

## Evaluación Grupal 2

Analizando y Respondiendo las preguntas solicitadas de la evaluación 2

```
Lista de pasajeros de vuelos de negocios:
Cesar
Lista de pasajeros de vuelos economicos:
Jessica

Process finished with exit code 0
```

Se muestra la lista de pasajeros de vuelos económicos y negocios creados

Se agrega el pasajero VIP Cesar al vuelo de negocios con éxito y luego se intenta remover al pasajero del vuelo de negocios pero el sistema no lo permite ya que ningún pasajero VIP puede ser removido de ningún vuelo.

Se intenta agregar la pasajera noVIP Jessica al vuelo de Negocios sin éxito ya que solo se aceptan pasajeros VIP, luego añadimos a la pasajera noVIP Jessica al vuelo Económico con éxito ya que es un pasajero Regular.

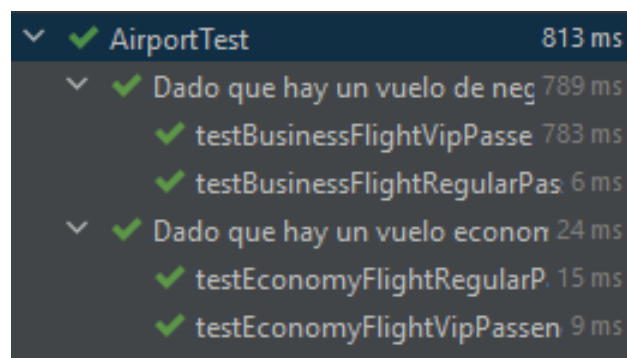
### Pregunta 1

-Los resultados que se muestran son: todas las pruebas pasadas satisfactoriamente y la medida porcentual del grado en que se ha ejecutado el código fuente del archivo.

El % de código de prueba de cobertura es igual a :

$$(\#lineasCodigoPruebaEjecutada)/(\#TotalLineasCodigoComponenteSistema) \times 100\%$$

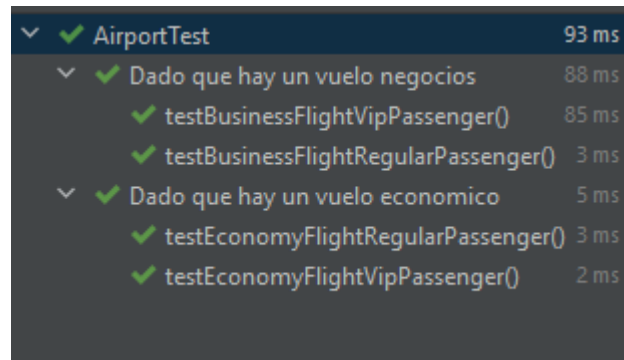
-No es del 100% porque algunos métodos no son llamados a lo largo de toda la prueba, pero el hecho que el código de prueba este a un 100% eso no significa que la prueba sea totalmente correcta o sea de buena calidad.



Coverage: AirportTest			
Element	Class, %	Method, %	Line, %
all	83% (10/12)	88% (32/36)	72% (100/138)
Airport	0% (0/1)	0% (0/1)	0% (0/16)
AirportTest	100% (3/3)	100% (8/8)	100% (29/29)
Flight	100% (1/1)	83% (5/6)	84% (16/19)
Passenger	100% (1/1)	100% (3/3)	100% (5/5)

## Pregunta 2.

-Para poder mejorar la comprensión e implementación de la lógica comercial de manera fácil y segura, evitando así que el software se degrade cada vez que éste tienda a cambiar ...



✓ AirportTest	93 ms
✓ Dado que hay un vuelo negocios	88 ms
✓ testBusinessFlightVipPassenger()	85 ms
✓ testBusinessFlightRegularPassenger()	3 ms
✓ Dado que hay un vuelo economico	5 ms
✓ testEconomyFlightRegularPassenger()	3 ms
✓ testEconomyFlightVipPassenger()	2 ms

