

Requerimientos funcionales

Nombre	RF#1 Mostrar la información de los vuelos
Resumen	Mostrar la información de un vuelo correspondiente a: hora de salida, aerolínea a la que pertenece el vuelo, código del vuelo, destino y puerta de embarque.
Entradas	Ninguna
Salidas	Información de los vuelos

Nombre	RF#2 Generar un listado aleatorio de vuelos
Resumen	Se debe poder generar un aleatorio de vuelos con información aleatoria. Por defecto el programa muestra un listado aleatorio y se debe poder generar uno nuevo cada vez que el usuario lo pida.
Entradas	Número de vuelos
Salidas	Listado nuevo de vuelos

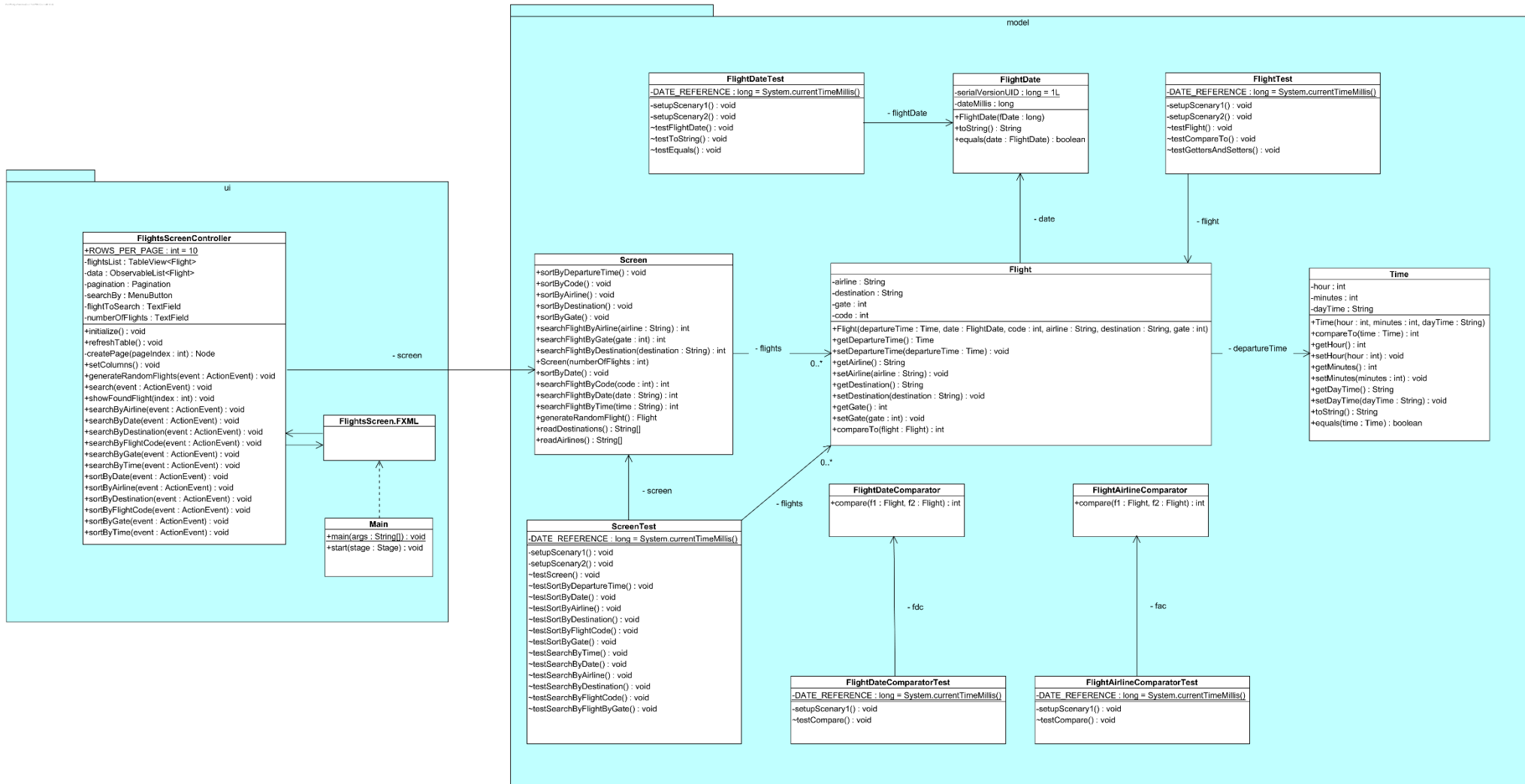
Nombre	RF#3 Buscar un vuelo
Resumen	Se debe poder buscar un vuelo por un criterio escogido por el usuario, ya sea por fecha, por hora de salida, por número de vuelo, por ciudad de destino y por puerta de embarque. Si hay más de un vuelo que concuerda con el criterio buscado, entonces se muestra el primer vuelo encontrado.
Entradas	Se define según cada criterio de búsqueda.
Salidas	Si la entrada para cada criterio de vuelo concuerda con al menos un vuelo, se muestra el primer vuelo encontrado, de lo contrario se informa al usuario que no hay un vuelo correspondiente a la entrada y criterio de búsqueda escogidos.

