Programación

Bloque 04 - Desarrollo de Clases

Desarrollo de clases

Todos las clases correspondientes a los ejercicios de esta relación se deben crear dentro del paquete `prog.unidad04.actividad402`, 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. Estamos desarrollando una aplicación en la que necesitamos manipular pizzas, por lo que se desea que crees una clase para representar una pizza (que llamaremos `Pizza`). Una pizza será de un tamaño a elegir de entre dos (*mediana* o *familiar*) y un tipo a elegir de entre tres (*margarita*, *cuatro quesos* y *funghi*). Una pizza puede estar en uno de dos estados: *pedida*, cuando la pizza se ha creado pero aún no se ha servido y *servida*, cuando la pizza ya se ha servido.
   Además se desea saber el número total de pizzas que se han servido y el número total de pizzas que se han servido desde el inicio de la aplicación. Se proporcionan los archivos `Actividad_4.02a_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_4.02a_doc.zip` que contiene la documentación de la clase `Pizza`.

Ejemplo de uso:

```
PROGRAMA DE PRUEBA DE PIZZAS 2
El parámetro tamanyo no es correcto. Debe ser uno de "mediana" o
"familiar"
El parámetro tipo no es correcto. Debe ser uno de "margarita",
"cuatro quesos" o "funghi"
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
Error. Esta pizza ya se ha servido
Número de pizzas creadas: 3
Número de pizzas servidas: 1
```

2. Aumenta el ejercicio anterior añadiendo a la clase Pizza un nuevo método: coste. Este debe devolver el coste de la pizza, sabiendo que:
   - La masa de pizza mediana vale 5 euros y la de familiar 7,50.
   - Cada ingrediente tiene un coste de 1 euro
   - La pizza Margarita tiene dos ingredientes, la cuatro quesos cuatro ingredientes (vaya sorpresa) y la funghi, tres.

   Se proporcionan los archivos `Actividad_4.02b_pruebas.zip` que contiene la clase `PruebaPizza2` que sirve para probar la clase y que debe generar la salida que se muestra a continuación. Asimismo se proporciona el archivo `Actividad_4.02b_doc.zip` que contiene la documentación de la clase `Pizza`.

Ejemplo de uso:

```
PROGRAMA DE PRUEBA DE PIZZAS 3
El parámetro tamanyo no es correcto. Debe ser uno de "mediana" o
"familiar"
El parámetro tipo no es correcto. Debe ser uno de "margarita",
"cuatro quesos" o "funghi"
Pizza 1. Tamaño: mediana, Tipo: margarita, Estado: pedida, Coste:
7.0
Pizza 2. Tamaño: familiar, Tipo: funghi, Estado: servida, Coste:
10.5
Pizza 3. Tamaño: mediana, Tipo: cuatro quesos, Estado: pedida,
Coste: 9.0
Error. Esta pizza ya se ha servido
Número de pizzas creadas: 3
Número de pizzas servidas: 1
```

3. Crea una clase `Persona` con los atributos: `nombre`, `edad`, `DNI`, `sexo` (`H` / `M`), `peso` y `altura`. Los valores por defecto para todos los atributos, excepto `DNI`, será cero para las cantidades y "" para las cadenas. Para el sexo será `M`. Crea varios constructores: Uno por defecto, otro con `nombre`, `edad` y `sexo` y el resto por defecto y otro con todos los atributos, excepto `DNI`. Los métodos a implementar son:
   - `getIMC()` Devuelve el Indice de Masa Corporal de la persona. Si la altura es cero, el IMC será cero.
   - `getPesoCorrecto()`. Devuelve -1 si la persona tiene bajo peso, 0 si tiene peso normal y 1 si tiene sobrepeso. Una persona tiene el peso normal si su IMC está entre 20 y 25, ambos incluidos. Por debajo es bajo peso y por encima es sobrepeso.
   - `esMayorDeEdad()` Devuelve `true` si la persona es mayor de edad o `false` si es menor.
   - obtenerCadena() Genera una cadena con la información de la persona, en el formato que quieras.
   - `generarDNI()`. Privado. Genera el DNI de la persona calculando 8 dígitos al azar y la letra correspondiente. El método para obtener la letra a partir de los números lo puedes localizar por Internet.
   - `comprobarSexo(char sexo)`. Privado. Devuelve el sexo proporcionado, si este es correcto o el sexo por defecto (`M`) si no lo es.

