JUNIT

Elaboración de un documento y clase JUNIT

Elaborado por Veronica Gonzalez Bravo

Índice

ELIMINAR

```java
/**
 * Elimina un elemento concreto previamente habiendolo buscado, retorna un boolean para indicar si se encontro el
 * registro
 * @param idBuscar
 * @return boolean : encontrado
 */
public boolean eliminarElementos(String idBuscar) {

    boolean encontrado = false;

    for (int i = 0; i < c.length && encontrado == false; i++) {
        if (c[i].getId().compareTo(idBuscar) == 0) {
            eliminar(c[i],i);
            encontrado = true;
        }

    }

    for (int i = 0; i < e.length && encontrado == false; i++) {
        if (e[i].getId().compareTo(idBuscar) == 0) {
            eliminar(e[i],i);
            encontrado = true;
        }

    }

    return encontrado;
}
```

Clases de equivalencia

Condiciones Entrada	Clases equivalencia validas	Clases equivalencia no validas
idBuscar no se encuentra en ningun vector de objetos	Se encuentra la id(1)	no se encuentra la id(2)
Entrada debe ser String	Es String(3)	No es String (4)

Casos prueba

Base	Clase Equivalencia	Resultado
“abc”	1,3	false
“O2”	2,3	true
null	4	ERROR

```java
/**
 * Metodo que elimina un cliente previamente introducido y reajusta el vector
 * @param caEliminar
 */
public static boolean eliminarCliente(Cliente caEliminar) {
    boolean encontrado = true;
    int pos = localizarPosCliente(caEliminar);

    numClientes--;
    Cliente v[] = new Cliente [numClientes];
    System.out.println("ENCONTRADO: " + caEliminar.getId() + " sera eliminado");

    for (int i = 0 ; i < pos; i++) {
        v[i] = c[i];
    }
    for (int i = pos ; i < numClientes; i++) {
        v[i] = c[i+1];
    }

    c = v;
    return encontrado;
}
```

Clases de equivalencia

Condiciones Entrada	Clases equivalencia validas	Clases equivalencia no validas
La clase Cliente esta dentro del vector	Esta dentro del vector(1)	No esta dentro del vector (2)
La entrada debe ser String	Es String(3)	No es String(4)

Casos prueba

Base	Clase Equivalencia	Resultado
Cliente c1	1,3	True
Cliente c5	2,3	False
null	4	ERROR

BUSCAR

```java
/**
 * Metodo que muestra una clase Cliente a partir de una id introducida, devuelve un booleano indicando si ha sido
 * encontrada
 * @param id
 * @return
 */
public static boolean buscarClientes(String id) {
    boolean encontrado = false;


    for (int i = 0; i < c.length && encontrado == false; i++) {
        if (c[i].getId().compareTo(id)==0) {
            System.out.println("!!!!!!ENCONTRADO!!!!!");
            System.out.println(c[i].toString());
            encontrado = true;
        }
    }
    return encontrado;
}
```

Casos prueba

Base	Clase Equivalencia	Resultado
Id debe estar en el vector de objetos	id esta en el vector de objetos(1)	id no esta en el vector de objetos(2)
La entrada debe ser String	Es String(3)	No es String(4)

Casos prueba

Base	Clase Equivalencia	Resultado
“C2”	1,3	TRUE
“C4”	2,3	FALSE
null	4	ERROR

```java
/**
 * Metodo que busca elementos dentro de los vectores de Consignaciones y Empleados de la clase Consigna
 * @param id
 * @return
 */
public boolean buscarElementos(String id) {
    boolean encontrado = false;

    for (int i = 0; i < c.length && encontrado == false; i++) {
        if (c[i].getId().compareTo(id) == 0) {
            System.out.println("!!!!!!ENCONTRADO!!!!!!");
            System.out.println(c[i].toString());
            encontrado = true;
        }

    }

    for (int i = 0; i < e.length && encontrado == false; i++) {
        if (e[i].getId().compareTo(id) == 0) {
            System.out.println("!!!!!!ENCONTRADO!!!!!!");
            System.out.println(e[i].toString());
            encontrado = true;
        }

    }

    return encontrado;

}
```

