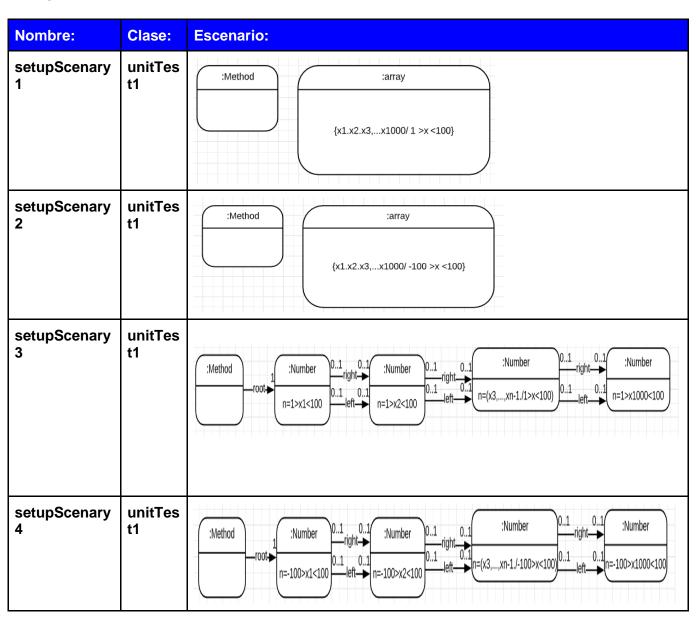
Configuración de escenarios:



Objetivo de la prueba: Verificar que el algoritmo mergesort esté implementado correctamente ordenando un arreglo de 100 enteros cuyo valor es mayor igual a 1 y menor igual a 100.

Clase:	Método:	Valores de entrada:	Resultado:
Methods	MergeSortTest1	$array = (x_1, x_2, x_3, \dots, x_{100} / \forall_x (1 > X < 100)))$	array=(X1 <x2<x3<x100))< td=""></x2<x3<x100))<>

Objetivo de la prueba: Verificar que el algoritmo mergesort esté implementado correctamente ordenando un arreglo de 100 enteros cuyo valor es mayor igual a -100 y menor igual a 100.

Clase:	Método:	Valores de entrada:	Resultado:
Methods	MergeSortTest2	$array = (x_1, x_2, x_3, \dots, x_{100} / \forall_x (-100 > X < 100)))$	array=(X1 <x2<x3<x100))< td=""></x2<x3<x100))<>

Objetivo de la prueba: Verificar que el algoritmo binary Tree Sort esté implementado correctamente ordenando un arreglo de 100 enteros cuyo valor es mayor igual a 1 y menor igual a 100.

Clase:	Método:	Valores de entrada:	Resultado:
Methods	BinaryTreeSortTest1	Se agregan 1000 Objetos tipo Number los cuales tendrán un atributo entero asignado mediante un método random desde 1 hasta 100, los cuales serán ordenados a medida que son agregados en el árbol.	Una List tipo Number la cual cumplira la que $X1 donde \forall_x(X \text{ es tipo Number})$

Objetivo de la prueba:						
Clase:	Método:	Valores de entrada:	Resultado:			
Methods	BinaryTreeSortTest2	Se agregan 1000 Objetos tipo Number los cuales tendrán un atributo entero asignado mediante un método random desde -100 hasta 100, los cuales serán ordenados a medida que son agregados en el árbol.	Una List tipo Number la cual cumplira la que $X1< X2< X3< X1000$ donde $\forall_x (X \text{ es tipo Number})$			