Figura 1

**Analizando resultados:**

**en el caso del vuelo económico:**

```
@Test
public void testEconomyFlightRegularPassenger() {
    Passenger jessica = new Passenger( name: "Jessica", vip: false); //inicializando el

    assertEquals( expected: "1", economyFlight.getId()); //comparando la Id esperado con
    assertEquals( expected: true, economyFlight.addPassenger(jessica)); //comparando true
    assertEquals( expected: 1, economyFlight.getPassengers().size()); //comparando 1 con
    assertEquals( expected: "Jessica", economyFlight.getPassengers().get(0).getName());

    assertEquals( expected: true, economyFlight.removePassenger(jessica)); //comparando t
    assertEquals( expected: 0, economyFlight.getPassengers().size()); //comparando 0 con
}
```

Al testear la prueba intentamos añadir y remover un pasajero noVIP a la lista de pasajeros del vuelo económico, así como obtener el tamaño de ésta lista, siendo todas las pruebas satisfactorias como se muestra en la Fig 1

```

@Test
public void testEconomyFlightVipPassenger() {
    Passenger cesar = new Passenger( name: "Cesar", vip: true);//inicializando el pa

    assertEquals( expected: "1", economyFlight.getId());//comparando la Id esperado c
    assertEquals( expected: true, economyFlight.addPassenger(cesar));//comparando tru
    assertEquals( expected: 1, economyFlight.getPassengers().size());//comparando 1 c
    assertEquals( expected: "Cesar", economyFlight.getPassengers().get(0).getName());

    assertEquals( expected: false, economyFlight.removePassenger(cesar));//comparando
    assertEquals( expected: 1, economyFlight.getPassengers().size());//comparando 1 c
}

```

-Al testear la prueba intentamos añadir y remover un pasajero VIP a la lista de pasajeros del vuelo económico, así como obtener el tamaño de ésta lista, siendo todas las pruebas satisfactorias como se muestra en la Fig 1.

**en el caso del vuelo de negocios:**

```

@Test
public void testBusinessFlightRegularPassenger() {
    Passenger jessica = new Passenger( name: "Jessica", vip: false);//inicia

    assertEquals( expected: false, businessFlight.addPassenger(jessica));//c
    assertEquals( expected: 0, businessFlight.getPassengers().size());//comp
    assertEquals( expected: false, businessFlight.removePassenger(jessica));
    assertEquals( expected: 0, businessFlight.getPassengers().size());//comp
}

```

-Al testear la prueba intentamos añadir y remover un pasajero noVIP a la lista de pasajeros del vuelo de Negocios, así como obtener el tamaño de ésta lista, siendo todas las pruebas satisfactorias como se muestra en la Fig 1.

```

@Test
public void testBusinessFlightVipPassenger() {
    Passenger cesar = new Passenger( name: "Cesar", vip: true); //inicializ

    assertEquals( expected: true, businessFlight.addPassenger(cesar)); //con
    assertEquals( expected: 1, businessFlight.getPassengers().size()); //con
    assertEquals( expected: false, businessFlight.removePassenger(cesar)); //con
    assertEquals( expected: 1, businessFlight.getPassengers().size()); //con
}

```

-Al testear la prueba intentamos añadir y remover un pasajero VIP a la lista de pasajeros del vuelo de Negocios, así como obtener el tamaño de ésta lista, siendo todas las pruebas satisfactorias como se muestra en la Fig 1.

### Pregunta 3.

- La medida porcentual del grado en que se ha ejecutado el código fuente del archivo es:

Se observa que el % de cobertura del código de prueba ha aumentado con respecto a la Fase1 aumentando el % de efectividad de la prueba, en un promedio de 17% aproximadamente

✓ AirportTest	570 ms
✓ Dado que hay un vuelo negocios	524 ms
✓ testBusinessFlightVipPassenger()	470 ms
✓ testBusinessFlightRegularPassenger()	54 ms
✓ Dado que hay un vuelo economico	46 ms
✓ testEconomyFlightRegularPassenger()	18 ms
✓ testEconomyFlightVipPassenger()	28 ms

Coverage: AirportTest (2) ×			
Element	Clas...	Metho...	Line, %
all	100%...	100% (...)	100% (...)
AirportTest	100%...	100% (...)	100% (...)
BusinessFlight	100%...	100% (...)	100% (...)
EconomyFlight	100%...	100% (...)	100% (...)
Flight	100%...	100% (...)	100% (...)
Passenger	100%...	100% (...)	100% (...)

