Evaluación de aprendizaje 3 – Programación básica 2 – Ivan Kek

```
package ar.edu.unlam.pb2.ea3;


import java.util.Comparator;

import java.util.HashSet;

import java.util.Iterator;

import java.util.Set;

import java.util.TreeSet;


public class EquipoDeFutbol {

	private String nombre;

	private Set<Jugador> jugadores;


	public EquipoDeFutbol(String nombre) {

		this.nombre = nombre;

		this.jugadores = new TreeSet<Jugador>();

	}


	/*

	 * La capacidad máxima de un equipo es 23. Cualquier intento de agregar más

	 * jugadores generará una excepción (CapacidadMaximaException). Además, no

	 * deberá permitir duplicar Jugadores (JugadorDuplicadoException).

	 */

	public void agregarJugador(Jugador jugador) throws Exception {


		if (buscarJugador(jugador) != null) {
```

```
                throw new JugadorDuplicadoException("No se pueden agregar dos jugadores iguales a la lista");

            }

            if (jugadores.size() < 23) {
                jugadores.add(jugador);
            } else {
                throw new CapacidadMaximaException("No se pueden agregar más de 23 jugadores a la lista");
            }

        }

        public Jugador buscarJugador(Jugador jugador) {
            Jugador jugadorEncontrado = null;
            for (Jugador jugadorLista : jugadores) {
                if (jugadorLista.equals(jugador)) {

                    jugadorEncontrado = jugador;
                }

            }
            return jugadorEncontrado;
        }

        /*
```

```
	 * Permite cambiar cualquier jugador. Un intento de cambiar un jugador no
	 * presente en el equipo generará una excepción (JugadoreInexistenteException).
	 */
	public Boolean cambiarJugador(Jugador saliente, Jugador entrante) throws JugadorInexistenteException {

		Jugador jugadorSaliente = buscarJugador(saliente);
		Boolean seCambio = false;
		if (buscarJugador(saliente) != null) {
			jugadores.remove(saliente);
			jugadores.add(entrante);
			seCambio = true;
		} else {
			throw new JugadorInexistenteException("No se encontró jugador para cambiar");

		}

		return seCambio;
	}

	public Integer getCantidadJugadores() {
		return jugadores.size();
	}

	public Set<Jugador> getJugadores() {
		return jugadores;
	}
```

```
	public void setJugadores(Set<Jugador> jugadores) {
		this.jugadores = jugadores;
	}

	public TreeSet<Jugador> devolverPlanteOrdenadoPorNombreDeJugador() {
		ordenPorNombre ordenPorNombre = new ordenPorNombre();
		TreeSet<Jugador> devolverPorOrdenApellidoYNombre = new
TreeSet<>(ordenPorNombre);
		devolverPorOrdenApellidoYNombre.addAll(jugadores);
		return devolverPorOrdenApellidoYNombre;
	}

	public TreeSet<Jugador> devolverPlanteOrdenadoPorPrecioDeCompraDeJugador() {
		ordenPorPrecioCompra ordenPorPrecio = new ordenPorPrecioCompra();
		TreeSet<Jugador> devolverPorPrecioDeCompra = new
TreeSet<>(ordenPorPrecio);
		devolverPorPrecioDeCompra.addAll(jugadores);
		return devolverPorPrecioDeCompra;
	}

	public TreeSet<Jugador> devolverPlanteOrdenadoPorNumeroDeCamisetaDeJugador()
{
		ordenPorNroCamiseta ordenPorCamiseta = new ordenPorNroCamiseta();
		TreeSet<Jugador> devolverPorNroCamiseta = new
TreeSet<>(ordenPorCamiseta);
		devolverPorNroCamiseta.addAll(jugadores);
		return devolverPorNroCamiseta;
	}
```

```
	private TreeSet<Jugador> ordenarELPlantelParaDevolver(Comparator
criterioDeOrdenacion) {

		TreeSet<Jugador> equipoOrdenado = new
TreeSet<Jugador>(criterioDeOrdenacion);

		equipoOrdenado.addAll(jugadores);