Casos prueba

Base	Clase Equivalencia	Resultado
Id debe estar en el vector de objetos	id esta en el vector de objetos(1)	id no esta en el vector de objetos(2)
La entrada debe ser String	Es String(3)	No es String(4)

Casos prueba

Base	Clase Equivalencia	Resultado
“E2”	1,3	TRUE
“ASHHASD”	2,3	FALSE
“O2”	1,3	TRUE
null	4	ERROR

INSERTAR

```java
/**
 * Añade una nueva consignacion a la consigna
 * @param clienteNuevo
 * @param peso
 * @param descripcion
 * @param numObjetos
 * @param fechaConsignacion
 */
public void añadirElementos(Cliente clienteNuevo, int peso, String descripcion, int numObjetos,
                            String fechaConsignacion) {
    int newTam = numConsignaciones+1;
    Consignacion newC = new Consignacion(clienteNuevo,peso,descripcion,numObjetos,fechaConsignacion);
    Consignacion []v = new Consignacion[newTam];

    for (int i = 0; i < this.c.length; i++) {
        v[i] = c[i];
    }

    v[numConsignaciones] = newC;
    this.c = v;
    numConsignaciones++;

}
```

Casos de equivalencia

Base	Clase Equivalencia	Resultado
Los datos introducidos siguen las exigencias de entrada arriba descritas	Las siguen (1)	No las siguen(2)

Casos prueba

Base	Clase Equivalencia	Resultado
Cliente c4(datos exigidos)	1	TRUE
Consignacion o1(datosExigidos)	2	ERROR

```java
/**
 * Añade un nuevo empleado a la consigna
 * @param newE
 */
public void añadirElementos(Empleado newE) {
    int newTam = numEmpleados+1;
    Empleado []v = new Empleado[newTam];

    for (int i = 0; i < this.e.length; i++) {
        v[i] = e[i];
    }

    v[numEmpleados] = newE;
    this.e = v;
    numEmpleados++;

}
```

Casos de equivalencia

Base	Clase Equivalencia	Resultado
Se mete una clase permitida dentro del método	Sigue la clase exigidada(1)	No sigue la clase exigida(2)

Casos prueba

Base	Clase Equivalencia	Resultado
Empleado e4	1	TRUE
Consignacion o1	2	ERROR

```java
/**
 * Metodo que añade clientes a un vector dentro de la clase Pruebas
 * @param newC
 */
public static void añadirClientes(Cliente newC) {

    int newTam = numClientes+1;
    Cliente v[] = new Cliente[newTam];


    for (int i = 0; i < c.length; i++) {
        v[i] = c[i];
    }

    v[c.length] = newC;
    c = v;
    numClientes++;

}
```

Casos de equivalencia

Base	Clase Equivalencia	Resultado
Se mete una clase permitida dentro del método	Sigue la clase exigidada(1)	No sigue la clase exigida(2)

Casos prueba

Base	Clase Equivalencia	Resultado
Cliente c4	1	TRUE
Consignacion o1	2	ERROR

Actualizar

```java
/**
 * Metodo que introduce una fecha concreta y , si ya no hay ninguna introducida, la cambia dentro del objeto
 * Consignacion
 * @param pos
 */
public void actualizarFecha(int pos,String FechaSalida) {
    if (c[pos].getFechaSalida() == "Aun consignado") {
        c[pos].setFechaSalida(FechaSalida);
        System.out.println("Actualizada la fecha");
        System.out.println(c[pos].toString());
    }else {
        System.out.println("El objeto ya no esta consignado");
    }

}
```

Casos de equivalencia

Base	Clase Equivalencia	Resultado
Se inserta un String	Se inserta String (1)	No se inserta String(2)
La pos tiene un objeto dentro del vector	Lo tiene (3)	No lo tiene (4)
Fecha es “Aun consignado”	Fecha es “Aun consignado”(5)	Fecha no es “Aun consignado”(6)

Casos prueba

Base	Clase Equivalencia	Resultado
1,”23/4/25”	1,3,5	TRUE
1,”25/4/25”	1,3,6	FALSE
6,”25/4/25”	1,4	ERROR
2,4	2,3	ERROR