


<b>Prueba Recuperatoria 2º Trimestre - Tiempo 3h</b>	<b>13 de Mayo de 2020 - 16:00h</b>
Nombre y Apellidos	DNI/NIE: Firma:
1º Desarrollo de Aplicaciones Web (Vespertino) Módulo: <b>Programación</b>	 IES Alonso de Avellaneda (Alcalá)

**Programas (10 puntos)** (Recogida de los fuentes del programa en repositorio github de cada alumno). Se hará una primera entrega cuando diga el profesor de lo que se tenga hecho hasta ese momento y después se hará la entrega final. El examen se entregará en el tiempo establecido.

### Ejercicio 1 ( 4 puntos)

Escribe la clase **ReconocimientoPatrones.java** que utilice un método con la signatura siguiente. Este método debe verificar si una array bidimensional (matriz) tiene 4 números consecutivos del mismo valor. ya sea horizontal, vertical como diagonal. **Sólo se pueden utilizar arrays unidimensionales y bidimensionales. No se pueden utilizar ni Colecciones, ni Arrays, ni ninguna librería adicional de manejo de arrays.)**

```
public static boolean tieneCuatroConsecutivos(int[][] valores)
```

Escribe un programa test **ReconocimientoPatronesTest.java** que solicite al usuario la dimensión del array bidimensional y luego se introduzcan los valores del propio array. El programa debe mostrar la matriz y decir si tiene 4 números consecutivos.

0	1	0	3	1	6	1
0	1	6	8	6	0	1
5	6	2	1	8	2	9
6	5	6	1	1	9	1
1	3	6	1	4	0	7
3	3	3	3	4	0	7

0	1	0	3	1	6	1
0	1	6	8	6	0	1
5	5	2	1	8	2	9
6	5	6	1	1	9	1
1	5	6	1	4	0	7
3	5	3	3	4	0	7

0	1	0	3	1	6	1
0	1	6	8	6	0	1
5	6	2	1	6	2	9
6	5	6	6	1	9	1
1	3	6	1	4	0	7
3	6	3	3	4	0	7

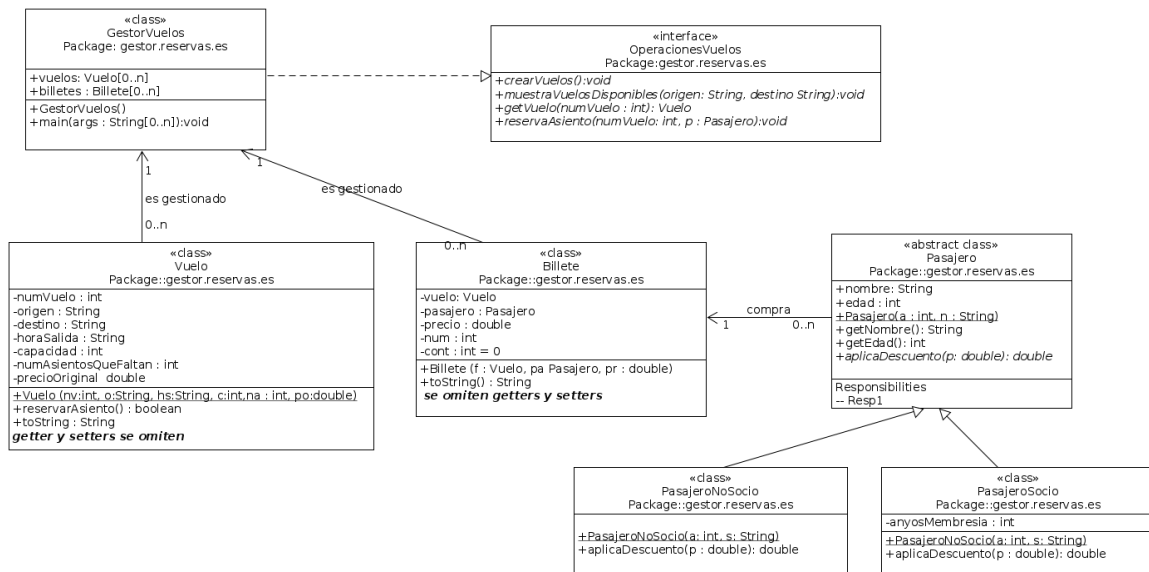
0	1	0	3	1	6	1
0	1	6	8	6	0	1
9	6	2	1	8	2	9
6	9	6	1	1	9	1
1	3	9	1	4	0	7
3	3	3	9	4	0	7

```
Introduce número de filas y columnas: 4 4
Introduce los valores:
1 0 0 0
1 1 0 0
0 0 1 0
0 0 0 1
La matriz:
1 0 0 0
1 1 0 0
0 0 1 0
0 0 0 1
tiene 4 consecutivos?: true
```

