ESTRUCTURA DE DATOS Y ALGORITMOS OBLIGATORIO

Veronica Busiello
212712

Grupo N3B
Nombre del docente: Michel Camarotta
Fecha: 23/11/2017

Table of Contents

1. Interfaz del Sistema: Pre y Post condiciones

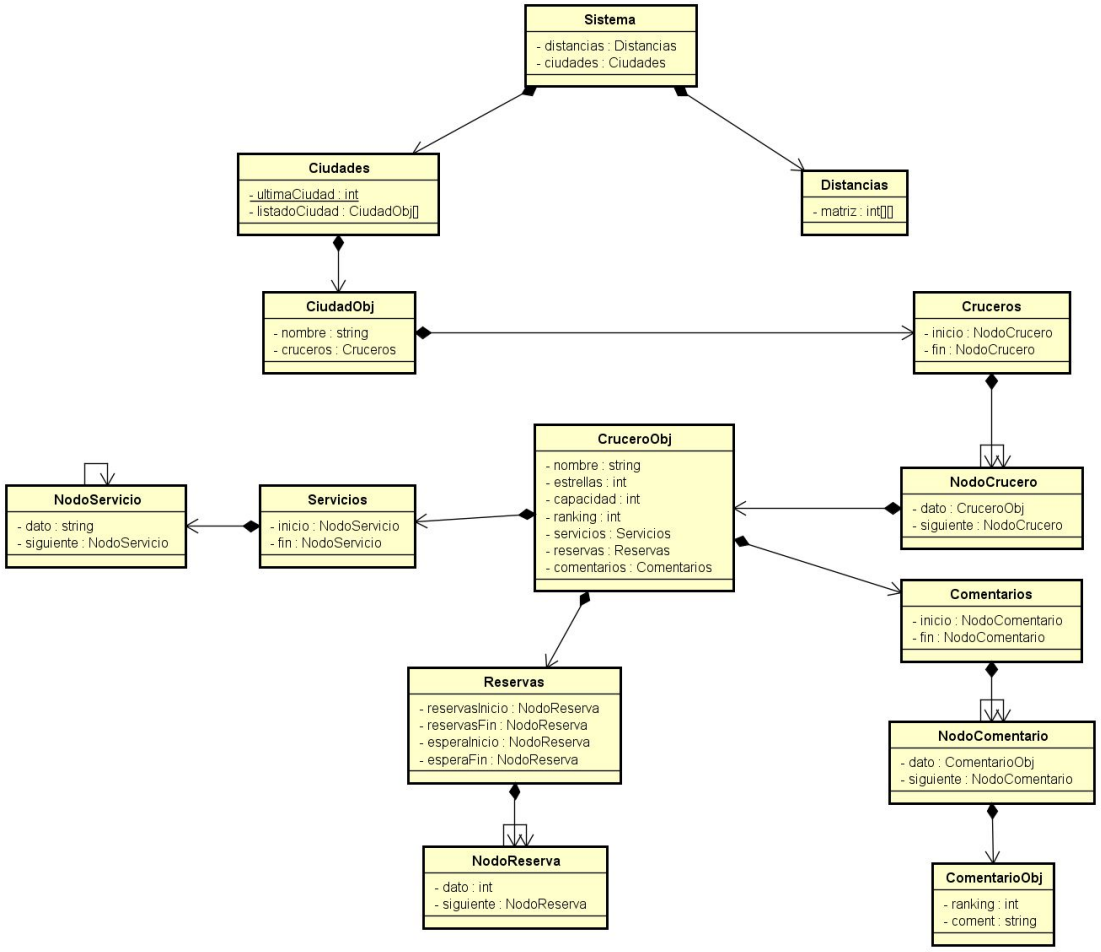
```
//PRE: CantCiudades es mayor a 0
//POS: Crea la estructura necesaria para representar el sistema. Si ya existe un sistema
//         de reservas, lo destruye previo a la creacion
Retorno crearSistemaReservas(int cantCiudades);

//PRE:
//POS: Destruye el sistema de reservas y todos sus elementos
Retorno destruirSistemaReservas();

//PRE: Ciudad no existe aun en el sistema
//POS: Registra la ciudad "ciudad"
Retorno registrarCiudad(String ciudad);

//PRE: Estrellas esta entre 1 y 5
//         Capacidad es mayor a 0
//         No existe ningun crucero "nombre"
//         Existe la ciudad "ciudad"
//POS: Registra el crucero de nombre "nombre" en la ciudad "ciudad" con cantidad de
//         estrellas "estrellas", capacidad "capacidad" y ranking 0
Retorno registrarCrucero(String ciudad, String nombre, int estrellas,int capacidad);

//PRE: Existe la ciudad "ciudad"
//         Existe el crucero "crucero" de la ciudad "ciudad"
//POS: Ingresa el servicio "servicio" al crucero "crucero"
Retorno ingresarServicio(String ciudad, String crucero, String servicio);

//PRE: Existe el crucero "crucero" de la ciudad "ciudad"
//         Existe el servicio "servicio" registrado en el crucero "crucero"
//         Existe la ciudad "ciudad"
//POS: Borra la primera occurrencia del servicio "servicio" del crucero "crucero"
Retorno borrarServicio(String ciudad, String crucero, String servicio);

//PRE: Existe el crucero "crucero" dentro de la ciudad "ciudad"
//         Existe la ciudad "ciudad"
//POS: Realiza la reserva de una habitacion en el crucero "crucero" dentro de la ciudad
//         "ciudad". En caso de que no haya cupo de habitaciones disponibles, el cliente
//         "cliente" quedara en lista de espera
Retorno realizarReserva(int cliente, String ciudad, String crucero);

//PRE: Existe un crucero "crucero" en la ciudad "ciudad"
//         El cliente tiene al menos una reserva en el crucero "crucero" de la ciudad
//                  "ciudad"
//         Existe la ciudad "ciudad"
//POS: Cancela la reserva en el crucero "crucero" en caso de que la cancelacion sea
//         exitosa y haya clientes en lista de espera, el cupo sera ocupado por el primer
//         cliente de la lista. En caso de que el cliente "cliente" tenga mas de una reserva
//         para el crucero "crucero" se cancelara solo la primera reserva en orden (primero
//         se buscara reservas de habitacion y luego en lista de espera.
Retorno cancelarReserva(int cliente, String ciudad, String crucero);

//PRE: Existe un crucero "crucero" en la ciudad "ciudad"
//         El ranking es mayor a 0 y menor a 5
```

```
//        Existe la ciudad "ciudad"
//POS: Ingresa el comentario "comentario" al crucero "crucero". El ranking general del
//        crucero quedara definido por el promedio de todas sus calificaciones.
//        Ademas, ordena los cruceros por ranking ascendente
Retorno ingresarComentario(String ciudad, String crucero, String comentario, int
ranking);

//PRE: Existe un crucero "crucero" en la ciudad "ciudad"
//        Existe la ciudad "ciudad"
//POS: lista todos los servicios prestados por el crucero "crucero" de la ciudad
//        "ciudad". En caso de que no haya servicios registrados en el crucero, se
//        imprimira: "No existen servicios registrados en el crucero <crucero><ciudad>"
Retorno listarServicios(String ciudad, String crucero);

//PRE: Existe la ciudad "ciudad"
//POS: lista todos los cruceros "crucero" de la ciudad "ciudad" ordenados por nombre.
//        En caso de que no haya cruceros registrados en la ciudad, se imprimira:
//        "No existen cruceros registrados en <ciudad>"
Retorno listarCrucerosCiudad(String ciudad);

//PRE: Existe la ciudad "ciudad"
//POS: lista todos los cruceros "crucero" de la ciudad "ciudad" ordenados por ranking
//        ascendente. En caso de que no haya cruceros registrados en la ciudad, se
//        imprimira: "No existen cruceros registrados en el sistema"
Retorno listarCrucerosRankingAsc(String ciudad);

//PRE: Existe la ciudad "ciudad"
//POS: lista todos los cruceros "crucero" de la ciudad "ciudad" ordenados por ranking
//        descendente. En caso de que no haya cruceros registrados en la ciudad, se
//        imprimira: "No existen cruceros registrados en el sistema"
Retorno listarCrucerosRankingDesc(String ciudad);

//PRE:
//POS:Lista todos los cruceros registrados en el sistemas ordenados por ranking
//        descendente
Retorno listarCrucerosRanking();

//PRE: Existe un crucero "crucero" en la ciudad "ciudad"
//        Existe la ciudad "ciudad"
//POS: Lista todos los comentarios ingresados para el Crucero "crucero" dentro de la
//        ciudad "ciudad". En caso que no existan comentarios para el crucero "crucero"
//        el sistema imprimira:
//        "No se han agregado comentarios al crucero <crucero> <ciudad>"
Retorno listarComentarios(String ciudad, String crucero);

//PRE: La matriz "ciudades" es del mismo tamaño que la cantidad de ciudades en
//        sistema
//POS: Carga las distancias en la matriz de distancias del sistema
Retorno cargarDistancias(int[][] ciudades);

//PRE: La matriz "m" es del mismo tamaño que la cantidad de ciudades en sistema
//        Existe la ciudad "origen"
//        Existe la ciudad "destino"
//        Hay un camino entre "origen" y "destino" con solo una ciudad intermedia
```

```
//POS: Carga las distancias en "m" y devuelve el camino mas corto para llegar del
//          origen a destino restringiendo la busqueda a caminos que solo tengan una
//          ciudad intermedia
Retorno buscarCamino(int [][] m, String origen, String destino);
```