		return equipoOrdenado;

	}

}
```

```
package ar.edu.unlam.pb2.ea3;


public class Jugador implements Comparable<Jugador> {


	private Integer numero;

	private String nombre;

	private String apellido;

	private Integer precio;


	public Jugador(Integer numero, String nombre, String apellido, Integer precio) {

		this.numero = numero;

		this.nombre = nombre;

		this.apellido = apellido;

		this.precio = precio;

	}


	public Integer getNumero() {

		return numero;
```

```
}

public void setNumero(Integer numero) {
	this.numero = numero;
}

public String getNombre() {
	return nombre;
}

public void setNombre(String nombre) {
	this.nombre = nombre;
}

public String getApellido() {
	return apellido;
}

public void setApellido(String apellido) {
	this.apellido = apellido;
}

public Integer getPrecio() {
	return precio;
}

public void setPrecio(Integer precio) {
```

```
		this.precio = precio;
}

@Override
public int hashCode() {
		final int prime = 31;
		int result = 1;
		result = prime * result + ((apellido == null) ? 0 : apellido.hashCode());
		result = prime * result + ((nombre == null) ? 0 : nombre.hashCode());
		return result;
}

@Override
public boolean equals(Object obj) {
		if (this == obj)
				return true;
		if (obj == null)
				return false;
		if (getClass() != obj.getClass())
				return false;
		Jugador other = (Jugador) obj;
		if (apellido == null) {
				if (other.apellido != null)
						return false;
		} else if (!apellido.equals(other.apellido))
				return false;
		if (nombre == null) {
```

```
                if (other.nombre != null)
                        return false;
            } else if (!nombre.equals(other.nombre))
                return false;
            return true;
        }

        @Override
        public int compareTo(Jugador o) {
            if (this.apellido.compareTo(o.getApellido()) == 0)
                return this.nombre.compareTo(o.getNombre());
            return this.apellido.compareTo(o.getApellido());
        }
}

package ar.edu.unlam.pb2.ea3;

public class CapacidadMaximaException extends Exception {

        public CapacidadMaximaException(String mensaje) {
            super(mensaje);
        }

}

package ar.edu.unlam.pb2.ea3;
```

```
public class JugadorDuplicadoException extends Exception {

	public JugadorDuplicadoException(String mensaje) {
		super(mensaje);
	}

}
```

```
package ar.edu.unlam.pb2.ea3;

public class JugadorInexistenteException extends Exception {

	public JugadorInexistenteException(String mensaje) {
		super(mensaje);
	}

}
```

```
package ar.edu.unlam.pb2.ea3;

import java.util.Comparator;

public class ordenPorNombre implements Comparator<Jugador> {

	@Override
	public int compare(Jugador jugador1, Jugador jugador2) {
```

```
                if (jugador1.getApellido().compareTo(jugador2.getApellido()) == 0)
                        return jugador1.getNombre().compareTo(jugador2.getNombre());
                        return jugador1.getApellido().compareTo(jugador2.getApellido());
        }

}
```

```
package ar.edu.unlam.pb2.ea3;

import java.util.Comparator;

public class ordenPorNroCamiseta implements Comparator<Jugador> {

        @Override
        public int compare(Jugador jugador1, Jugador jugador2) {

                return jugador1.getNumero().compareTo(jugador2.getNumero());

        }

}
```

```
package ar.edu.unlam.pb2.ea3;

import java.util.Comparator;

public class ordenPorPrecioCompra implements Comparator<Jugador> {
```