   Se propociona el archivo `Actividad_4.02c_doc.zip` que contiene la documentación de la clase `Persona` y el archivo `Actividad_4.02c_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
Nombre: "Paco", Edad: 17, Sexo: H, Peso: 0.0, Altura: 0.0
Nombre: "Paqui", Edad: 26, Sexo: M, Peso: 76.6, Altura: 1.67
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
```

4. Crea una clase `Lavadora` con las siguientes características:
   - Tiene un precio base (luego se verá como se calcula el precio final a partir del base), un color (`blanco`, `negro`, `rojo`, `azul` y `gris`), consumo energético (letras entre la `A` y la `F`, ambas incluidas, peso y carga.
   - Deberá tener los siguientes constructores:
     - Por defecto (precio base = `200`, color = `blanco`, consumo energético = `F`, peso = `15` kg, carga = `5` kg)
     - Con precio y peso. El resto como por defecto.
     - Con todos los atributos.
   - Y los siguientes métodos
     - `get` de todos los atributos.
     - `getPrecioFinal()`. Devuelve el precio final de la lavadora teniendo en cuenta que según el consumo energético se añade al precio base los siguientes suplementos: A = 100, B = 80, C = 60, D = 50, E = 30, F = 10. Además por peso también se añade 10 si pesa hasta 20 kg, 50 si pesa de 20 hasta 50 kg., 80 si pesa de 50 a 80 kg. y 100 si pesa 80 ó más kg. Por último si la carga es superior a 30 kg. también se le añade un suplemento de 50 euros.
   - Añade los siguientes métodos privados
     - `comprobarConsumo(char letra)`. Devuelve la misma letra de entrada, si esta es correcta o la letra por defecto (F) si no lo es.
     - `comprobarColor(String color)`. Devuelve el mismo color de entrada, si es correcto o el color por defecto (blanco) si no lo es.

   Se propociona el archivo `Actividad_4.02i_doc.zip` que contiene la documentación de la clase `Persona` y el archivo `Actividad_4.02i_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: 0.0
Precio Base: 500.0, Color: blanco, Consumo Energético: F, Peso:
25.0, Carga Máxima: 0.0
Precio Base: 650.0, Color: rojo, Consumo Energético: A, Peso:
75.0, Carga Máxima: 0.0
Imprimimos los precios finales de las tres lavadoras
Precio Lavadora 1 = 220.0
Precio Lavadora 2 = 560.0
Precio Lavadora 3 = 830.0
```

5. Vamos a crear una clase que sirva para vender las entradas de una zona determinada de un teatro. La clase `Zona` debería tener al menos:
   - Constructor con un entero. Inicializa la zona con el número de entradas dadas

- `getEntradasPorVender`. Devuelve el número de entradas por vender.
- `vender`. Recibe un número entero positivo de entradas a comprar. Si hay entradas suficientes se compra el número indicado (que se resta de las que quedan por vender). Si no hay entradas suficientes, no se compra ninguna. En ambos casos se devuelve el número de entradas vendidas.

A partir de la clase `Zona` crea un programa (`VendeEntradas`) que venda las entradas de un teatro con tres zonas (platea, con 150 asientos, palco, con 50 asientos y atico con 100 asientos). El programa deberá mostrar un menú con tres opciones: 1.- Mostrar número de entradas por vender por zona, 2.- Vender entradas (hay que solicitar zona y número de entradas y se debe mostrar el resultado) y 3.- Salir de la aplicación. Se propociona el archivo `Actividad_4.02f_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: 145
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: 5
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: 6
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): 2

VENTA DE ENTRADAS
-----------------
Seleccione la zona en la que se localizan los asientos a vender (1
= Platea, 2 = Palco, 3 = Ático): 2
Introduzca el número de asientos a vender: 45

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: 5
Zona Palco: 5
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.....
```

6. Crea las clases `Punto`, `Triangulo`, `Circulo` y `Rectangulo`. Cada una de ellas tiene que tener las propiedades adecuadas a sus caracteristicas y el constructor adecuado para cada una, teniendo en cuenta que un triángulo se define por tres puntos, un rectángulo por dos (las coordenadas de una esquina y la opuesta) y un círculo por un punto (el centro) y una distancia (el radio). Haz que cada una de ellas tenga un método `area` que devuelva el área de la forma y `perimetro` que devuelva la longitud del perímetro. Añade un método a `Rectangulo` llamado `esCuadrado` que devuelva `true` si el rectángulo es en realidad un cuadrado (tiene los cuatro lados iguales) o `false` si no lo es. **Nota: Para calcular el área del triángulo, lo más sencillo es usar el método de Heron.**

   Se proporcionan los archivos `Actividad_4.02e_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_4.02e_doc.zip` que contiene la documentación de las clases `Punto`, `Triangulo`, `Circulo` y `Rectangulo`.

Ejemplo de uso:

