# Programación

# Bloque 04 - Desarrollo de Clases

# Desarrollo de clases con Excepciones

**Todos las clases correspondientes a los ejercicios de esta relación se deben crear dentro del paquete `prog.unidad04.actividad403`, creando sub-paquetes de nombre `ejercicio01`, `ejercicio02`, etc., uno por ejercicio de la relación. Asimismo hay que documentar todas las clases usando el formato JavaDoc**

1. Crea una nueva versión mejorada de la clase Pizza creada en el ejercicio 01 de la relación 4.02 para que mejorarla de forma que el estado de los objetos no pueda ser inconsistente. Mas concretamente, deberás realizar los siguientes cambios:
   - Si el tamaño o tipo de la pizza que se pasa al constructor no es uno de los permitidos (ver la documentación de la clase), se debe lanzar una excepción no chequeada `IllegalArgumentException`.
   - Si se intenta servir una pizza ya servida se debe lanzar una excepción chequeada, que deberás crear, de tipo `PizzaAlreadyServedException`.

   Se proporcionan los archivos `Actividad_4.03a_pruebas.zip` que contiene la clase `PruebaPizza` que sirve para probar la clase y que debe generar la salida que se muestra a continuación. Asimismo se proporciona el archivo `Actividad_4.03a_doc.zip` que contiene la documentación actualizada de la clase `Pizza`.

**Ejemplo de uso:**

```
PROGRAMA DE PRUEBA DE PIZZAS 2
Se ha intentando crear una pizza con valores erróneos y ha fallado
Se ha intentando crear una pizza con valores erróneos y ha fallado
La pizza se sirvió con éxito
Pizza 1. Tamaño: mediana, Tipo: margarita, Estado: pedida
Pizza 2. Tamaño: familiar, Tipo: funghi, Estado: servida
Pizza 3. Tamaño: mediana, Tipo: cuatro quesos, Estado: pedida
Ok. La pizza ya se había servido y ha fallado al hacerlo otra vez
Número de pizzas creadas: 3
Número de pizzas servidas: 1
```

2. Crea una nueva versión mejorada de la clase `Persona` creada en el ejercicio 03 de la relación 4.02. En concreto deberás cambiar:
   - Los constructores para asegurar que se cumple lo siguiente (en caso de que no se cumpla se deberá lanzar una excepción de tipo `IllegalArgumentException`):
     - `edad` debe ser mayor o igual a cero
     - `sexo` deberá ser uno de 'H' o 'M', en mayúsculas

- `peso` deberá ser mayor o igual a cero
- `altura` deberá ser mayor o igual a cero

Se propociona el archivo `Actividad_4.03b_doc.zip` que contiene la documentación de la clase `Persona` y el archivo `Actividad_4.03b_pruebas.zip` con el programa de pruebas que proporciona la salida que se muestra a continuación.

**Ejemplo de uso:**

```
PERSONAS
Imprimimos las tres personas
Nombre: "", Edad: 0, Sexo: M, Peso: 0.0, Altura: 0.0, DNI:
81640543L
Nombre: "Paco", Edad: 17, Sexo: H, Peso: 0.0, Altura: 0.0, DNI:
79678783K
Nombre: "Paqui", Edad: 26, Sexo: M, Peso: 76.6, Altura: 1.67, DNI:
81976704N
Imprimimos los IMCs de las tres personas
IMC Persona 1 = 0.0
IMC Persona 2 = 0.0
IMC Persona 3 = 27.466026031768795
Imprimimos si los pesos de las personas es correcto o no
Peso correcto Persona 1 = -1
Peso correcto Persona 2 = -1
Peso correcto Persona 3 = 1
Son las personas mayores de edad?
Persona 1: no
Persona 2: no
Persona 3: si
Intentando crear una persona con edad incorrecta
OK. Se ha impedido la creación de la persona con la edad
incorrecta
Intentando crear una persona con sexo incorrecto
OK. Se ha impedido la creación de la persona con sexo incorrecto
Intentando crear una persona con peso incorrecto
OK. Se ha impedido la creación de la persona con peso incorrecto
Intentando crear una persona con altura incorrecta
OK. Se ha impedido la creación de la persona con altura incorrecta
```

3. Crea una nueva versión mejorada de la clase `Lavadora` creada en el ejercicio 04 de la relación 4.02. En concreto deberás cambiar:
   - Los constructores para asegurar que se cumple lo siguiente (en caso de que no se cumpla se deberá lanzar una excepción de tipo `IllegalArgumentException`):
     - precioBase debe ser positivo.
     - color debe ser uno de "blanco", "negro", "azul", "rojo", "gris"
     - consumoEnergetico deberá ser una letra de la 'A' a la 'F', mayúscula.
     - peso y carga deben ser mayores de cero

   Se propociona el archivo `Actividad_4.03c_doc.zip` que contiene la documentación de la clase `Persona` y el archivo

`Actividad_4.03c_pruebas.zip` con el programa de pruebas que proporciona la salida que se muestra a continuación.

