**Formato de escenarios y casos de uso**

**Configuración de los Escenarios**

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Clase** | **Escenario** |
| setupStage1 | QueueTest | Cola vacia |
| setupStage2 | QueueTest | Cola con 2 passengers object  Passenger p1( name:” Santiago” ,surname:” Hernandez” first Class:false, seat:6, miles:100, id: 1105362267)  object Passenger p2( name:” Sergio” ,surname:” florez” first Class: false, seat:15, miles:1000, id: 6543217890) |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Clase** | **Escenario** |
| setupStage1 | PriorityQueueIngress | PriorityQueue vacia |
| setupStage2 | PriorityQueueIngress | Con 3 pasajeros  Passenger p1( name:” Valeria” ,surname:” Ortiz” first Class:true, seat:6, miles:50, id: 1105362267)  object Passenger p2( name:” pipe” ,surname:” bueno” first Class: true, seat:22, miles: 897, id: 654321789)  object Passenger p2( name:” Danielle” ,surname:” marsh” first Class: true, seat:7, miles: 1111, id: 99999999) |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Clase** | **Escenario** |
| setupStage1 | PriorityQueueOut | PriorityQueue vacia |
| setupStage2 | PriorityQueueOut | Con 3 pasajeros  Passenger p1( name:” Valeria” ,surname:” Ortiz” first Class:true, seat:6, miles:50, id: 1105362267)  LocalTime.of(15, 30, 0)  object Passenger p2( name:” pipe” ,surname:” bueno” first Class: true, seat:22, miles: 897, id: 654321789)  LocalTime.of(14, 30, 0)  object Passenger p2( name:” Danielle” ,surname:” marsh” first Class: true, seat:7, miles: 1111, id: 99999999)  LocalTime.of(16, 30, 0) |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Clase** | **Escenario** |
| setupStage1 | HashTable | Hashtable vacia, con capacidad de 3 |
| setupStage2 | HashTable | Hash table con 3 pasajeros  Passenger p1( name:” Valeria” ,surname:” Ortiz” first Class:true, seat:6, miles:50, id: 12345678)  object Passenger p2( name:” pipe” ,surname:” bueno” first Class: true, seat:22, miles: 897, id: 39876543)  object Passenger p2( name:” Danielle” ,surname:” marsh” first Class: true, seat:7, miles: 1111, id: 99999999) |
|  |  |  |

\* El nombre de los escenarios puede ser setupStage1, setupStage2, etc.

\* La clase es la clase de testing correspondiente al modelo donde acontece el escenario. Por ejemplo si usted está probando User, clase será UserTest.

\* El escenario es la descripción de las condiciones iniciales del escenario.