2. Solución escogida

2.1 Diagrama de la estructura de datos



Nota: El diagrama debe mostrar claramente el contenido de cada una de las estructuras y como se relacionan entre ellas.

2.2 Justificación

Sistema	
Estructura	private Ciudades ciudades; private Distancias distancias;
Justificación	- Ciudades contiene un array de ciudades. - Distancias contiene una matriz con distancias entre las ciudades. - Para identificar una ciudad en la matriz Distancias, basta con buscar la ciudad en el indice deseado. Los indices de las distancias y del array, coinciden. - Se implemento de esta manera para que haya reducir los tiempos de ejecución al buscar cruceros en una ciudad.

Distancias	
Estructura	private int[][] matriz;
Justificación	- Para identificar una ciudad en la matriz Distancias, basta con buscar la ciudad en el indice deseado. Los indices de las distancias y del array, coinciden. - Se implemento de esta manera para poder realizar la parte 2 del obligatorio

Ciudades	
Estructura	private static int ultimaCiudad; CiudadObj[] listadoCiudad;
Justificación	- La variable estatica es para poder ingresar una ciudad en el primer lugar disponible del arreglo - El listadoCiudad contiene las ciudades con sus elementos

CiudadObj	
Estructura	String nombre; Cruceros cruceros;
Justificación	- Se guarda una lista de cruceros que existen en esa ciudad - Se implemento de esta manera para que haya reducir los tiempos de ejecución al buscar cruceros en una ciudad.