Nombre	RF#4 Ordenar los vuelos
Resumen	Se debe poder ordenar los vuelos por los diferentes criterios (hora de salida, nombre de aerolínea, código de vuelo, destino y puerta de desembarque), el criterio debe ser escogido por el usuario. Por defecto el programa ordena los vuelos por hora de salida.
Entradas	Criterio de ordenamiento
Salidas	Vuelos ordenados

Requerimientos no funcionales

Nombre	RNF#1 Uso de JavaFX para la interfaz gráfica.
Resumen	Se hace uso de las clases de JavaFX y el software de SceneBuilder para construir la interfaz gráfica que se muestra al usuario.

Diagrama de clases



Trazabilidad

Requerimiento	Método(s)	Clase(s)
Mostrar la información de los vuelos	setColumns createPage refreshTable initialize	FlightsScreenController
Generar un listado aleatorio de vuelos	generateRandomFlights generateRandomFlight	FlightsScreenController Screen
Buscar un vuelo	search searchBy* searchBy*	FlightsScreenController Screen
Ordenar los vuelos	sortBy* sortBy*	FlightsScreenController Screen

Escenarios

Ilustración 1

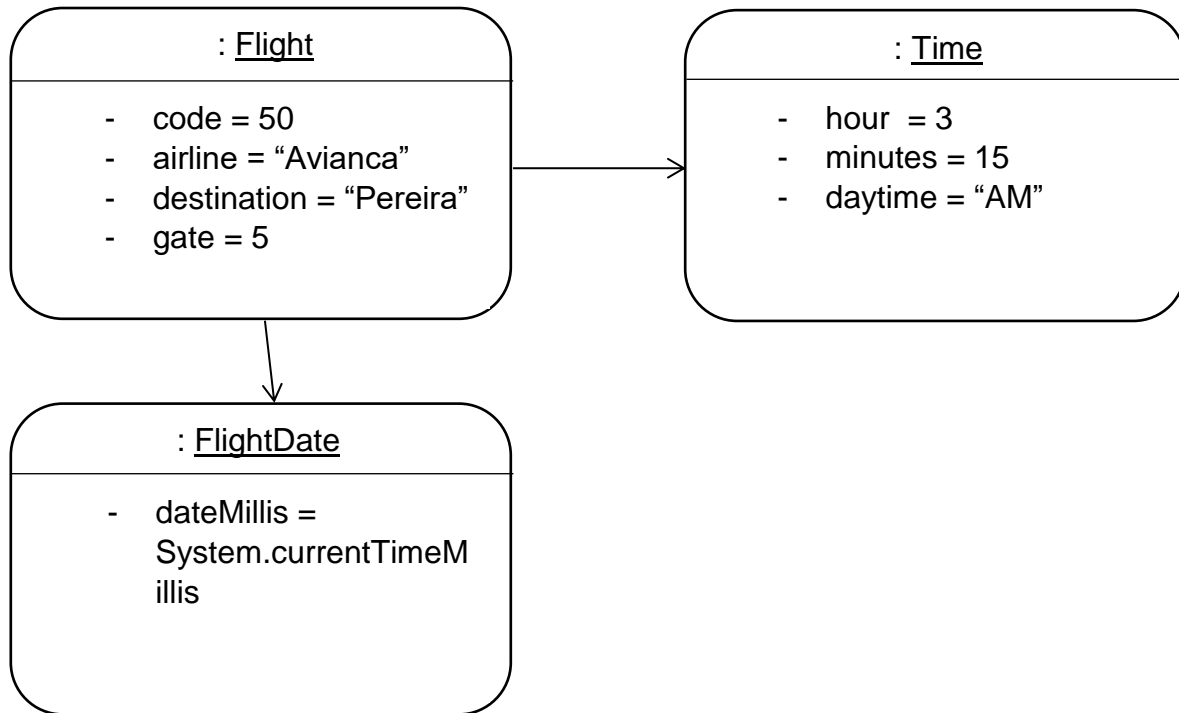


Ilustración 2

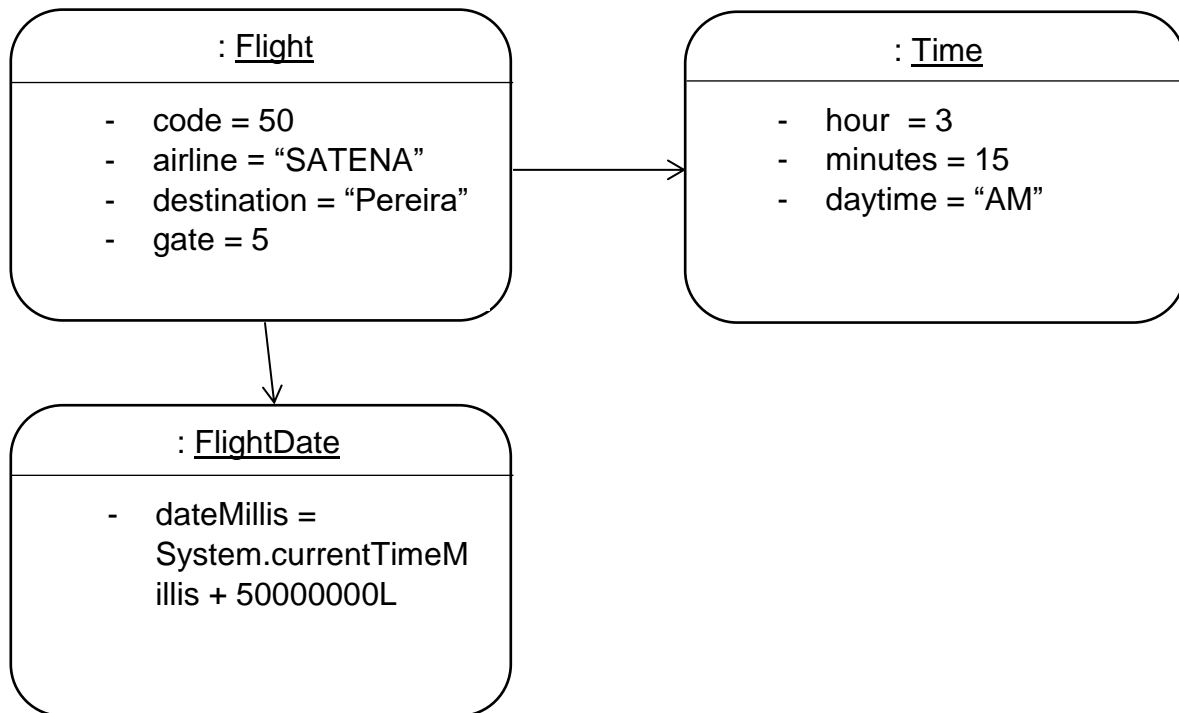
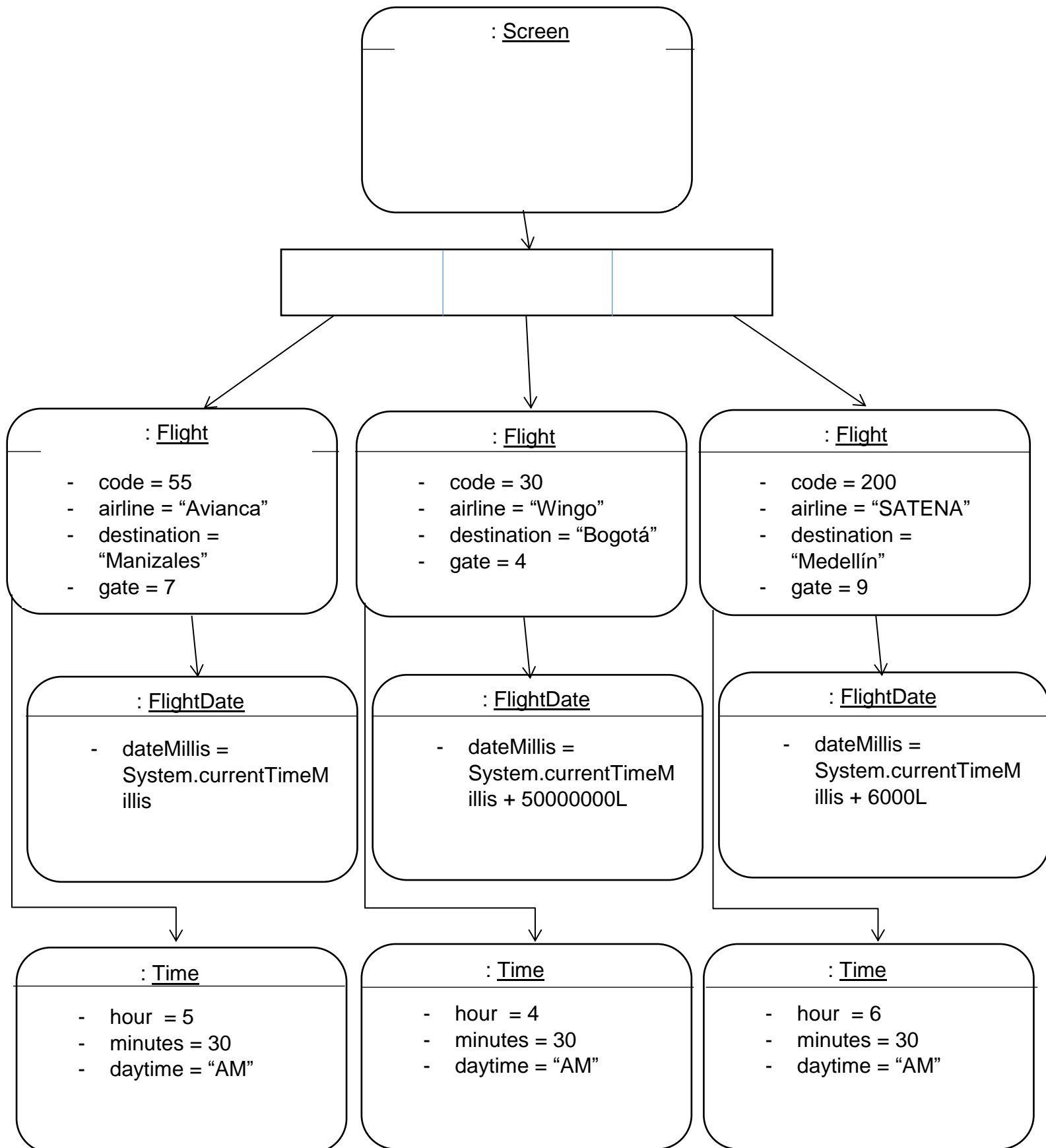



