# Segunda práctica de Programación 2

La segunda práctica consistirá en el diseño y posterior implementación como proyecto de Netbeans de dos versiones sobre el mismo problema.

- El tema central de ambas versiones será la programación orientada a objetos y, en ambas, no solamente usaremos clases predefinidas sino que tendremos que definir nuevas clases.
- La diferencia entre las dos versiones es que en la primera de ellas la interfaz será en base a un `ConsoleProgram` y en la segunda en un `GraphicsProgram`.
- La primera parte valdrá 7 puntos y la segunda 3, siendo esta última optativa.

## Introducción

El enunciado de la práctica está inspirado en el siguiente problema que apareció en la revista Communications of the ACM, Vol. 52, No. 8 de Agosto de 2009:

> *Cien personas embarcan en un avión cuyas plazas han sido vendidas en su totalidad. Desgraciadamente, la primera persona de la cola ha perdido su tarjeta de embarque al entrar en el avión y se sienta en un asiento elegido al azar. Los pasajeros siguientes se sentarán en el asiento indicado por su tarjeta si éste está libre o bien harán como el primer pasajero y elegirán un asiento libre al azar. ¿Cuál es la probabilidad de que el último pasajero encuentre el asiento correspondiente a su tarjeta de embarque libre?*

## Diseño orientado a objetos

En esta parte del enunciado presentaremos algunas de las clases que deberéis implementar, indicando su nombre, constructor y algunos de los métodos que deberán tener (obviamente podréis añadir todos los métodos privados adicionales que queráis) y daremos indicaciones sobre su funcionamiento y algunas pistas sobre su implementación. Tened en cuenta que el diseño será ligeramente diferente a la historia original en la que está inspirado.

### Clase Passenger

- Representará a cada uno de los pasajeros.

- Tendrá un constructor con un parámetro que sea un número que indicará el asiento que tiene asignado.
- Tendrá tres métodos públicos:
  - `int getAssignedSeat()` que devolverá el asiento que tiene asignado
  - `void loseBoardingTicket()` que hará que ese pasajero pierda la tarjeta de embarque
  - `boolean hasLostBoardingCard()` que responderá si ese pasajero ha perdido o no la tarjeta de embarque.

## Clase `Passengers`

- Esta clase representará al grupo de pasajeros (pasaje) que están esperando el avión.
- Tendrá un constructor con un parámetro entero (n) que indicará el número de pasajeros a considerar. Internamente lo que hará el constructor será:
  - Crear un array de n referencias a `Passenger`
  - Inicializar cada elemento del array con un `Passenger` (asignado a un asiento diferente).
  - Desordenar el array de pasajeros usando lo que se comenta en el apéndice de la práctica.
  - Indicando que el primer pasajero ha perdido la tarjeta de embarque.
- Como los pasajeros entrarán en el avión de uno en uno, nos inspiraremos en la clase `StringTokenizer`, e implementaremos dos métodos públicos:
  - `boolean hasMorePassengers()` para saber si aún quedan pasajeros por embarcar
  - `Passenger nextPassenger()` que devuelve el siguiente pasajero por embarcar.

## Clase `Plane`

- Representará el avión y será quien se encargue de gestionar los asientos.
- Tendrá un constructor con un parámetro entero que indicará el número de asientos del avión y otro que será el programa principal (su porqué lo explicaremos más adelante).
  - Internamente mantendrá un vector de `Passenger` en el que el índice del vector indicará el asiento (menos uno, ya que no seremos frikis y los asientos irán numerados desde uno) y el valor será el pasajero que lo ocupa (o `null` si el

asiento está libre). Inicialmente todos los asientos estarán libres.

- Definirá el siguiente método público:
  - `void board(Passengers)` que ejecutará el embarque del pasaje en el avión, asignando a cada pasajero un asiento. Para ello usará el método `assignSeat`.
- Como mínimo tendremos los siguientes métodos privados (ya que como alguno de ellos es complejo, lo descompondremos en métodos más pequeños[1]):
  - `int assignSeat(Passenger)` es quien asigna un asiento al pasajero dado en función de si éste ha perdido la tarjeta de embarque o no y de si el asiento que le corresponde está libre u ocupado. Es quién llamará a los diferentes métodos que vayan mostrando la ejecución.

# Parte obligatoria (7 puntos)

En esta parte los métodos de visualización simplemente escriben mensajes en la Pantalla. Por ello tendremos la clase:

## Clase `ConsoleBoarding`

Que será el programa principal y que extenderá `ConsoleProgram`. Ademán contendrá los métodos que vayan mostrando la traza de la ejecución, que serán:

- `void showAssignedCorrectSeat(Passenger)` que sirve para indicar que se ha asignado al pasajero el asiento que le correspondía.
- `void showAssignedRandomSeat(Passenger, int)` que sirve para indicar que se ha asignado al pasajero un asiento al azar indicado por el entero.