Cruceros	
Estructura	NodoCrucero inicio; NodoCrucero fin;
Justificación	- Hay un puntero al principio de la lista y otro al final - Esto se implemento asi para tener mas control sobre la estructura

NodoCrucero	
Estructura	private CruceroObj dato; private NodoCrucero siguiente;
Justificación	Nodo con dato y un puntero al siguiente objeto

CruceroObj	
Estructura	String nombre; int estrellas; int capacidad; int ranking; Servicios servicios; Reservas reservas; Comentarios comentarios;
Justificación	Cada crucero cuenta con su lista de reservas, servicios y comentarios, ademas de sus atributos basicos

Servicios	
Estructura	NodoServicio inicio; NodoServicio fin;
Justificación	- Hay un puntero al principio de la lista y otro al final - Esto se implemento asi para tener mas control sobre la estructura

NodoServicio	
Estructura	private String dato; private NodoServicio siguiente;
Justificación	Nodo con dato del tipo string y un puntero al siguiente objeto

Reservas	
Estructura	NodoReserva reservasInicio; NodoReserva reservasFin; NodoReserva esperaInicio; NodoReserva esperaFin;
Justificación	- La estructura cuenta con 2 listas, una para almacenar las reservas de habitaciones y la otra para almacenar la lista de espera - Hay un puntero al principio de la lista y otro al final para cada lista - Esto se implemento asi para tener mas control sobre la estructura y sus listas

NodoReserva	
Estructura	private int dato; private NodoReserva siguiente;
Justificación	Nodo con dato del tipo int y un puntero al siguiente objeto

Comentarios	
Estructura	NodoComentario inicio; NodoComentario fin;
Justificación	- Hay un puntero al principio de la lista y otro al final - Esto se implemento asi para tener mas control sobre la estructura

NodoComentario	
Estructura	ComentarioObj dato; NodoComentario siguiente;
Justificación	Nodo con dato y un puntero al siguiente objeto

ComentarioObj	
Estructura	int ranking; String coment;
Justificación	Cada comentario cuenta con su ranking y su comentario escrito

3. Pruebas Ejecutadas

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package obli;

import org.junit.Before;
import org.junit.Test;
import static org.junit.Assert.*;

/**
 *
 * @author veronicab-ot
 */
public class SistemaTest {

        private Sistema sis;
        private Retorno r;

        @Before
        public void setUp() throws Exception {
       sis = new Sistema();
        }

        @Test
        public void testCrearSistemaReservasMenorQueCero() {
      assertEquals(Retorno.Resultado.ERROR_1,
sis.crearSistemaReservas(-15).resultado);
        }

        @Test
        public void testCrearSistemaReservasOkIgualACero() {
      assertEquals(Retorno.Resultado.OK,
sis.crearSistemaReservas(0).resultado);
        }

        @Test
        public void testCrearSistemaReservasOkMayorACero() {
      assertEquals(Retorno.Resultado.OK,
sis.crearSistemaReservas(6).resultado);
        }

        @Test
        public void testCrearSistemaReservasOkYaExiste() {
```

```
        sis.crearSistemaReservas(3);
        assertEquals(Retorno.Resultado.OK,
sis.crearSistemaReservas(6).resultado);
         }

         @Test
         public void testDestruirSistemaReservasMenosQueCeroCiudades() {
        sis = new Sistema();
        sis.crearSistemaReservas(-5);
        assertEquals(Retorno.Resultado.OK,
sis.destruirSistemaReservas().resultado);
         }

         @Test
         public void testDestruirSistemaReservasOk() {
        sis = new Sistema();
        sis.crearSistemaReservas(5);
        assertEquals(Retorno.Resultado.OK,
sis.destruirSistemaReservas().resultado);
         }

         @Test
         public void testRegistrarCiudadOk() {
        sis.crearSistemaReservas(4);
        assertEquals(Retorno.Resultado.OK,
sis.registrarCiudad("Seattle").resultado);
         }

     @Test
         public void testRegistrarCiudadNoQuedaLugar() {
        sis.crearSistemaReservas(2);

        assertEquals(Retorno.Resultado.OK,
sis.registrarCiudad("Valparaiso").resultado);
        assertEquals(Retorno.Resultado.OK,
sis.registrarCiudad("Montevideo").resultado);
        assertEquals(Retorno.Resultado.ERROR_2,
sis.registrarCiudad("Paris").resultado);
         }

         @Test
         public void testRegistrarCiudadExistente() {
        sis.crearSistemaReservas(4);

        assertEquals(Retorno.Resultado.OK,
sis.registrarCiudad("Valparaiso").resultado);
        assertEquals(Retorno.Resultado.ERROR_1,
sis.registrarCiudad("Valparaiso").resultado);
```

```
        }

        @Test
        public void testRegistrarCruceroMenosEstrellas() {
        sis.crearSistemaReservas(4);

        sis.registrarCiudad("Montevideo");

        assertEquals(Retorno.Resultado.ERROR_1,
sis.registrarCrucero("Montevideo", "Donkey Kong", 0, 100).resultado);
        }

        @Test
        public void testRegistrarCruceroMasEstrellas() {
        sis.crearSistemaReservas(4);

        sis.registrarCiudad("Montevideo");

        assertEquals(Retorno.Resultado.ERROR_1,
sis.registrarCrucero("Montevideo", "Donkey Kong", 6, 100).resultado);
        }

        @Test
        public void testRegistrarCruceroMenosCapacidad() {
        sis.crearSistemaReservas(4);

        sis.registrarCiudad("Paris");

        assertEquals(Retorno.Resultado.ERROR_2, sis.registrarCrucero("Paris",
"Titanic", 5, -1).resultado);
        }

        @Test
        public void testRegistrarCruceroNoExisteCiudad() {
        sis.crearSistemaReservas(4);

        sis.registrarCiudad("Paris");

        assertEquals(Retorno.Resultado.ERROR_4,
sis.registrarCrucero("Maldonado", "Capitan Miranda", 5, 100).resultado);
        }

        @Test
        public void testRegistrarCruceroOk() {
        sis.crearSistemaReservas(4);

        sis.registrarCiudad("Auckland");
```

```
        assertEquals(Retorno.Resultado.OK, sis.registrarCrucero("Auckland",
"Libertad", 5, 100).resultado);
         }

         @Test
         public void testRegistrarCruceroExistente() {
                  sis.crearSistemaReservas(4);

        sis.registrarCiudad("Cuenca ");
        sis.registrarCrucero("Cuenca ", "Jose", 5, 100);

        assertEquals(Retorno.Resultado.ERROR_3, sis.registrarCrucero("Cuenca ",
"Jose", 2, 120).resultado);
         }

         @Test
         public void testIngresarServicioNoExisteCiudad() {
        sis.crearSistemaReservas(4);

        assertEquals(Retorno.Resultado.ERROR_2,
sis.ingresarServicio("Tarariras", "Crucerito", "Casino").resultado);
         }