-Si debido a que simplificamos algunas líneas de código de la clase Flight, que no estaban siendo utilizadas en la ejecución del código, mejorando así la calidad del código.

#### Pregunta 4.

-Esta regla de Tres nos evita la redundancia del código innecesaria, facilitando en gran medida los cambios a futuro del código ahorrando así memoria caché y dándole al código una estructura mas modular.

-Se relaciona debido a que las pruebas al inicializar los pasajeros se redundan al menos 2 veces tanto en la clase **economyFlightTest** como en **businessFlightTest** de la Fase3.

```
22      @Test
23      public void testEconomyFlightRegularPassenger() {
24          Passenger jessica = new Passenger( name: "Jessica", vip: false);
25
35      @Test
36      public void testEconomyFlightVipPassenger() {
37          Passenger cesar = new Passenger( name: "Cesar", vip: true);
38
60      @Test
61      public void testBusinessFlightRegularPassenger() {
62          Passenger jessica = new Passenger( name: "Jessica", vip: false);
63
72      @Test
73      public void testBusinessFlightVipPassenger() {
74          Passenger cesar = new Passenger( name: "Cesar", vip: true);
75
```

#### Pregunta 5.

PremiumFlight.java

```
public class PremiumFlight extends Flight {

    public PremiumFlight(String id){
        super(id);
    }

    @Override
    public boolean addPassenger(Passenger passenger) {
        return false;
    }

    @Override
    public boolean removePassenger(Passenger passenger) {
        return false;
    }
}
```

## Pregunta 6.

```
public class PremiumFlight extends Flight {  
  
    public PremiumFlight(String id){  
        super(id);  
    }  
  
    @Override  
    public boolean addPassenger(Passenger passenger) {  
        return false;  
    }  
  
    @Override  
    public boolean removePassenger(Passenger passenger) {  
        return false;  
    }  
}
```

✖ AirportTest	152 ms
✖ Dado que hay un vuelo premium	110 ms
✖ Cuando tenemos un pasajero VIP	105 ms
✖ Luego agregarlo pero no puedes eliminarlo de un vi	105 ms
> ✔ Cuando tenemos un pasajero regular	5 ms
> ✔ Dado que hay un vuelo de negocios	13 ms
✔ Dado que hay un vuelo economico	29 ms
> ✔ Cuando tenemos un pasajero VIP	17 ms
✔ Cuando tenemos un pasajero regular	12 ms
✔ Luego puede agregarlo y eliminarlo de un vuelo econ	12 ms

```
public class PremiumFlight extends Flight {  
  
    public PremiumFlight(String id){  
        super(id);  
    }  
  
    @Override  
    public boolean addPassenger(Passenger passenger) {  
        if (passenger.isVip()) {  
            return passengers.add(passenger);  
        }  
        return false;  
    }  
  
    @Override  
    public boolean removePassenger(Passenger passenger) {  
        if (passenger.isVip()) {  
            return passengers.remove(passenger);  
        }  
        return false;  
    }  
}
```

✔ AirportTest	156 ms
✔ Dado que hay un vuelo premium	114 ms
✔ Cuando tenemos un pasajero VIP	106 ms
✔ Luego agregarlo pero no puedes eliminarlo de un vi	106 ms
> ✔ Cuando tenemos un pasajero regular	8 ms
> ✔ Dado que hay un vuelo de negocios	20 ms
✔ Dado que hay un vuelo economico	22 ms
> ✔ Cuando tenemos un pasajero VIP	13 ms
✔ Cuando tenemos un pasajero regular	9 ms
✔ Luego puede agregarlo y eliminarlo de un vuelo econc	9 ms

### Pregunta 7.

-Agregando la logica comercial solo para pasajeros VIP en la clase PremiumFlight del paquete de la Fase5