Para que podamos usarlos dentro de la clase `Plane`, pasaremos la instancia correspondiente al programa principal cuando creemos la instancia de `Plane` en el método `run`, es decir:

```
1 public class ConsoleBoarding extends ConsoleProgram {
2   ...
3   public void run() {
4     ...
```

[1] La evaluación tendrá en cuanta si habéis realizado diseño descendente (descomposición en submétodos más pequeños), si el código es legible, nombres de variables, etc, etc.

```
    int size = readInt("Número de pasajeros: ");
    Plane plane = new Plane(size, this);
    ...
  }
  public void showAssignedCorrectSeat(Passenger p) {
    ...
  }
  public void showAssignedRandomSeat(Passenger p, int s) {
     ...
  }
}
```

Así que dentro de la clase `Plane`, nos guardaremos una referencia a esta instancia, es decir:

```
public class Plane {
  ...
  private ConsoleBoarding program;
  public Plane(int n, ConsoleBoarding program) {
    ...
    this.program = program;
  }
  ...
}
```

Y usaremos esta referencia dentro del método que asigna el asiento para que el programa muestre qué ha pasado, es decir:

```
public int assignSeat(Passenger passenger) {
  ...
  this.program.showAssignedCorrectSeat(passenger);
  ...
}
```

Implementando de esta manera este método el programa irá indicando a qué pasajero se le asigna un asiento y si dicho asiento es el que le correspondía o bien se le ha asignado al azar.

## Parte optativa (3 puntos)

En esta parte sustituiremos la interfaz "console" por una que muestre visualmente qué está pasando.

Os comento una posible interfaz sencilla que podéis complicar o modificar[2] (p. ej. mostrando cómo se mueve un pasajero por el avión desde la entrada hasta su asiento) a vuestro antojo:

- Cada asiento estará representado por un rectángulo que será verde si está libre, rojo si está asignado a un pasajero al que no le tocaba y amarillo si está asignado al pasajero que le tocaba.
- En la zona inferior de la pantalla habrá un `GLabel` que usaremos para mostrar el texto de lo que está pasando (a modo similar a lo que hacíamos en la parte obligatoria).
- Esperaremos a que el usuario pulse el botón del ratón (con el método `waitForClick()` de la clase `GraphicsProgram`) entre asignaciones, para ir viendo cómo se modifican los asientos.

## Clase `GraphicsBoarding`

Que será el programa principal y extenderá `GraphicsProgram`. En el run creará los elementos de la interfície (los que representan los asientos, etc, etc) e implementará los métodos:

- `void showAssignedCorrectSeat(Passenger)`
- `void showAssignedRandomSeat(Passenger, int)`

que ahora mostrarán gráficamente qué está pasando.
Para que todo funcione, deberéis modificar ligeramente las clase `Plane` de la solución del apartado anterior:

- Ahora lo que `Plane` guarda es una referencia a `GraphicsBoarding`
- Dentro del método que va asignando los pasajes añadiremos una llamada a `waitForClick` sobre esta referencia.

# Apéndice: El algoritmo de desordenamiento

Para desordenar el vector de pasajeros podéis usar el Algortimo de Fisher-Yates:

```
1 // Desordenar un array a de n elementos
2 for (int i = n-1; i > 0; i--) {
3   int j = número al azar tal que 0<=j<=i
4   intercambiar a[i] con a[j]
5 }
```

[2] Un consejo: comenzad por una simple como ésta que os permitirá familiarizaros con los elementos gráficos e id complicándola poco a poco. Si en algún momento os quedáis encallados (p.e. simulando el movimiento de los pasajeros en el avión), tendréis al menos una versión correcta para entregar.

Para generar el número al azar (y de hecho para asignar asientos al azar) usaréis la clase `RandomGenerator` explicada en los apuntes.

## Formato de la entregas:

La entrega se hará vía **sakai** y consistirá en un archivo (comprimido con **ZIP**) que contenga:

- los **proyectos de netbeans** que solucionen los problemas
- un **pequeño informe** (máximo 5 páginas) en **PDF** explicando los aspectos relevantes de vuestra implementación y qué partes os han costado más, qué problemas habéis encontrado, en qué os habéis encallado, así como una explicación de la implementación de vuestras clases (p.e. usando diagramas que muestren cómo modificáis los vectores, etc.), etc.
- cada clase deberá tener la documentación en **javadoc** para cada uno de los métodos públicos.
- la práctica se realiza **individualmente**.
- la práctica es un **20%** de la nota de la asignatura.
- la fecha límite es el **lunes 6 de junio a las 8 de la mañana**.