         @Test
         public void testIngresarServicioNoExisteCrucero() {
        sis.crearSistemaReservas(4);

        sis.registrarCiudad("Montevideo");
        assertEquals(Retorno.Resultado.ERROR_1,
sis.ingresarServicio("Montevideo", "Felipe II", "Pesca").resultado);
         }

         @Test
         public void testIngresarServicioOk() {
        sis.crearSistemaReservas(4);

        sis.registrarCiudad("Punta del este");
        sis.registrarCrucero("Punta del este", "Francisco Papa", 5, 100);

        assertEquals(Retorno.Resultado.OK, sis.ingresarServicio("Punta del este",
"Francisco Papa", "Teatro").resultado);
         }

         @Test
         public void testIngresarServicioRepetido() {
        sis.crearSistemaReservas(4);

        sis.registrarCiudad("Punta del este");
```

```
        sis.registrarCrucero("Punta del este", "Francisco Papa", 5, 100);
        sis.ingresarServicio("Punta del este", "Francisco Papa", "Teatro");

        assertEquals(Retorno.Resultado.OK, sis.ingresarServicio("Punta del este",
"Francisco Papa", "Teatro").resultado);
         }

         @Test
         public void testBorrarServicioNoExisteCiudad() {
        sis.crearSistemaReservas(4);

        assertEquals(Retorno.Resultado.ERROR_3, sis.borrarServicio("Pando",
"German", "Origami").resultado);
         }

         @Test
         public void testBorrarServicioNoExisteCrucero() {
        sis.crearSistemaReservas(4);

        sis.registrarCiudad("Madrid");
        assertEquals(Retorno.Resultado.ERROR_1, sis.borrarServicio("Madrid",
"Sonrisa", "Origami").resultado);
         }

         @Test
         public void testBorrarServicioNoExisteServicio() {
        sis.crearSistemaReservas(4);

        sis.registrarCiudad("Madrid");
        sis.registrarCrucero("Madrid", "Sonrisa", 5, 100);
        assertEquals(Retorno.Resultado.ERROR_2, sis.borrarServicio("Madrid",
"Sonrisa", "Origami").resultado);
         }

         @Test
         public void testBorrarServicioOk() {
        sis.crearSistemaReservas(4);

        sis.registrarCiudad("Madrid");
        sis.registrarCrucero("Madrid", "Crucerito", 5, 100);
        sis.ingresarServicio("Madrid", "Crucerito", "Origami");
        assertEquals(Retorno.Resultado.OK, sis.borrarServicio("Madrid",
"Crucerito", "Origami").resultado);
         }

         @Test
         public void testRealizarReservaNoExisteCiudad() {
        sis.crearSistemaReservas(4);
```

```
        assertEquals(Retorno.Resultado.ERROR_2, sis.realizarReserva(1,"Pando",
"Crucerito").resultado);
         }

         @Test
         public void testRealizarReservaNoExisteCrucero() {
        sis.crearSistemaReservas(4);

        sis.registrarCiudad("Rocha");
        assertEquals(Retorno.Resultado.ERROR_1, sis.realizarReserva(1,"Rocha",
"Rocanova").resultado);
         }

         @Test
         public void testRealizarReservaOk() {
        sis.crearSistemaReservas(4);

        sis.registrarCiudad("Buenos Aires");
        sis.registrarCrucero("Buenos Aires", "Libre", 5, 2);
        assertEquals(Retorno.Resultado.OK, sis.realizarReserva(1,"Buenos Aires",
"Libre").resultado);
         }

         @Test
         public void testRealizarReservaEspera() {
        sis.crearSistemaReservas(4);

        sis.registrarCiudad("Valparaiso");
        sis.registrarCrucero("Valparaiso", "Crucero", 5, 2);
        assertEquals(Retorno.Resultado.OK, sis.realizarReserva(1,"Valparaiso",
"Crucero").resultado);
        assertEquals(Retorno.Resultado.OK, sis.realizarReserva(2,"Valparaiso",
"Crucero").resultado);
        assertEquals(Retorno.Resultado.OK, sis.realizarReserva(3,"Valparaiso",
"Crucero").resultado);
         }

         @Test
         public void testCancelarReservaNoExisteCiudad() {
        sis.crearSistemaReservas(4);

        assertEquals(Retorno.Resultado.ERROR_3, sis.cancelarReserva(1,"Pando",
"Cru1").resultado);
         }

         @Test
         public void testCancelarReservaNoExisteCrucero() {
```

```
        sis.crearSistemaReservas(4);

        sis.registrarCiudad("Buenos aires");

        assertEquals(Retorno.Resultado.ERROR_1, sis.cancelarReserva(1,"Buenos
Aires", "MSC Fantasia").resultado);
         }

         @Test
         public void testCancelarReservaNoExisteReserva() {
        sis.crearSistemaReservas(4);

        sis.registrarCiudad("Londres");
        sis.registrarCrucero("Londres", "MSC Opera", 5, 2);
        assertEquals(Retorno.Resultado.ERROR_2,
sis.cancelarReserva(1,"Londres", "MSC Opera").resultado);
         }

         @Test
         public void testCancelarReservaOk() {
        sis.crearSistemaReservas(4);

        sis.registrarCiudad("Londres");
        sis.registrarCrucero("Londres", "MSC Opera", 5, 2);
        sis.realizarReserva(1,"Londres", "MSC Opera");
        assertEquals(Retorno.Resultado.OK, sis.cancelarReserva(1,"Londres",
"MSC Opera").resultado);
         }

         @Test
         public void testCancelarReservaYaCancelada() {
        sis.crearSistemaReservas(4);

        sis.registrarCiudad("Londres");
        sis.registrarCrucero("Londres", "MSC Opera", 5, 2);
        sis.realizarReserva(1,"Londres", "Cru1");
        sis.cancelarReserva(1,"Londres", "MSC Opera");

        assertEquals(Retorno.Resultado.ERROR_2,
sis.cancelarReserva(1,"Londres", "MSC Opera").resultado);
         }