Ilustración 3




Nombre	Clase	Escenario
setupScenary1	Flight	vacío
setupScenary2	Flight	Ilustración 1
setupScenary1	Screen	vacío
setupScenary2	Screen	Ilustración 3

Pruebas

Objetivo de la prueba: Probar que el constructor asigna los valores correctamente a los atributos				
Clase	Método	Escenario	Valores de entrada	Resultado
Flight	Flight	setupScenary1	Objeto Time (3,15,"AM") , Objeto FlightDate (System.currentMillis()), 50, "Avianca", "Pereira", 5	Los atributos se inicializan correctamente.

Objetivo de la prueba: Probar que el método compareTo hace bien el trabajo de comparar a la instancia actual con el objeto tipo Flight que se pasa como parámetro, con el criterio por defecto que es el de departure time (hora de salida).				
Clase	Método	Escenario	Valores de entrada	Resultado
Flight	compareTo	setupScenary2	3 objetos tipo flight con los siguientes atributos 	Los objetos se comparan correctamente.

Objetivo de la prueba: Probar que los métodos get devuelven los valores de los atributos correctamente y los métodos set modifican los valores de sus respectivos atributos.				
Clase	Método	Escenario	Valores de entrada	Resultado
Flight	get* set*	setupScen ary2	Se modifican los valores de los atributos de la instancia de Flight de la siguiente manera: 	Los métodos get devuelven los valores de los atributos que han sido previamente modificados con los métodos set.

Objetivo de la prueba: Probar que el constructor asigna correctamente los valores pasados como parámetros a los atributos de la instancia.				
Clase	Método	Escenario	Valores de entrada	Resultado
Screen	Screen	setupScen ary1	Número de vuelos a crear: 3	La instancia de la clase Flight se crea correctamente y sin errores, se crean aleatoriamente 3 vuelos y se agregan al arreglo de vuelos que está como atributo de la clase.

Objetivo de la prueba: Probar que los métodos de búsqueda devuelven el índice del arreglo en el que se encuentra el primer objeto tipo Flight que corresponde al criterio de búsqueda.				
Clase	Método	Escenario	Valores de entrada	Resultado
Screen	searchByTime	setupScen ary2	"5:30 AM"	El método devuelve un índice que referencia al objeto que corresponde al primer objeto encontrado que corresponde al criterio de hora de salida.

Screen	searchByDate	setupScenario2	"Apr 12 2019"	El método devuelve un índice que referencia al objeto que corresponde al primer objeto encontrado que corresponde al criterio de fecha del vuelo.
Screen	searchByAirline	setupScenario2	"Avianca"	El método devuelve un índice que referencia al objeto que corresponde al primer objeto encontrado que corresponde al criterio de aerolínea.
Screen	searchByDestination	setupScenario2	"Medellín"	El método devuelve un índice que referencia al objeto que corresponde al primer objeto encontrado que corresponde al criterio de ciudad de destino.
Screen	searchByCode	setupScenario2	30	El método devuelve un índice que referencia al objeto que corresponde al primer objeto encontrado que corresponde al criterio de código de vuelo.
Screen	searchByGate	setupScenario2	7	El método devuelve un índice que referencia al objeto que corresponde al primer objeto encontrado que corresponde al criterio de puerta de embarque.

Objetivo de la prueba: Probar que los métodos de ordenamiento sitúan a cada objeto del arreglo de la instancia de la clase Screen en el lugar adecuado según sus respectivos criterios.				
Clase	Método	Escenario	Valores de entrada	Resultado
Screen	sortByDepartureTime	setupScenario2	Ninguno	El método ordena de manera correcta y ascendente a cada objeto del arreglo de vuelos de acuerdo con el criterio de hora de salida.
Screen	sortByDate	setupScenario2	Ninguno	El método ordena de manera correcta y ascendente a cada objeto del arreglo de vuelos de acuerdo con el criterio de fecha de vuelo.
Screen	sortByAirline	setupScenario2	Ninguno	El método ordena de manera correcta y ascendente a cada objeto del arreglo de vuelos de acuerdo con el criterio de aerolínea.
Screen	sortByDestination	setupScenario2	Ninguno	El método ordena de manera correcta y ascendente a cada objeto del arreglo de vuelos de acuerdo con el criterio de ciudad de destino.
Screen	sortByCode	setupScenario2	Ninguno	El método ordena de manera correcta y ascendente a cada objeto del arreglo de vuelos de acuerdo con el criterio de código de vuelo.
Screen	sortByGate	setupScenario2	Ninguno	El método ordena de manera correcta y ascendente a cada objeto del arreglo de vuelos de acuerdo con el criterio de puerta de embarque.

Objetivo de la prueba: Probar que los métodos “compare” de las clases comparadoras retornan un valor adecuado para cada caso.				
Clase	Método	Escenario	Valores de entrada	Resultado
FlightAirlineComparator	compare	setupScenario1	Ilustración 1 Ilustración 2	El método retorna un valor menor que 0 ya que al compararse por una cadena de texto “Avianca” representa un valor menor que “SATENA”
FlightDateComparator	compare	setupScenario1	Ilustración 1 Ilustración 2	El método retorna un valor menor que 0 ya que se compara por una fecha entonces la fecha del objeto de la Ilustración 1 debe ser menor que la del objeto de la Ilustración 2