## Ejercicio 2 (4 puntos)

La empresa de viajes **GestorReservasVuelos** requiere la implementación de un programa en el que se puedan crear vuelos con la información necesaria de los mismos: número, capacidad, origen, destino, precio inicial, hora y fecha de salida, etc. Cada vuelo tiene un número de plazas disponibles. Los pasajeros pueden reservar asientos en los vuelos y el precio de compra puede verse reducido si el pasajero es miembro del club de socios. Si el pasajero tiene más de 1 año y hasta 5 años de membresía opta a una reducción del 10% del precio. Si tiene más de 5 años el descuento es del 50%. Si el pasajero tiene más de 65 años tiene un descuento del 10% pero no contará si es miembro en las anteriores condiciones. Se deben controlar las plazas disponibles, mostrando un error en caso de que no existan.

A continuación se presenta el diagrama de clases y una descripción de las mismas:



- Se pueden escribir otros métodos auxiliares que se necesiten. A continuación se explican los métodos (se omiten las cabeceras de forma completa, puede comprobar las cabeceras en el diagrama UML).
- Debes implementar el programa siguiendo el diagrama de clases dado con todas sus clases, interfaces, clases abstractas, relaciones, atributos y métodos. (0.3 p)
- Todos los métodos `toString()` debidamente implementados: (0.2 p)
- Constructores que inicialicen los parámetros debidamente: (0.25 p)

### GestorVuelos.java

**main**: se debe crear en el método main un menú sencillo que pueda gestionar las reservas y vuelos. (0.3 p).

### OperacionesVuelos.java (estos métodos se implementan donde corresponda)

- **crearVuelos()** Se deben poder crear vuelos con los siguientes parámetros: numVuelo, origen, destino, hora y fecha de salida, capacidad de vuelo (asientos libres). precio inicial del vuelo. Se pueden crear tantos vuelos como se quieran. **Las fechas y horas** deben tener el formato: dd/mm/aaaa hh:mm:ss. La capacidad del vuelo puede ser un valor de 200 plazas. Aunque se puede probar con un valor cercano a 0 ya que se deben controlar que existan plazas disponibles. El precio inicial del vuelo luego puede variar por los descuentos. (0.5 p)
- **void muestraVuelosDisponibles(String origen, String destino)**: muestra todos los vuelos creados por origen y destino (0.5)
- **getVuelo(int numVuelo)**: Conocer la información de un vuelo dado su número (0.25)

- `void reservaAsiento(int numVuelo, Pasajero p)`: reserva un sitio **si quedan disponibles** teniendo en cuenta que si un pasajero tiene edad  $\geq 65$  años o es miembro del club pasajero frecuente de entre más de 1 año y hasta 4 de antigüedad, debe tener un descuento del 10%. Si tiene más de 5 años el descuento es de 50%. Este método deberá preguntar por los asientos mediante el método `getNumSitiosQueFaltan()` y aplicar el descuento con `aplicaDescuento(double p)` dependiendo el tipo de pasajero que es. (1p)

#### **PasajeroNoSocio.java y PasajeroSocio.java**

- `aplicaDescuento(double p)`: depende si es un pasajero socio o no lo es. Si es un pasajero socio y tiene más de 5 años de membresía tendrá un descuento del 50%, si tiene entre más de 1 año y 5 tendrá un descuento del 10%. Si no es pasajero socio, tendrá un descuento del 10% si tiene 65 años o más. (0.25 cada uno)