         @Test
         public void testIngresarComentarioMenosPuntos() {
        sis.crearSistemaReservas(4);

        assertEquals(Retorno.Resultado.ERROR_1,
sis.ingresarComentario("Tarariras", "MSC Fantasia", "Espantoso!",-1).resultado);
```

```
        }

        @Test
        public void testIngresarComentarioMasPuntos() {
        sis.crearSistemaReservas(4);

        assertEquals(Retorno.Resultado.ERROR_1,
sis.ingresarComentario("Tarariras", "MSC Fantasia", "Espantoso!",6).resultado);
        }

        @Test
        public void testIngresarComentarioNoExisteCiudad() {
        sis.crearSistemaReservas(4);

        assertEquals(Retorno.Resultado.ERROR_3,
sis.ingresarComentario("Montevideo", "MSC Fantasia",
"Espantoso!",1).resultado);
        }

        @Test
        public void testIngresarComentarioNoExisteCrucero() {
        sis.crearSistemaReservas(4);

        sis.registrarCiudad("Montevideo");

        assertEquals(Retorno.Resultado.ERROR_2,
sis.ingresarComentario("Montevideo", "MSC Fantasia",
"Espantoso!",1).resultado);
        }

        @Test
        public void testIngresarComentarioOk() {
        sis.crearSistemaReservas(4);

        sis.registrarCiudad("Montevideo");
        sis.registrarCrucero("Montevideo", "MSC Fantasia", 5, 2);

        assertEquals(Retorno.Resultado.OK,
sis.ingresarComentario("Montevideo", "MSC Fantasia",
"Espantoso!",1).resultado);
        }

        @Test
        public void testListarServiciosNoExisteCiudad() {
        sis.crearSistemaReservas(4);

        assertEquals(Retorno.Resultado.ERROR_2, sis.listarServicios("Tarariras",
"MSC Opera").resultado);
```

```
        }

        @Test
        public void testListarServiciosNoExisteCrucero() {
        sis.crearSistemaReservas(4);

        sis.registrarCiudad("Montevideo");

        assertEquals(Retorno.Resultado.ERROR_1,
sis.listarServicios("Montevideo", "MSC Opera").resultado);
        }

        @Test
        public void testListarServiciosOk() {
        sis.crearSistemaReservas(4);

        sis.registrarCiudad("Montevideo");
        sis.registrarCiudad("Valparaiso");
        sis.registrarCrucero("Montevideo", "MSC Opera", 2, 22);
        sis.registrarCrucero("Montevideo", "MSC Fantasia", 5, 2);
        sis.registrarCrucero("Valparaiso", "MSC Fantasia", 3, 4);
        sis.ingresarServicio("Montevideo", "MSC Opera", "Malabares");
        sis.ingresarServicio("Montevideo", "MSC Opera", "Orquesta");
        sis.ingresarServicio("Montevideo", "MSC Opera", "Mago");
        sis.ingresarServicio("Montevideo", "MSC Opera", "Animador");
        sis.ingresarServicio("Montevideo", "MSC Opera", "Casino");
        sis.ingresarServicio("Montevideo", "MSC Fantasia", "Casino");

        r = sis.listarServicios("Montevideo", "MSC Opera");
        assertEquals(Retorno.Resultado.OK, r.resultado);
        assertTrue(r.valorString.indexOf("Malabares") <
r.valorString.indexOf("Orquesta"));
        assertTrue(r.valorString.indexOf("Orquesta") <
r.valorString.indexOf("Mago"));
        assertTrue(r.valorString.indexOf("Mago") <
r.valorString.indexOf("Animador"));
        assertTrue(r.valorString.indexOf("Animador") <
r.valorString.indexOf("Casino"));
        }

        @Test
        public void testListarCrucerosCiudadNoExisteCiudad() {

        sis.crearSistemaReservas(4);

        assertEquals(Retorno.Resultado.ERROR_1,
sis.listarCrucerosCiudad("Pando").resultado);
        }
```

```
 @Test
 public void testListarCrucerosCiudadNoHayCruceros() {

sis.crearSistemaReservas(4);
sis.registrarCiudad("Montevideo");

r = sis.listarCrucerosCiudad("Montevideo");
assertEquals(Retorno.Resultado.OK, r.resultado);

String strR = r.valorString;
assertEquals("No existen cruceros registrados en Montevideo", strR);
 }

 @Test
 public void testListarCrucerosCiudadOk() {

sis.crearSistemaReservas(4);

sis.registrarCiudad("Montevideo");
sis.registrarCrucero("Montevideo", "Cru1", 5, 2);
sis.ingresarComentario("Montevideo", "Cru1", "Espantoso!",1);
sis.registrarCrucero("Montevideo", "Cru4", 5, 2);
sis.ingresarComentario("Montevideo", "Cru4", "Maomeno!",3);
sis.registrarCrucero("Montevideo", "Cru2", 5, 2);
sis.ingresarComentario("Montevideo", "Cru2", "Biennn!",4);
sis.ingresarComentario("Montevideo", "Cru2", "Espantoso!",2);
sis.registrarCrucero("Montevideo", "Cru3", 5, 2);
sis.ingresarComentario("Montevideo", "Cru3", "Esselllente!!",5);

r = sis.listarCrucerosCiudad("Montevideo");
assertEquals(Retorno.Resultado.OK, r.resultado);

String strR = r.valorString;
assertTrue(strR.indexOf("Montevideo")>-1);
assertTrue(strR.indexOf("Cru1")<strR.indexOf("5",strR.indexOf("Cru1"))
&&

strR.indexOf("5",strR.indexOf("Cru1"))<strR.indexOf("1",strR.indexOf("5",strR.in
dexOf("Cru1"))) &&
            strR.indexOf("1",strR.indexOf("5",strR.indexOf("Cru1"))) <
strR.indexOf("Cru2"));
assertTrue(strR.indexOf("Cru2")<strR.indexOf("5",strR.indexOf("Cru2"))
&&

strR.indexOf("5",strR.indexOf("Cru2"))<strR.indexOf("3",strR.indexOf("5",strR.in
dexOf("Cru2"))) &&
```

