**Diseño de Pruebas unitarias.**

Especificación de Escenarios de prueba:

|  |  |  |
| --- | --- | --- |
| Nombre: | Clase: | Especificación: |
| Stage1 | Date | Vacío |
| Stage2 | Flight | Vacío |
| Stage3 | Airport | Vacío |
| Stage4 | Airport | Una instancia de Airport ya ha sido creada. |
| Stage5 | Airport |  |

Diseño de pruebas:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prueba No 1: | Objetivo: Probar que el programa crea correctamente instancias de las diferentes clases del modelo. | | | |
| Clase | **Método** | **Escenario** | **Entradas** | **Salidas** |
| DateTest | Date() | Stage1 |  | Se crea un objeto de la clase Date con datos al azar |
| DateTest | Date(int,int,int,int,int) | Stage1 | d = 12  m = 6  y = 2019  h = 14  m = 16 | Se crea un objeto de la clase Date con los datos enviados por parámetro.  getDate() = “2019-06-12”  getTime() = “02:16-PM”  toString() = “2019-06-12 – 02:16-PM” |
| DateTest | Date(String) | Stage1 | s=“2019-06-12 – 02:16-PM” | Se traduce de la cadena el objeto date adecuado. |
| FlightTest | Flight(String, String, String, int) | Stage2 | al = “Copa Airlines”  fn = “C80192”  d = “Paris”  bg = 2 | Se crea un objeto de la clase Flight con los datos enviados por parámetro. |
| AirportTest | Airport() | Stage3 |  | Se crea el objeto Airport. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prueba No 2: | Objetivo: Probar que el aeropuerto se llena sin mayor problema con datos al azar y que cada vuelo tiene un identificador único. | | | |
| Clase | **Método** | **Escenario** | **Entradas** | **Salidas** |
| AirportTest | generateFlights(int) | stage4 | 1000 | Ninguna de las 1000 entradas es creada con errores, todos los vuelos tienen un identificador diferente. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prueba No 3: | Objetivo: Probar que los métodos de búsqueda y ordenamiento funcionan correctamente. | | | |
| Clase | **Método** | **Escenario** | **Entradas** | **Salidas** |
| AirportTest | search(Sortings, String) | Stage5 | Sortings.AIRLINE, “Avianca” | Una lista de un único elemento con el tercer elemento de la lista original en ella. |
| AirportTest | search(Sortings, String) | Stage5 | Sortings.AIRLINE, “Ecolines” | Se entrega una lista vacía. |
| AirportTest | Search(Sortings, String) | Stage5 | Sortings.DESTINATION, “Bogota” | Una lista de un único elemento con el cuarto elemento de la lista original en ella. |
| AirportTest | search(Sortings, String) | Stage5 | Sortings.DESTINATION, “Ciudad de Mexico” | Se entrega una lista vacía. |
| AirportTest | Search(Sortings, String) | Stage5 | Sortings.FLIGHT\_NUMBER, “C80192” | Una lista de un único elemento con el segundo elemento de la lista original en ella. |
| AirportTest | Search(Sortings, String) | Stage5 | Sortings.FLIGHT\_NUMBER, “D80192” | Se entrega una lista vacía. |
| AirportTest | Search(Sortings, String) | Stage5 | Sortings.BOARDING\_GATE, 4 | Una lista de un único elemento con el cuarto elemento de la lista original en ella. |
| AirportTest | Search(Sortings, String) | Stage5 | Sortings.BOARDING\_GATE, 5 | Se entrega una lista vacía. |
| AirportTest | Sort() | Stage5 | Sortings.AIRLINE | La lista se ordena en el siguiente orden: cuarto, tercero, segundo, primero. |
| AirportTest | Sort() | Stage5 | Sortings.DESTINATION | La lista se ordena en el siguiente orden: cuarto, primero, segundo, tercero. |
| AirportTest | Sort() | Stage5 | Sortings.FLIGHT\_NUMBER | La lista se ordena en el siguiente orden: tercero, cuarto, segundo, primero. |
| AirportTest | Sort() | Stage5 | Sortings.BOARDING\_GATE | La lista se ordena en el siguiente orden: primero, segundo, tercero, cuarto. |