**Diseño de Casos de Prueba**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** Probar el correcto funcionamiento del método enqueue | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado esperado** |
| Queue | enqueue | setupStage1 | Passenger p1( name:” Santiago” ,surname:” Hernandez” first Class:false, seat:6, miles:100, id: 1105362267) | Al imprimir el queue, nos debe imprimir “name=Santiago” |
| Queue | enqueue | setupStage2 | object Passenger p2( name:” Sergio” ,surname:” florez” first Class: false, seat:15, miles:1000, id: 654321789)  Passenger p1( name:” Santiago” ,surname:” Hernandez” first Class:false, seat:6, miles:100, id: 1105362267) | Al imprimir el queue, nos debe imprimir  “name=Santiagoname=Sergioname=Sergioname=Santiago” |
| Queue | enqueue | setupStage1 | object Passenger p2( name:” Sergio” ,surname:” florez” first Class: false, seat:15, miles:1000, id: 654321789) | Al imprimir el queue, nos debe imprimir “name=Sergio” |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** Probar el correcto funcionamiento de dequeue | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado esperado** |
| Queue | dequeue | setupStage2 | Llamaremos el método una vez | Debería luego imprimir “Santiago” |
| Queue | dequeue | setupStage2 | Llamaremos el método dos vez | Juntaríamos los 2 elementos y debería impirimir “Santiago Sergio” |
| Queue | dequeue | setupStage1 | LLamariamos el método una vez | Debería soltar la excepción EmptyStackException(); |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** Probar el correcto funcionamiento de peek | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado esperado** |
| Queue | peek | setupStage2 | Llamaremos el método una vez | Debería luego imprimir “Santiago” |
| Queue | peek | setupStage2 | Llamaremos el método dos vez | Juntaríamos los 2 elementos y debería impirimir “SantiagoSantiago” |
| Queue | peek | setupStage1 | LLamariamos el método una vez | Debería soltar la excepción EmptyStackException(); |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** Probar el correcto funcionamiento de insert de PriotiyQueueIngress | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado esperado** |
| PriotiyQueueIngress | insert | setupStage1 | Passenger p1( name:” Valeria” ,surname:” Ortiz” first Class:true, seat:6, miles:50, id: 1105362267)  object Passenger p2( name:” pipe” ,surname:” bueno” first Class: true, seat:22, miles: 897, id: 654321789) | Debería luego imprimir “name=pipename=Valeria” |
| PriotiyQueueIngress | insert | setupStage2 | object Passenger p2( name:” carlos” ,surname:” hernandez” first Class: true, seat:22, miles: 897, id: 654321789)  object Passenger p2( name:” albert” ,surname:” hernandez” first Class: true, seat:22, miles: 897, id: 6543289) | Juntaríamos los 2 elementos y debería impirimir “name=Daniellename=albertname=pipename=Valerianame=Carlos” |
| PriotiyQueueIngress | insert | setupStage2 | Passenger p1( name:” Valeria” ,surname:” Ortiz” first Class:true, seat:6, miles:50, id: 1105362267) | Debería impimir” name=Daniellename=Valerianame=pipename=Valeria” |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** Probar el correcto funcionamiento de delete de PriotiyQueueIngress | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado esperado** |
| PriotiyQueueIngress | delete | setupStage2 | Aplicamos el metodo una vez | Debería luego imprimir “name=pipename=Valeria” |
| PriotiyQueueIngress | delete | setupStage2 | Aplicamos el metodo 3 vez | Debe retornar “”; |
| PriotiyQueueIngress | delete | setupStage1 | Aplicamos el metodo una vez | Debe mostrar un NoSuchElementException |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** Probar el correcto funcionamiento de insert de PriotiyQueueOut | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado esperado** |
| PriotiyQueueOut | insert | setupStage1 | Passenger p1( name:” Valeria” ,surname:” Ortiz” first Class:true, seat:6, miles:50, id: 1105362267)  LocalTime.of(15, 30, 0)  object Passenger p2( name:” pipe” ,surname:” bueno” first Class: true, seat:22, miles: 897, id: 654321789)  LocalTime.of(15, 30, 0) | Debería luego imprimir “name=Valerianame=pipe” |
| PriotiyQueueInOut | insert | setupStage2 | object Passenger p2( name:” carlos” ,surname:” hernandez” first Class: true, seat:22, miles: 897, id: 654321789)  LocalTime.of(14, 30, 0)  object Passenger p2( name:” albert” ,surname:” hernandez” first Class: true, seat:7, miles: 897, id: 6543289)  LocalTime.of(16, 30, 0) | Juntaríamos los 2 elementos y debería impirimir “name=pipename=Carlosname=Daniellename=Valerianame=albert” |
| PriotiyQueueOut | insert | setupStage2 | Passenger p1( name:” Valeria” ,surname:” Ortiz” first Class:true, seat:8, miles:50, id: 1105362267)  LocalTime.of(14, 27, 0) | Debería impimir” name=Valerianame=pipename=Daniellename=Valeria” |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** Probar el correcto funcionamiento de delete de PriotiyQueueOut | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado esperado** |
| PriotiyQueOut | delete | setupStage2 | Aplicamos el metodo una vez | Debería luego imprimir “name=Daniellename=Valeria” |
| PriotiyQueOut | delete | setupStage2 | Aplicamos el metodo 3 vez | Debe retornar “”; |
| PriotiyQueueOut | delete | setupStage1 | Aplicamos el metodo una vez | Debe mostrar un NoSuchElementException |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** Probar el correcto funcionamiento de delete del HashTable | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado esperado** |
| HashTable | insert | setupStage1 | Passenger p1 = new Passenger("Valeria","Ortiz",true,6,50,12345678);  Y como clave el nombre | Debería luego imprimir “name=Valeria,” |
| HashTable | insert | setupStage2 | Passenger p1 = new Passenger("Cole","Min",true,9,1000,11099205);  Passenger p2 = new Passenger("Dani","Ramirez",true,7,1111,99999999);  Como clave el nombre | Debe retornar “name=Cole, name=Valeria, name=pipe, name=Dani, ”; |
| HashTable | insert | setupStage2 | Passenger p2 = new Passenger("Danielle","marsh",true,7,1111,99999999);  Passenger p4 = new Passenger("Danielle","marsh",true,7,1111,99999999);  Como clave el nombre | Debe mostrar un “name=Danielle, name=Danielle, name=Danielle, name=Valeria, name=pipe,” |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** Probar el correcto funcionamiento de remove de HashTable | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado esperado** |
| HashTable | remove | setupStage2 | h.remove("Danielle"); | Debería luego imprimir "name=Valeria, name=pipe, " |
| HashTable | remove | setupStage2 | h.remove("Cole");  h.remove("pipe"); h.remove("Danielle");  h.remove("Valeria"); | Debe retornar “”; |
| HashTable | remove | setupStage1 | h.remove("Cole"); | “”; |

\* Una prueba se compone de un conjunto de casos de prueba.

\* Cada fila representa un ***caso de prueba*** difente

\* En el objetivo de la prueba debe escribir una descripción sobre qué es lo que específicamente está probando del modelo del programa.

\* La clase es la clase del modelo que está siendo puesto a prueba.

\* El método es específicamente el método de la clase que está siendo puesto a prueba.

\* El escenario se refiere al nombre del escenario que usted definió. Todos los casos de prueba corresponden a escenarios.

\* Los valores de entrada son valores que entran al método puesto a prueba.

\* El resultado esperado es lo que se espera que suceda luego de ejecutar el método.