```
                strR.indexOf("3",strR.indexOf("5",strR.indexOf("Cru2"))) <
strR.indexOf("Cru3"));
        assertTrue(strR.indexOf("Cru3")<strR.indexOf("5",strR.indexOf("Cru3"))
&&

strR.indexOf("5",strR.indexOf("Cru3"))<strR.indexOf("5",strR.indexOf("5",strR.in
dexOf("Cru3"))+1) &&
                strR.indexOf("5",strR.indexOf("5",strR.indexOf("Cru3"))+1) <
strR.indexOf("Cru4"));
        assertTrue(strR.indexOf("Cru4")<strR.indexOf("5",strR.indexOf("Cru4"))
&&

strR.indexOf("5",strR.indexOf("Cru4"))<strR.indexOf("3",strR.indexOf("5",strR.in
dexOf("Cru4"))));

         }

         @Test
         public void testListarCrucerosRankingAscNoExisteCiudad() {
        sis.crearSistemaReservas(4);

        assertEquals(Retorno.Resultado.ERROR_1,
sis.listarCrucerosRankingAsc("Pando").resultado);
         }

         @Test
         public void testListarCrucerosRankingAscNoHayCruceros() {
        sis.crearSistemaReservas(4);
        assertEquals(Retorno.Resultado.ERROR_1,
sis.listarCrucerosRankingAsc("Tarariras").resultado);
        sis.registrarCiudad("Montevideo");

        r = sis.listarCrucerosRankingAsc("Montevideo");
        assertEquals(Retorno.Resultado.OK, r.resultado);

        String strR = r.valorString;
        assertEquals("No existen cruceros registrados en el sistema", strR);
         }

         @Test
         public void testListarCrucerosRankingAscOk() {
        sis.crearSistemaReservas(4);
        assertEquals(Retorno.Resultado.ERROR_1,
sis.listarCrucerosRankingAsc("Tarariras").resultado);
        sis.registrarCiudad("Montevideo");
        sis.registrarCrucero("Montevideo", "Cru2", 5, 2);
        sis.ingresarComentario("Montevideo", "Cru2", "Biennn!",4);
        sis.registrarCrucero("Montevideo", "Cru3", 5, 2);
```

```
        sis.ingresarComentario("Montevideo", "Cru3", "Exelente!",5);
        sis.registrarCrucero("Montevideo", "Cru1", 5, 2);
        sis.ingresarComentario("Montevideo", "Cru1", "Espantoso!",1);
        sis.registrarCrucero("Montevideo", "Cru5", 5, 2);
        sis.ingresarComentario("Montevideo", "Cru5", "Un asco",1);
        sis.ingresarComentario("Montevideo", "Cru5", "Meh!!",3);
        sis.registrarCrucero("Montevideo", "Cru4", 5, 2);
        sis.ingresarComentario("Montevideo", "Cru4", "Maomeno!",3);

        r = sis.listarCrucerosRankingAsc("Montevideo");
        assertEquals(Retorno.Resultado.OK, r.resultado);

        String strR = r.valorString;
        assertTrue(strR.indexOf("Montevideo")>-1);
        assertTrue(strR.indexOf("Cru1")<strR.indexOf("1",strR.indexOf("Cru1"))
&&
                strR.indexOf("1",strR.indexOf("Cru1"))<strR.indexOf("Cru4"));
        assertTrue(strR.indexOf("Cru4")<strR.indexOf("3",strR.indexOf("Cru4"))
&&
                strR.indexOf("3",strR.indexOf("Cru4"))<strR.indexOf("Cru2"));
        assertTrue(strR.indexOf("Cru2")<strR.indexOf("4",strR.indexOf("Cru2"))
&&
                strR.indexOf("4",strR.indexOf("Cru2"))<strR.indexOf("Cru3"));
        assertTrue(strR.indexOf("Cru5")<strR.indexOf("2",strR.indexOf("Cru5"))
&&
                strR.indexOf("2",strR.indexOf("Cru5"))<strR.indexOf("Cru4"));
        assertTrue(strR.indexOf("Cru3")<strR.indexOf("5",strR.indexOf("Cru3")));
         }

         @Test
         public void testListarCrucerosRankingDescNoExisteCiudad() {
        sis.crearSistemaReservas(4);
        assertEquals(Retorno.Resultado.ERROR_1,
sis.listarCrucerosRankingAsc("Tarariras").resultado);
    }

         @Test
         public void testListarCrucerosRankingDescNoHayCruceros() {
        sis.crearSistemaReservas(4);
        sis.registrarCiudad("Montevideo");

        r = sis.listarCrucerosRankingDesc("Montevideo");
        assertEquals(Retorno.Resultado.OK, r.resultado);

        String strR = r.valorString;
        assertEquals("No existen cruceros registrados en el sistema", strR);
         }
```

```
 @Test
 public void testListarCrucerosRankingDescOk() {
sis.crearSistemaReservas(4);
sis.registrarCiudad("Montevideo");
sis.registrarCrucero("Montevideo", "Cru2", 5, 2);
sis.ingresarComentario("Montevideo", "Cru2", "Biennn!",4);
sis.registrarCrucero("Montevideo", "Cru1", 5, 2);
sis.ingresarComentario("Montevideo", "Cru1", "Espantoso!",1);
sis.registrarCrucero("Montevideo", "Cru3", 5, 2);
sis.ingresarComentario("Montevideo", "Cru3", "Esselllente!!",5);
sis.registrarCrucero("Montevideo", "Cru5", 5, 2);
sis.ingresarComentario("Montevideo", "Cru5", "Un asco",1);
sis.ingresarComentario("Montevideo", "Cru5", "Meh!!",3);
sis.registrarCrucero("Montevideo", "Cru4", 5, 2);
sis.ingresarComentario("Montevideo", "Cru4", "Maomeno!",3);

r = sis.listarCrucerosRankingDesc("Montevideo");
assertEquals(Retorno.Resultado.OK, r.resultado);