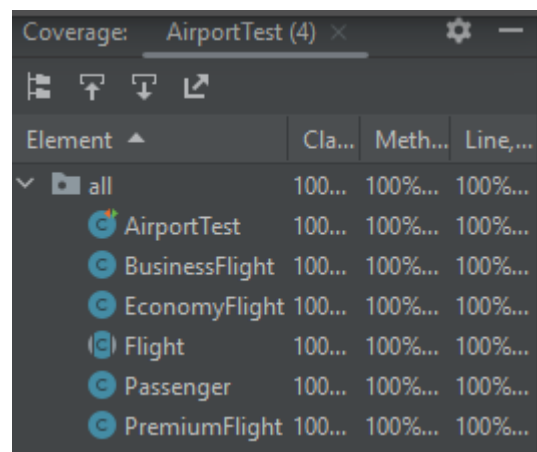
```
public class PremiumFlight extends Flight {

    public PremiumFlight(String id){
        super(id);
    }

    @Override
    public boolean addPassenger(Passenger passenger) {
        if (passenger.isVip()) {
            return passengers.add(passenger);
        }
        return false;
    }

    @Override
    public boolean removePassenger(Passenger passenger) {
        if (passenger.isVip()) {
            return passengers.remove(passenger);
        }
        return false;
    }
}
```

-Mostrando que la cobertura de código de prueba alcanza un 100% en su totalidad.



The screenshot shows the Coverage tool interface with a table of test results. The title bar indicates 'Coverage: AirportTest (4)'. The table has columns for 'Element', 'Cla...', 'Meth...', and 'Line,...'. All elements listed have 100% coverage across all metrics.

Element	Cla...	Meth...	Line,...
all	100...	100%...	100%...
AirportTest	100...	100%...	100%...
BusinessFlight	100...	100%...	100%...
EconomyFlight	100...	100%...	100%...
Flight	100...	100%...	100%...
Passenger	100...	100%...	100%...
PremiumFlight	100...	100%...	100%...



### Pregunta 8.

-Mostrando que las pruebas unitarias pasan satisfactoriamente para cada caso solicitado, incluyendo las pruebas de que un pasajero solo pueda ser añadido a un vuelo una única vez.

✓	✓	AirportTest	325 ms
✓	✓	Dado que hay un vuelo premium	186 ms
✓	✓	Cuando tenemos un pasajero VIP	181 ms
	✓	Luego agregarlo pero no puedes eliminarlo de un vuelo	144 ms
✓	✓	Entonces no puedes agregarlo a un vuelo premium	37 ms
	✓	repetition 1 of 5	15 ms
	✓	repetition 2 of 5	4 ms
	✓	repetition 3 of 5	7 ms
	✓	repetition 4 of 5	9 ms
	✓	repetition 5 of 5	2 ms
✓	✓	Cuando tenemos un pasajero regular	5 ms
	✓	Entonces no puedes agregarlo o eliminarlo de un vuelo	5 ms
✓	✓	Dado que hay un vuelo de negocios	64 ms
✓	✓	Cuando tenemos un pasajero VIP	59 ms
	✓	Luego puedes agregarlo pero no puedes eliminarlo de un vuelo	34 ms
✓	✓	Entonces no puedes agregarlo a un vuelo de negocios	25 ms
	✓	repetition 1 of 5	7 ms
	✓	repetition 2 of 5	3 ms
	✓	repetition 3 of 5	3 ms
	✓	repetition 4 of 5	7 ms
	✓	repetition 5 of 5	5 ms
✓	✓	Cuando tenemos un pasajero regular	5 ms
	✓	Entonces no puedes agregarlo o eliminarlo de un vuelo	5 ms
✓	✓	Dado que hay un vuelo economico	75 ms
✓	✓	Cuando tenemos un pasajero VIP	33 ms
	✓	Luego puedes agregarlo pero no puedes eliminarlo de un vuelo	10 ms
✓	✓	Entonces no puedes agregarlo a un vuelo economico	23 ms
	✓	repetition 1 of 5	8 ms
	✓	repetition 2 of 5	3 ms
	✓	repetition 3 of 5	5 ms
	✓	repetition 4 of 5	3 ms
	✓	repetition 5 of 5	4 ms
✓	✓	Cuando tenemos un pasajero regular	42 ms
	✓	Luego puedes agregarlo y eliminarlo de un vuelo	18 ms
✓	✓	Entonces no puedes agregarlo a un vuelo economico	24 ms
	✓	repetition 1 of 5	6 ms
	✓	repetition 2 of 5	5 ms
	✓	repetition 3 of 5	4 ms
	✓	repetition 4 of 5	7 ms
	✓	repetition 5 of 5	2 ms