```
PROGRAMA DE PRUEBA DE FIGURAS
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
```

7. La empresa Manego quiere una aplicación para gestionar sus tarjetas regalo. Vamos a crear, por lo tanto una clase `TarjetaRegalo` que tiene un saldo (siempre mayor o igual a cero) y un número identificativo de 5 dígitos que se crea automáticamente al crear una nueva tarjeta. La tarjeta debe tener los siguientes métodos, al menos:
   - `gasta(double)` al que se le pasa una cantidad de dinero a gastar. Si la tarjeta tiene saldo suficiente hay que actualizarlo y si no lo tiene se debe mostrar un mensaje de error.
   - `fusionaCon(TarjetaRegalo)` en el cual se fusionan las dos tarjetas y se obtiene una nueva tarjeta con el saldo combinado de las dos tarjetas originales (que se quedarán con un saldo de cero) y un nuevo número.
   - `obtenerCadena()` en el cual se obtiene una representación en formato cadena con la información de la tarjeta en el formato "Tarjeta nº NNNNN - Saldo DDDD€", donde NNNNN es el número de identificación de la tarjeta y DDDD es el saldo de la misma.

   Se propociona el archivo `Actividad_4.02g_doc.zip` que contiene la documentación de la clase `TarjetaRegalo` y el archivo `Actividad_4.02g_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 10443 - Saldo 100.0€
```

```
Tarjera nº 43039 - Saldo 120.0€
Hacemos algún gasto en las tarjetas
Error. No se puede gastar ese importe porque la tarjeta tiene un
saldo inferior
Después del gasto las tarjetas tienen el siguiente estado
Tarjera nº 10443 - Saldo 50.5500000000000004€
Tarjera nº 43039 - Saldo 115.0€
Fusionamos las dos tarjetas y obtenemos una nueva
Después de la fusión, las tarjetas tienen el siguiente estado
Tarjera nº 10443 - Saldo 0.0€
Tarjera nº 43039 - Saldo 0.0€
Tarjera nº 30047 - Saldo 165.55€
```

8. Crea una clase, llamada `IntervaloTiempo` que contiene, como su nombre indica, un intervalo de tiempo en horas, minutos y segundos. Debe tener un constructor que admita una cantidad de horas, minutos y segundos. También debe implementarse un método `obtenerCadena` para convertir un intervalo a una cadena de la forma "AAh BBm CCs" donde AA son las horas, BB los minutos y CC los segundos. También se deben implementar los métodos `suma` y `resta` que toman otro `IntervaloTiempo` y lo suman o restan al objeto que recibe el mensaje. y devuelven un nuevo objeto `IntervaloTiempo` con el resultado.

   Se proporcionan los archivos `Actividad_4.02d_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_4.02d_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): 2
Introduzca el número de minutos del primer intervalo (positivo): 0
Introduzca el número de segundos del primer intervalo (positivo):
59
Introduzca el número de horas del segundo intervalo (positivo): 2
Introduzca el número de minutos del segundo intervalo (positivo):
0
Introduzca el número de segundos del segundo intervalo (positivo):
1
Los resultados de las operaciones son
Suma: 4h 1m 0s
Resta: 0h 0m 58s
```

9. Crea una clase `Fraccion` que almacene una fracción. Debe soportar las operaciones aritméticas básicas (suma, resta, multiplicación y division) así como la de simplificacion, la cual reduce la fracción a la expresión mínima. Asimismo, debe tener un mensaje (`obtenerString`) para obtener la representación en una cadena de la fracción, en formado `"n / d"`, donde n es el numerador de la fracción y d el denominador.

   Se proporcionan los archivos `Actividad_4.02c_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_4.02c_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: 1
Introduzca el denominador de la segunda fracción (no debe ser
cero): 4
Los resultados de las operaciones son
Suma: 6 / 8
Resta: 2 / 8
Producto: 1 / 8
Division: 4 / 2
Los resultados simplificados de las operaciones son
Suma: 3 / 4
Resta: 1 / 4
Producto: 1 / 8
Division: 2 / 1
```