String strR = r.valorString;
assertTrue(strR.indexOf("Montevideo")>-1);
assertTrue(strR.indexOf("Cru3")<strR.indexOf("5",strR.indexOf("Cru3"))
&&
        strR.indexOf("5",strR.indexOf("Cru3"))<strR.indexOf("Cru2"));
assertTrue(strR.indexOf("Cru2")<strR.indexOf("4",strR.indexOf("Cru2"))
&&
        strR.indexOf("4",strR.indexOf("Cru2"))<strR.indexOf("Cru4"));
assertTrue(strR.indexOf("Cru4")<strR.indexOf("3",strR.indexOf("Cru4"))
&&
        strR.indexOf("3",strR.indexOf("Cru4"))<strR.indexOf("Cru1"));
assertTrue(strR.indexOf("Cru5")<strR.indexOf("2",strR.indexOf("Cru5"))
&&
        strR.indexOf("2",strR.indexOf("Cru5"))<strR.indexOf("Cru1"));
assertTrue(strR.indexOf("Cru1")<strR.indexOf("1",strR.indexOf("Cru1")));
 }

 @Test
 public void testListarCrucerosRankingNoHayCruceros() {

sis.crearSistemaReservas(4);
sis.registrarCiudad("Montevideo");
sis.registrarCiudad("Melo");
sis.registrarCiudad("Trinidad");

r = sis.listarCrucerosRanking();
assertEquals(Retorno.Resultado.OK, r.resultado);

String strR = r.valorString;
```

```
        assertEquals("No hay registros de cruceros en el sistema", strR);
         }

         @Test
         public void testListarCrucerosRankingOk() {
        sis.crearSistemaReservas(4);
        sis.registrarCiudad("Montevideo");
        sis.registrarCiudad("Melo");
        sis.registrarCiudad("Trinidad");
        sis.registrarCrucero("Montevideo", "Cru2", 5, 2);
        sis.ingresarComentario("Montevideo", "Cru2", "Biennn!",4);
        sis.registrarCrucero("Melo", "Cru1", 5, 2);
        sis.ingresarComentario("Melo", "Cru1", "Espantoso!",1);
        sis.registrarCrucero("Trinidad", "Cru3", 5, 2);
        sis.ingresarComentario("Trinidad", "Cru3", "Esselllente!!",5);
        sis.registrarCrucero("Melo", "Cru4", 5, 2);
        sis.ingresarComentario("Melo", "Cru4", "Maomeno!",3);
        sis.registrarCrucero("Montevideo", "Cru5", 5, 2);
        sis.ingresarComentario("Montevideo", "Cru5", "Horrible!",1);
        sis.ingresarComentario("Montevideo", "Cru5", "Meh",3);

        r = sis.listarCrucerosRanking();
        assertEquals(Retorno.Resultado.OK, r.resultado);

        String strR = r.valorString;
        assertTrue(strR.indexOf("Cru3")<strR.indexOf("5",strR.indexOf("Cru3"))
&&
                strR.indexOf("5",strR.indexOf("Cru3"))<strR.indexOf("Cru2"));
        assertTrue(strR.indexOf("Cru2")<strR.indexOf("4",strR.indexOf("Cru2"))
&&
                strR.indexOf("4",strR.indexOf("Cru2"))<strR.indexOf("Cru4"));
        assertTrue(strR.indexOf("Cru4")<strR.indexOf("3",strR.indexOf("Cru4"))
&&
                strR.indexOf("3",strR.indexOf("Cru4"))<strR.indexOf("Cru1"));
        assertTrue(strR.indexOf("Cru5")<strR.indexOf("2",strR.indexOf("Cru5"))
&&
                strR.indexOf("2",strR.indexOf("Cru5"))<strR.indexOf("Cru1"));
        assertTrue(strR.indexOf("Cru1")<strR.indexOf("1",strR.indexOf("Cru1")));
         }

         @Test
         public void testListarComentariosNoExisteCiudad() {
        sis.crearSistemaReservas(4);
        assertEquals(Retorno.Resultado.ERROR_2,
sis.listarComentarios("Montevideo", "Cru2").resultado);
         }

         @Test
```

```
    public void testListarComentariosNoExisteCrucero() {
        sis.crearSistemaReservas(4);
        sis.registrarCiudad("Montevideo");
        assertEquals(Retorno.Resultado.ERROR_1,
sis.listarComentarios("Montevideo", "Cru2").resultado);
    }

    @Test
    public void testListarComentarios() {
        sis.crearSistemaReservas(4);
        sis.registrarCiudad("Montevideo");
        sis.registrarCrucero("Montevideo", "Cru2", 5, 2);
        sis.ingresarComentario("Montevideo", "Cru2", "Maomeno!",3);
        sis.ingresarComentario("Montevideo", "Cru2", "Esselllente!!",5);
        sis.ingresarComentario("Montevideo", "Cru2", "Genial!!",5);
        sis.ingresarComentario("Montevideo", "Cru2", "Espantoso!",1);
        sis.ingresarComentario("Montevideo", "Cru2", "Biennn!",4);
        sis.ingresarComentario("Montevideo", "Cru2", "Se pasa lindo!",4);

        r = sis.listarComentarios("Montevideo", "Cru2");
        assertEquals(Retorno.Resultado.OK, r.resultado);

        System.out.println(r.valorString);

        String strR = r.valorString;
        assertTrue(strR.indexOf("Biennn")<strR.indexOf("Espantoso"));
        assertTrue(strR.indexOf("Espantoso")<strR.indexOf("Esselllente"));
        assertTrue(strR.indexOf("Esselllente")<strR.indexOf("Maomeno"));
    }
    @Test
    public void testListarComentariosNoHayComentarios() {
        sis.crearSistemaReservas(4);
        sis.registrarCiudad("Montevideo");
        sis.registrarCrucero("Montevideo", "Cru2", 5, 2);

        r = sis.listarComentarios("Montevideo", "Cru2");
        assertEquals(Retorno.Resultado.OK, r.resultado);

