

## TEMA 2

### Conceptos de Java para Estructuras de Datos

### Fallos de ejecución: clasificación, representación y tratamiento

#### EJERCICIO ADICIONAL RESUELTO

Diseña las clases *PersonalDeVuelo*, *Auxiliar*, *Piloto* y *Vuelo*, de forma que::

- Para cada miembro del personal de vuelo de una compañía se quiere conocer su nombre y si sabe hablar inglés
- Hay dos categorías para el personal de vuelo: auxiliares de vuelo y pilotos. Todo miembro del personal debe pertenecer a una de esas dos categorías
- Entre pilotos y auxiliares, no se permite crear más de 100 personas. Si se intenta crear un nuevo miembro con la plantilla llena se avisará con la excepción *ExcepcionPlantillaLlena*
- Un vuelo tiene una ciudad de origen y una de destino (de tipo *String*) y una tripulación formada por ocho miembros del personal de vuelo
- En un vuelo habrán dos métodos para añadir los miembros a la tripulación:

```
public void nuevoPiloto(String nombre, boolean hablaIngles) throws ExcepcionErrorTripulacion
public void nuevoAuxiliar(String nombre, boolean hablaIngles) throws ExcepcionErrorTripulacion
```

Cualquier problema al configurar la tripulación se notificará con la excepción *ExcepcionErrorTripulacion*

- Un vuelo tiene un método *volar()* que muestra por pantalla el mensaje “Despegar!!!” si todo es correcto: deben haber 2 pilotos, 6 auxiliares y, al menos, 2 miembros que sepan inglés. Si hay algún error se avisará con la excepción *ExcepcionErrorTripulacion*

#### Solución:

```
public class ExcepcionPlantillaLlena extends Exception {
    public ExcepcionPlantillaLlena() {
        super();
    }
    public ExcepcionPlantillaLlena(String msg) {
        super(msg);
    }
}
```

```
public class ExcepcionErrorTripulacion extends Exception {
    public ExcepcionErrorTripulacion() {
        super();
    }
    public ExcepcionErrorTripulacion(String msg) {
        super(msg);
    }
}
```

```

public abstract class PersonalDeVuelo {
    protected static final int MAX_CONTRATOS = 100;
    protected static int contratos = 0;
    protected String nombre;
    protected boolean hablaIngles;

    public PersonalDeVuelo(String nombre, boolean hablaIngles)
        throws ExcepcionPlantillaLlena {
        if (contratos >= MAX_CONTRATOS)
            throw new ExcepcionPlantillaLlena("Todas las vacantes ocupadas");
        this.nombre = nombre;
        this.hablaIngles = hablaIngles;
        contratos++;
    }
    public String nombre() {
        return nombre;
    }
    public boolean hablaIngles() {
        return hablaIngles;
    }
}

```

```

public class Auxiliar extends PersonalDeVuelo {
    public Auxiliar(String nombre, boolean hablaIngles)
        throws ExcepcionPlantillaLlena {
        super(nombre, hablaIngles);
    }
}

```

```

public class Piloto extends PersonalDeVuelo {
    public Piloto(String nombre, boolean hablaIngles)
        throws ExcepcionPlantillaLlena {
        super(nombre, hablaIngles);
    }
}

```

```

public class Vuelo {
    private String origen, destino;
    private PersonalDeVuelo[] tripulacion;
    private int numTripulantes;
    public static final int NUM_TRIPULANTES = 8;

    public Vuelo(String origen, String destino) {
        tripulacion = new PersonalDeVuelo[NUM_TRIPULANTES];
        this.origen = origen;
        this.destino = destino;
        this.numTripulantes = 0;
    }
    public void nuevoPiloto(String nombre, boolean hablaIngles)
        throws ExcepcionErrorTripulacion {
        if (numTripulantes >= NUM_TRIPULANTES)
            throw new ExcepcionErrorTripulacion("Tripulación completa");
        try {
            tripulacion[numTripulantes++] = new Piloto(nombre, hablaIngles);
        } catch (ExcepcionPlantillaLlena e) {
            throw new ExcepcionErrorTripulacion(e.getMessage());
        }
    }
}

```

```

public void nuevoAuxiliar(String nombre, boolean hablaIngles)
    throws ExcepcionErrorTripulacion {
    if (numTripulantes >= NUM_TRIPULANTES)
        throw new ExcepcionErrorTripulacion("Tripulación completa");
    try {
        tripulacion[numTripulantes++] = new Auxiliar(nombre, hablaIngles);
    } catch (ExcepcionPlantillaLlena e) {
        throw new ExcepcionErrorTripulacion(e.getMessage());
    }
}

public void volar() throws ExcepcionErrorTripulacion {
    if (numTripulantes != NUM_TRIPULANTES)
        throw new ExcepcionErrorTripulacion("Número incorrecto de tripulantes");
    int numPilotos = 0, numIngles = 0;
    for (int i = 0; i < tripulacion.length; i++) {
        if (tripulacion[i] instanceof Piloto) numPilotos++;
        if (tripulacion[i].hablaIngles()) numIngles++;
    }
    if (numPilotos != 2)
        throw new ExcepcionErrorTripulacion("Número incorrecto de pilotos");
    if (numIngles < 2) throw new ExcepcionErrorTripulacion("Faltan tripulantes "
        + "con conocimientos de inglés");
    System.out.println("Despegar!!!");
}
}

```