#### **Vuelo.java**

- `reservaDeAsiento()`: método que verifica que el número de asientos es mayor que 0 y decrementa la capacidad en uno cuando el asiento ha sido reservado. (0.2 p)

```

Introduzca c si quiere Crear un vuelo
Introduzca m si quiere Mostrar todos los vuelos
Introduzca i si quiere Saber información de un vuelo
Introduzca r si quiere Reservar un asiento
Introduzca s si quiere Salir del programa
c
Introduce Número de Vuelo: 303
Introduce origen del Vuelo (p.e. Madrid): París
Introduce destino del Vuelo (p.e. Barcelona): Londres
Introduce horay fecha de salida del vuelo (10:10 01/12/2020): 09:09:09
02/12/2020
Introduce capacidad del vuelo (asientos libres): 30
Introduce el precio original del vuelo: 700
El siguiente vuelo ha sido creado:
Flight 303,París to Londres,09:09:09 02/12/2020, original price: 700.0$
Introduzca c si quiere Crear un vuelo
Introduzca m si quiere Mostrar todos los vuelos
Introduzca i si quiere Saber información de un vuelo
Introduzca r si quiere Reservar un asiento
Introduzca s si quiere Salir del programa
m
Introduzca origen del vuelo: Madrid
Introduzca destino del vuelo: Barcelona
Listado de vuelos disponibles:
Flight 707,Madrid to Barcelona,10:10:10 01/12/2020, original price: 500.0$
Introduzca c si quiere Crear un vuelo
Introduzca m si quiere Mostrar todos los vuelos
Introduzca i si quiere Saber información de un vuelo
Introduzca r si quiere Reservar un asiento
Introduzca s si quiere Salir del programa
i
Introduzca número de vuelo: 303
Información para el vuelo: 303:
Flight 303,París to Londres,09:09:09 02/12/2020, original price: 700.0$
Introduzca c si quiere Crear un vuelo
Introduzca m si quiere Mostrar todos los vuelos
Introduzca i si quiere Saber información de un vuelo
Introduzca r si quiere Reservar un asiento
Introduzca s si quiere Salir del programa

```

```
r
Si el pasajero no es miembro pulsar n, si el pasajero es miembro pulsar m
m
Introduzca edad del pasajero: 67
Introduzca nombre del pasajero: Jhon Smith
Introduzca el número de vuelo: 303
Por cuántos años ha sido miembro el pasajero: 2
Usted ha reservado con éxito un asiento del vuelo 303
ticket: Jhon Smith, Flight 303, París to Londres, 09:09:09 02/12/2020, original
price: 700.0$, ticket price: 630.0$
```

### Ejercicio 3 ( 2 puntos)

Escribe el programa **VerificaPassword.java** que tenga un método que verifique un password según las siguiente reglas:

**Se pueden utilizar StringBuilder, Strings, Objetos de envoltorio o Wrappers, arrays y matrices**

- Un password debe tener al menos 10 caracteres.
- Un password sólo puede tener sólo letras y dígitos.
- Un password debe tener al menos 3 dígitos.
- Un password debe tener una letra en mayúsculas.

```
boolean esValidoPassword (String password)
```

### Rúbrica

**Todos los programas, métodos, clases, etc. deben estar bien especificados, implementados y con ejecución correcta. Las salidas no puntúan por separado sino que son una consecuencia del correcto funcionamiento de métodos, programas, etc.**

```
E1: 1p por cada patrón especificación, implementación y ejecución correctas. 0
en caso contrario
E2: 100% cada apartado con especificación, implementación y ejecución correctas.
0 en caso contrario
E3: 0.5 puntos cada parte con especificación, implementación y ejecución
correctas. 0 en caso contrario.
```

**Esta prueba será revisada mediante software antiplagios entre exámenes entregados así como de recursos de internet**