**Ejemplo de uso:**

```
LAVADORAS
Imprimimos las tres lavadoras
Precio Base: 200.0, Color: blanco, Consumo Energético: F, Peso:
15.0, Carga Máxima: 5.0
Precio Base: 500.0, Color: blanco, Consumo Energético: F, Peso:
25.0, Carga Máxima: 5.0
Precio Base: 650.0, Color: rojo, Consumo Energético: A, Peso:
75.0, Carga Máxima: 8.0
Imprimimos los precios finales de las tres lavadoras
Precio Lavadora 1 = 220.0
Precio Lavadora 2 = 560.0
Precio Lavadora 3 = 830.0
Intentando crear una lavadora con un precio inválido
OK. Se ha impedido la creación de una lavadora con un precio
incorrecto
Intentando crear una lavadora con un color niválido
OK. Se ha impedido la creación de una lavadora con un color
inválido
Intentando crear una lavadora con una clasificación energética
inválida
OK. Se ha impedido la creación de una lavadora con una
clasificación energética inválida
Intentando crear una lavadora con un peso inválido
OK. Se ha impedido la creación de una lavadora con un peso
inválido
Intentando crear una lavadora con una carga inválida
OK. Se ha impedido la creación de una lavadora con una carga
inválida
```

4. Crea una nueva versión mejorada de la clase `Zona` creada en el ejercicio 05 de la relación 4.02. En concreto deberás cambiar:
   - Mejorar el constructor de `Zona` para que no permita que el número de entradas de una zona sea inferior a 1. En caso de que se intente se debe lanzar la excepción chequeada `ZonaException`.
   - Cambiar el método `vender` para que no devuelva ningún valor. En su lugar se supondrá que se venden las entradas y en caso de que no haya entradas suficientes se lanzará la excepción `ZonaException`.

   Modifica la aplicación `VendeEntradas` para que se adapte a estos cambios. Se propociona el archivo `Actividad_4.03d_doc.zip` que contiene la documentación de la clase `Zona`.

**Ejemplo de uso:**

```

GESTION DE ENTRADAS
-------------------
1.- Mostrar asientos libres por zona
2.- Vender asientos
3.- Salir del programa
Elija una opción (1, 2, 3): 1
```

```
ASIENTOS LIBRES POR ZONA
------------------------
Zona Platea: 150
Zona Palco: 50
Zona Ático: 100

GESTION DE ENTRADAS
-------------------
1.- Mostrar asientos libres por zona
2.- Vender asientos
3.- Salir del programa
Elija una opción (1, 2, 3): 2

VENTA DE ENTRADAS
-----------------
Seleccione la zona en la que se localizan los asientos a vender (1
= Platea, 2 = Palco, 3 = Ático): 1
Introduzca el número de asientos a vender: 20
Venta realizada correctamente

GESTION DE ENTRADAS
-------------------
1.- Mostrar asientos libres por zona
2.- Vender asientos
3.- Salir del programa
Elija una opción (1, 2, 3): 1

ASIENTOS LIBRES POR ZONA

------------------------
Zona Platea: 130
Zona Palco: 50
Zona Ático: 100
GESTION DE ENTRADAS
-------------------
1.- Mostrar asientos libres por zona
2.- Vender asientos
3.- Salir del programa
Elija una opción (1, 2, 3): 2

VENTA DE ENTRADAS
-----------------
Seleccione la zona en la que se localizan los asientos a vender (1
= Platea, 2 = Palco, 3 = Ático): 1
Introduzca el número de asientos a vender: 140

No se pudo realizar la venta de entradas. Probablemente no haya
entradas suficientes en la zona elegida.
GESTION DE ENTRADAS
-------------------
1.- Mostrar asientos libres por zona
2.- Vender asientos
```

```
3.- Salir del programa
Elija una opción (1, 2, 3): 1

ASIENTOS LIBRES POR ZONA
------------------------
Zona Platea: 130
Zona Palco: 50
Zona Ático: 100

GESTION DE ENTRADAS
-------------------
1.- Mostrar asientos libres por zona
2.- Vender asientos
3.- Salir del programa
Elija una opción (1, 2, 3): 3
Saliendo del programa.....
```

5. Crea una nueva versión mejorada de las clases creadas en el ejercicio 06 de la relación 4.02 (Punto, Triangulo, etc.) En concreto deberás cambiar:
   - Mejorar el constructor de `Triangulo` para que produzca una excepción no chequeada del tipo `FiguraException` en caso de que los tres vértices que se proporcionen estén alineados.
   - Modificar también el constructor de `Circulo` para que lance la misma excepción en caso de que el radio sea menor de cero.

   Se proporcionan los archivos `Actividad_4.03e_pruebas.zip` que contiene la clase `PruebaFiguras` que sirve para probar las clases y que debe generar la salida que se muestra a continuación. Asimismo se proporciona el archivo `Actividad_4.03e_doc.zip` que contiene la documentación de las clases `Punto`, `Triangulo`, `Circulo` y `Rectangulo`.

