la matriz en cada caso |
| Graph | getConection() | setupScenary3() |  | Retorna los vértices del grafo y sus relaciones |
| GraphArray | addVertex( ) | setUpScenary1( ) | VertexArray<Island>(new Island(50, 50, 0)) | Comprueba que se añada un vértice de forma correcta comprobando la raíz |
| GraphArray | listOfVertex( ) | setUpScenary2( ) |  | Comprueba que el tamaño de la lista sea igual a la cantidad de vértices insertados |
| GraphArray | adyacencyMatrix( ) | setUpScenary2( ) |  | Comprueba que la matriz de adyacencia sea la correcta para los vértices insertados |
| Graph y  GraphArray | floydWarshall() | SetUpScenary4() |  | Comprueba los caminos mínimos dentro de la matriz a mapa |
| Graph y  GraphArray | dijsktra() |  |  | Comprueba el camino mínimo con un arreglo de vértices previos |

**Test de la Solución**

|  |  |  |
| --- | --- | --- |
| Nombre | Clase | Escenario |
| setUpEscenary1 | PirateChase | PirateChase pc |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Clase | Método | Escenario | Valores de entrada | Resultado |
| PirateChase | MakeMapEasy() | setUpEscenary1 |  | Comprueba que el mapa fácil se esté creando correctamente  (De la misma manera se crean los otros mapas) |
| PirateChase | ClickOnIsland() | setUpEscenary1 |  | Comprueba que al hacer click el usuario se le asigne la isla correspondiente, restando la energía y verificando que gano o perdió |
| PirateChase | CheckEnergy() | setUpEscenary1 |  | Comprueba si una isla tiene poder de recarga de energía y se lo suma al usuario |
| PirateChase | SetMorganMovements(int[] dijkstraPrev) | SetUpEscenary1 | Arreglo de previos generado por el dijkstra | Comprueba la correcta asignación de movimientos a partir del dijkstra |
| PirateChase | RecalculatePosUser(int newIsland) | setUpEscenary1 | Entero representando la nueva isla | Verifica que si se esté acomodando el usuario en la nueva isla |
| PirateChase | RecalculatePosMorgan(int newIsland) | setUpEscenary1 | Entero representando la nueva isla | Verifica que si se esté acomodando a morgan en la nueva isla |