```
	@Override
	public int compare(Jugador jugador1, Jugador jugador2) {

		return jugador1.getPrecio().compareTo(jugador2.getPrecio());

	}

}

package ar.edu.unlam.pb2.ea3;

import static org.junit.Assert.*;

import java.util.Iterator;

import org.junit.Test;

public class EquipoDeFultbolTest {

	@Test
	public void queSePuedaCrearUnEquipoVacio() {
		EquipoDeFutbol estudiantes = new EquipoDeFutbol("Estudiantes De La Plata");
		assertEquals(0, estudiantes.getJugadores().size());
	}

	@Test
```

```
	public void queSePuedanAgregarJugadorAlEquipo() throws Exception {

		EquipoDeFutbol estudiantes = new EquipoDeFutbol("Estudiantes De La Plata");

		Jugador Veron = new Jugador(11, "Juan Sebastian", "Verón", 1000);


		estudiantes.agregarJugador(Veron);

		assertEquals(1, estudiantes.getCantidadJugadores(), 0);

	}


	@Test(expected = Exception.class)

	public void queAlIntentarAgregarUnJugadorExistenetLanceExcepcion() throws
Exception {


		EquipoDeFutbol estudiantes = new EquipoDeFutbol("Estudiantes De La Plata");

		Jugador Veron = new Jugador(11, "Juan Sebastian", "Verón", 1000);


		estudiantes.agregarJugador(Veron);

		estudiantes.agregarJugador(Veron);


	}


	@Test(expected = Exception.class)

	public void queAlIntentarAgregarCantidadExcesivaDeJugadoresLanceExcepcion()
throws Exception {


		EquipoDeFutbol estudiantes = new EquipoDeFutbol("Estudiantes De La Plata");

		Jugador Veron = new Jugador(11, "Juan Sebastian", "Verón", 10000);

		Jugador Lujambio = new Jugador(12, "Josemir", "Lujambio", 1000);

		Jugador Calderon = new Jugador(9, "Jose Luis", "Calderoón", 4000);
```

```
Jugador Delorte = new Jugador(19, "Alejandro", "Delorte", 2000);
Jugador Sosa = new Jugador(7, "Jose Ernesto", "Sosa", 6000);
Jugador Desabato = new Jugador(2, "Leandro", "Desabato", 6000);
Jugador Braña = new Jugador(22, "Rodrigo", "Braña", 8000);
Jugador Pavone = new Jugador(16, "Mariano", "Pavone", 5000);
Jugador Carrusca = new Jugador(23, "Marcelo", "Carrusca", 3000);
Jugador Luguercio = new Jugador(8, "Pablo Ariel", "Luguercio", 4000);
Jugador Gelabert = new Jugador(13, "Marcos", "Gelabert", 3000);
Jugador Andujar = new Jugador(1, "Mariano", "Andujar", 7000);
Jugador Damonte = new Jugador(5, "Israel", "Damonte", 3000);
Jugador Alayes = new Jugador(6, "Agustin", "Alayes", 5000);
Jugador Alvarez = new Jugador(3, "Pablo", "Alvarez", 2000);
Jugador Casierra = new Jugador(15, "Juan", "Casierra", 3000);
Jugador Boselli = new Jugador(17, "Mauro", "Boselli", 5000);
Jugador Carrillo = new Jugador(14, "Guido", "Carrillo", 4000);
Jugador Correa = new Jugador(21, "Joaquin", "Correa", 6000);
Jugador Perez = new Jugador(18, "Enzo", "Perez", 8000);
Jugador Fernandez = new Jugador(10, "Gaston", "Fernandez", 7000);
Jugador Angeleri = new Jugador(4, "Marcos", "Angeleri", 4000);
Jugador Cellay = new Jugador(20, "Christian", "Cellay", 3000);
Jugador Schunke = new Jugador(24, "Jonathan", "Schunke", 1000);

estudiantes.agregarJugador(Veron);
estudiantes.agregarJugador(Lujambio);
estudiantes.agregarJugador(Calderon);
estudiantes.agregarJugador(Delorte);
estudiantes.agregarJugador(Sosa);
```

```
        estudiantes.agregarJugador(Desabato);
        estudiantes.agregarJugador(Braña);
        estudiantes.agregarJugador(Pavone);
        estudiantes.agregarJugador(Carrusca);
        estudiantes.agregarJugador(Luguercio);
        estudiantes.agregarJugador(Gelabert);
        estudiantes.agregarJugador(Andujar);
        estudiantes.agregarJugador(Damonte);