**Ejemplo de uso:**

```
PROGRAMA DE PRUEBA DE FIGURAS
Intentado crear un triángulo con los vertices alineados
OK. Se evitó crear el triángulo
Intentado crear un círculo con el radio negativo
OK. Se evitó crear el círculo
Area triangulo = 0.49999999999999983
Perimetro triangulo = 3.414213562373095
Area circulo = 19.634954084936208
Perimetro circulo (circunferencia) = 15.707963267948966
Area rectángulo no cuadrado = 6.0
Perimetro rectángulo no cuadrado = 10.0
Area rectángulo cuadrado = 4.0
Perimetro rectángulo cuadrado = 8.0
El primer rectángulo es cuadrado?: no
El segundo rectángulo es cuadrado?: si
```

6. Crea una nueva versión mejorada de la clase `TarjetaRegalo` creada en el ejercicio 07 de la relación 4.02. En concreto deberás cambiar:
   - Mejorar el constructor para que no se admita un saldo negativo inicial. Se deberá lanzar `IllegalArgumentException` en este caso.

- Mejorar gasta para que se lance una excepción si se intenta gastar una cantidad superior al saldo de la tarjeta. La excepción será la no chequeada `InsufficientFundsException`

Se propociona el archivo `Actividad_4.03f_doc.zip` que contiene la documentación de la clase `TarjetaRegalo` y el archivo `Actividad_4.03f_pruebas.zip` con el programa de pruebas que proporciona la salida que se muestra a continuación.

**Ejemplo de uso:**

```
TARJETAS REGALO MANEGO
Las tarjetas recien creadas tienen la siguiente información
Tarjera nº 68077 - Saldo 100.0€
Tarjera nº 43352 - Saldo 120.0€
Intentamos crear una tarjeta con saldo negativo
OK. Se ha impedido la creación de la tarjeta
Hacemos algún gasto en las tarjetas
Intentando gastar en una tarjeta más cantidad que el saldo
OK. Se ha prohibido gastar más que el saldo
Después del gasto las tarjetas tienen el siguiente estado
Tarjera nº 68077 - Saldo 50.550000000000004€
Tarjera nº 43352 - Saldo 115.0€
Fusionamos las dos tarjetas y obtenemos una nueva
Después de la fusión, las tarjetas tienen el siguiente estado
Tarjera nº 68077 - Saldo 0.0€
Tarjera nº 43352 - Saldo 0.0€
Tarjera nº 51481 - Saldo 165.55€
```

7. Crea una nueva versión mejorada de la clase IntervaloTiempo creada en el ejercicio 08 de la relación 4.02. En concreto deberás cambiar:
   - Modificar el constructor para que sólo admita valores de minutos y segundos entre 0 y 59, ambos incluidos. Si no se cumple esto debe lanzar la excepción `IllegalArgumentException.`
   - Modificar el método resta para que lance la excepción chequeada `IntervaloTiempoException` en caso de que se intente restar un intervalo de tiempo más largo de uno corto.

   Se proporcionan los archivos `Actividad_4.03g_pruebas.zip` que contiene la clase `PruebaIntervaloTiempo` que sirve para probar la clase y que debe generar la salida que se muestra a continuación. Asimismo se proporciona el archivo `Actividad_4.03g_doc.zip` que contiene la documentación de la clase `IntervaloTiempo`.

**Ejemplo de uso:**

```
PROGRAMA DE PRUEBA DE INTERVALOS DE TIEMPO
Introduzca el número de horas del primer intervalo (positivo): 1
Introduzca el número de minutos del primer intervalo (positivo): 2
Introduzca el número de segundos del primer intervalo (positivo):
3
Introduzca el número de horas del segundo intervalo (positivo): 2
Introduzca el número de minutos del segundo intervalo (positivo):
3
Introduzca el número de segundos del segundo intervalo (positivo):
4
```

```
Los resultados de las operaciones son
Suma: 3h 5m 7s
La resta no se pudo realizar porque el segundo intervalo es mayor
que el primero
```

8. Crea una nueva versión mejorada de la clase `Fraccion` creada en el ejercicio 09 de la relación 4.03 mediante el uso de excepciones. Más concretamente deberías:
    - Modificar el constructor de `Fraccion` para que no admita cero como valor de denominador y lance una excepción `IllegalArgumentException` en ese caso.

    Se proporcionan los archivos `Actividad_4.03h_pruebas.zip` que contiene la clase `PruebaFraccion` que sirve para probar la clase y que debe generar la salida que se muestra a continuación. Asimismo se proporciona el archivo `Actividad_4.03h_doc.zip` que contiene la documentación de la clase `Fraccion`.

**Ejemplo de uso:**

```
PROGRAMA DE PRUEBA DE FRACCIONES
Introduzca el numerador de la primera fracción: 1
Introduzca el denominador de la primera fracción (no debe ser
cero): 2
Introduzca el numerador de la segunda fracción: 3
Introduzca el denominador de la segunda fracción (no debe ser
cero): 0
Error. El denominador de una fracción no puede ser cero.
Terminando
```