        System.out.println(r.valorString);

        String strR = r.valorString;
        assertEquals("No se han agregado comentarios al crucero Cru2
Montevideo", strR);
    }

    @Test
    public void testCargarDistanciasOk() {
        sis.crearSistemaReservas(6);
```

```
        sis.registrarCiudad("Montevideo");
        sis.registrarCiudad("Santiago");
        sis.registrarCiudad("Lima");
        sis.registrarCiudad("San Pablo");
        sis.registrarCiudad("Panama");
        sis.registrarCiudad("New York");

        assertEquals(Retorno.Resultado.OK,sis.cargarDistancias(new int[][] {
              {0,10,25,15,30,0},
              {10,0,20,0,0,0},
              {25,20,0,0,0,40},
              {15,0,0,0,0,45},
              {30,0,0,0,0,25},
              {0,0,40,45,25,0}
        }).resultado);
         }

         @Test
         public void testCargarDistanciasArrayDistintoTam() {
        sis.crearSistemaReservas(4);
        sis.registrarCiudad("Montevideo");
        sis.registrarCiudad("Santiago");
        sis.registrarCiudad("Lima");
        sis.registrarCiudad("San Pablo");

        assertEquals(Retorno.Resultado.ERROR_1,sis.cargarDistancias(new int[][]
{
              {0,10,25,15,30,0},
              {10,0,20,0,0,0},
              {25,20,0,0,0,40},
              {15,0,0,0,0,45},
              {30,0,0,0,0,25},
              {0,0,40,45,25,0}
        }).resultado);
         }

         @Test
         public void testBuscarCaminoDistintoTam() {

                sis.crearSistemaReservas(5);
                sis.registrarCiudad("Montevideo");
                sis.registrarCiudad("Santiago");
                sis.registrarCiudad("Lima");
                sis.registrarCiudad("San Pablo");
                sis.registrarCiudad("Panama");

                int[][] mat = new int[][] {
                        {0,10,25,15,30,0},
```

```
                {10,0,20,0,0,0},
                {25,20,0,0,0,40},
                {15,0,0,0,0,45},
                {30,0,0,0,0,25},
                {0,0,40,45,25,0}
        };

  r = sis.buscarCamino(mat,"Montevideo","New York");
        assertEquals(Retorno.Resultado.ERROR_1,r.resultado);
}

@Test
public void testBuscarCaminoNoExisteCamino() {

        sis.crearSistemaReservas(6);
        sis.registrarCiudad("Montevideo");
        sis.registrarCiudad("Santiago");
        sis.registrarCiudad("Lima");
        sis.registrarCiudad("San Pablo");
        sis.registrarCiudad("Panama");
        sis.registrarCiudad("New York");

        int[][] mat = new int[][] {
                {0,0,0,0,0,0},
                {0,0,0,0,0,0},
                {0,0,0,0,0,0},
                {0,0,0,0,0,0},
                {0,0,0,0,0,2},
                {0,0,0,0,0,0}
        };

  r = sis.buscarCamino(mat,"Montevideo","New York");
        assertEquals(Retorno.Resultado.ERROR_4,r.resultado);
}

@Test
public void testBuscarCaminoNoExisteOrigen() {

        sis.crearSistemaReservas(6);
        sis.registrarCiudad("Montevideo");
        sis.registrarCiudad("Santiago");
        sis.registrarCiudad("Lima");
        sis.registrarCiudad("San Pablo");
        sis.registrarCiudad("Panama");
        sis.registrarCiudad("New York");

        int[][] mat = new int[][] {
                {0,0,0,0,0,0},
```

```
                {0,0,0,0,0,0},
                {0,0,0,0,0,0},
                {0,0,0,0,0,0},
                {0,0,0,0,0,2},
                {0,0,0,0,0,0}
        };

  r = sis.buscarCamino(mat,"Pando","New York");
        assertEquals(Retorno.Resultado.ERROR_2,r.resultado);
}

@Test
public void testBuscarCaminoNoExisteDestino() {

        sis.crearSistemaReservas(6);
        sis.registrarCiudad("Montevideo");
        sis.registrarCiudad("Santiago");
        sis.registrarCiudad("Lima");
        sis.registrarCiudad("San Pablo");
        sis.registrarCiudad("Panama");
        sis.registrarCiudad("New York");

        int[][] mat = new int[][] {
                {0,0,0,0,0,0},
                {0,0,0,0,0,0},
                {0,0,0,0,0,0},
                {0,0,0,0,0,0},
                {0,0,0,0,0,2},
                {0,0,0,0,0,0}
        };

  r = sis.buscarCamino(mat,"Montevideo","Guadalajara");
        assertEquals(Retorno.Resultado.ERROR_3,r.resultado);
}

@Test
public void testBuscarCaminoOk() {

        sis.crearSistemaReservas(6);
        sis.registrarCiudad("Montevideo");
        sis.registrarCiudad("Santiago");
        sis.registrarCiudad("Lima");
        sis.registrarCiudad("San Pablo");
        sis.registrarCiudad("Panama");
        sis.registrarCiudad("New York");

        int[][] mat = new int[][] {
                {0,10,25,15,30,0},
```

```
                {10,0,20,0,0,0},
                {25,20,0,0,0,40},
                {15,0,0,0,0,45},
                {30,0,0,0,0,25},
                {0,0,40,45,25,0}
            };

        r = sis.buscarCamino(mat,"Montevideo","New York");
            assertEquals(Retorno.Resultado.OK,r.resultado);

            String strR = r.valorString;
            assertTrue(strR.indexOf("Montevideo")<strR.indexOf("Panama"));
            assertTrue(strR.indexOf("Panama")<strR.indexOf("New York"));
        }
}
```