        estudiantes.agregarJugador(Alayes);
        estudiantes.agregarJugador(Alvarez);
        estudiantes.agregarJugador(Casierra);
        estudiantes.agregarJugador(Boselli);
        estudiantes.agregarJugador(Carrillo);
        estudiantes.agregarJugador(Correa);
        estudiantes.agregarJugador(Perez);
        estudiantes.agregarJugador(Fernandez);
        estudiantes.agregarJugador(Angeleri);
        estudiantes.agregarJugador(Cellay);
        estudiantes.agregarJugador(Schunke);

}

@Test
public void queSePuedaCambiarUnJugador() throws Exception {

        EquipoDeFutbol estudiantes = new EquipoDeFutbol("Estudiantes De La Plata");
        Jugador Veron = new Jugador(11, "Juan Sebastian", "Verón", 10000);
```

```
        Jugador Lujambio = new Jugador(12, "Josemir", "Lujambio", 1000);
        Jugador Calderon = new Jugador(9, "Jose Luis", "Calderoón", 4000);
        Jugador Delorte = new Jugador(19, "Alejandro", "Delorte", 2000);

        estudiantes.agregarJugador(Veron);
        estudiantes.agregarJugador(Lujambio);
        estudiantes.agregarJugador(Calderon);
        estudiantes.agregarJugador(Delorte);

        Jugador Giunta = new Jugador(5, "Blas", "Giunta", 5000);

        assertTrue(estudiantes.cambiarJugador(Lujambio, Giunta));
        assertNull(estudiantes.buscarJugador(Lujambio));
        assertNotNull(estudiantes.buscarJugador(Giunta));

}

@Test(expected = Exception.class)
public void queAlCambiarUnJugadorInexistenteLanceExcepcion() throws Exception {

        EquipoDeFutbol estudiantes = new EquipoDeFutbol("Estudiantes De La Plata");
        Jugador Veron = new Jugador(11, "Juan Sebastian", "Verón", 10000);
        Jugador Lujambio = new Jugador(12, "Josemir", "Lujambio", 1000);
        Jugador Calderon = new Jugador(9, "Jose Luis", "Calderoón", 4000);
        Jugador Delorte = new Jugador(19, "Alejandro", "Delorte", 2000);

        estudiantes.agregarJugador(Veron);
```

```
		estudiantes.agregarJugador(Lujambio);
		estudiantes.agregarJugador(Calderon);
		estudiantes.agregarJugador(Delorte);

		assertEquals(4, estudiantes.getCantidadJugadores(), 0);

		Jugador jugadorInexistente = new Jugador(8, "Facundo", "Sanchez", 500);
		Jugador Giunta = new Jugador(5, "Blas", "Giunta", 5000);

		assertTrue(estudiantes.cambiarJugador(jugadorInexistente, Giunta));

	}

	@Test
	public void queElEquipoPresenteLosJugadoresOrdenadosPorNombre() throws
Exception {
		// Lo ordené por apellido y nombre :)

		EquipoDeFutbol estudiantes = new EquipoDeFutbol("Estudiantes De La Plata");
		ordenPorNombre ordenPorApellidoYNombre = new ordenPorNombre();
		Jugador Veron = new Jugador(11, "Juan Sebastian", "Verón", 10000);
		Jugador Alayes = new Jugador(6, "Agustin", "Alayes", 5000);
		Jugador Cellay = new Jugador(20, "Christian", "Cellay", 3000);

		estudiantes.agregarJugador(Veron);
		estudiantes.agregarJugador(Alayes);
		estudiantes.agregarJugador(Cellay);
```

```
        estudiantes.setJugadores(estudiantes.devolverPlanteOrdenadoPorNombreDeJugador()
);

                Iterator<Jugador> it = estudiantes.getJugadores().iterator();
                Integer i = 1;

                while (it.hasNext()) {
                        Jugador j = it.next();
                        switch (i) {
                        case 1:
                                assertEquals("Alayes", j.getApellido());
                                i++;
                                break;

                        case 2:
                                assertEquals("Cellay", j.getApellido());
                                i++;
                                break;

                        case 3:
                                assertEquals("Verón", j.getApellido());
                        }

                }

        }
```

```
	@Test
	public void queElEquipoPresenteLosJugadoresOrdenadosPorPrecioDeCompra() throws Exception {
		EquipoDeFutbol estudiantes = new EquipoDeFutbol("Estudiantes De La Plata");
		ordenPorPrecioCompra ordenPorPrecioCompra = new ar.edu.unlam.pb2.ea3.ordenPorPrecioCompra();
		Jugador Veron = new Jugador(11, "Juan Sebastian", "Verón", 10000);
		Jugador Alayes = new Jugador(6, "Agustin", "Alayes", 5000);
		Jugador Cellay = new Jugador(20, "Christian", "Cellay", 3000);

		estudiantes.agregarJugador(Veron);
		estudiantes.agregarJugador(Alayes);
		estudiantes.agregarJugador(Cellay);

	estudiantes.setJugadores(estudiantes.devolverPlanteOrdenadoPorPrecioDeCompraDeJugador());

		Iterator<Jugador> it = estudiantes.getJugadores().iterator();
		Integer i = 1;

		while (it.hasNext()) {
			Jugador j = it.next();
			switch (i) {
			case 1:
				assertEquals(3000, j.getPrecio(), 0.0);
				i++;
				break;
```

```
                case 2:
                        assertEquals(5000, j.getPrecio(), 0.0);
                        i++;
                        break;

                case 3:
                        assertEquals(10000, j.getPrecio(), 0.0);
                        i++;
                        break;
                }

        }

    }

    @Test
    public void queElEquipoPresenteLosJugadoresOrdenadosPorNumeroDeCamiseta()
throws Exception {

        EquipoDeFutbol estudiantes = new EquipoDeFutbol("Estudiantes De La Plata");
        ordenPorNroCamiseta ordenPorNroCamiseta = new ordenPorNroCamiseta();
        Jugador Veron = new Jugador(11, "Juan Sebastian", "Verón", 10000);
        Jugador Alayes = new Jugador(6, "Agustin", "Alayes", 5000);
        Jugador Cellay = new Jugador(20, "Christian", "Cellay", 3000);

        estudiantes.agregarJugador(Veron);
        estudiantes.agregarJugador(Alayes);
        estudiantes.agregarJugador(Cellay);
```

```
        estudiantes.setJugadores(estudiantes.devolverPlanteOrdenadoPorNumeroDeCamiseta
DeJugador());

                Iterator<Jugador> it = estudiantes.getJugadores().iterator();
                Integer i = 1;

                while (it.hasNext()) {
                        Jugador j = it.next();
                        switch (i) {
                        case 1:
                                assertEquals(6, j.getNumero(), 0.0);
                                i++;
                                break;

                        case 2:
                                assertEquals(11, j.getNumero(), 0.0);
                                i++;
                                break;

                        case 3:
                                assertEquals(20, j.getNumero(), 0.0);
                                i++;
                                break;
                        }

                }
```

```
	}
}
package ar.edu.unlam.pb2.ea3;

import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class JugadorTest {

	@Test
	public void queUnJugadorSeaIgualAOtroPorSuNombre() {
		Jugador Veron = new Jugador(11, "Juan Sebastian", "Verón", 10000);
		Jugador Veron1 = new Jugador(12, "Juan Sebastian", "Verón", 10000);

		assertEquals(Veron.getNombre(), Veron.getNombre());
	}
}
```