PRÁCTICA

1

(1ª parte)

Atributos

(Temperaturas)

Diseño del Software

Grado en Ingeniería Informática del Software

Curso 2023-2024

Escuela de Ingeniería Informática
Universidad de Oviedo

Raúl Izquierdo
César Acebal

Temperaturas

¿Cómo podemos convertir entre grados Celsius y Fahrenheit?

Enunciado

- Tenemos el siguiente código en el que se toman unas temperaturas de unos sensores en Celsius
- Posteriormente se utilizarán indistintamente en distintos cálculos, algunos de los cuales serán en Celsius y otros en Fahrenheit

```java
public static void main(String[] args)
{
     double[] temperaturas = new double[100];

     // Toma de datos
     for (int i = 0; i < temperaturas.length; i++)
          temperaturas[i] = leeSensorCelsius();

     // Cálculo con los datos en Celsius
     double mediaCelsius = 0;
     for (int i = 0; i < temperaturas.length; i++)
          mediaCelsius += temperaturas[i];
     mediaCelsius = mediaCelsius / temperaturas.length;
     System.out.printf("La media en Celsius es: %.1f ºFC\n", mediaCelsius);

     // Cálculo con los datos en Fahrenheit
     double mediaFahrenheit = 0;
     for (int i = 0; i < temperaturas.length; i++)
          mediaFahrenheit += temperaturas[i] * 9 / 5 + 32;
     mediaFahrenheit = mediaFahrenheit / temperaturas.length;
     System.out.printf("La media en Fahrenheit es: %.1f ºF\n", mediaFahrenheit);

     // Otro cálculo con los datos en Fahrenheit
     double varianza = 0;
     for (int i = 0; i < temperaturas.length; i++)
          varianza += Math.pow((temperaturas[i] * 9 / 5 + 32) - mediaFahrenheit, 2);
     varianza = varianza / temperaturas.length;
     System.out.printf("La varianza en Fahrenheit es: %.1f\n", varianza);
}
```

¿Cómo podríamos mejorar este diseño haciéndolo orientado a objetos?

Pista

- Se pretenden realizar en el futuro varias aplicaciones que trabajen con temperaturas (tanto en Celsius como en Fahrenheit)

Problema

- ¿Cuál era el principal defecto de ese programa?

Problema

- **¿Cuál era el principal defecto de ese programa?**
  - Que el código para convertir grados Celsius a Fahrenheit (y a la inversa, si fuera necesario) se repite en varios sitios

Problema

- **¿Cuál era el principal defecto de ese programa?**
  - Que el código para convertir grados Celsius a Fahrenheit (y a la inversa, si fuera necesario) se repite en varios sitios
- **La solución estructurada (o funcional) sería sacar ese código a una función**
  - Con un parámetro «temperatura» y que hiciese la conversión y devolviese el valor correspondiente

Problema

- **¿Cómo sería la solución orientada a objetos?**
  - Asignar esa responsabilidad a una clase
- **¿Cuál?**
  - ¿Una clase `Util`?
- **No: la propia clase `Temperatura`**
  - Es la que debería saber cómo convertirse a sí misma

(Al menos de momento y en este ejemplo tan simple.)

Ejercicio

- Diseñar e implementar una clase que represente temperaturas y se encargue de convertir de una a otra unidad
- Modificar el código del programa principal para que utilice la nueva clase

Añadamos una nueva funcionalidad...

Ampliación

- Ahora mismo se puede pedir la temperatura, pero no cambiarla
- Se quiere poder cambiar el valor (los objetos serán mutables) e indicar la nueva temperatura tanto en Celsius como en Fahrenheit

¿Una cosa así?

Responsabilidades

- `setCelsius(value: double)`
- `setFahrenheit(value: double)`
- `asCelsius(): double`
- `asFahrenheit(): doble`

Es decir, la clase va a permitir modificar el valor de cada objeto temperatura, indicándolo o bien en Celsius o bien en Fahrenheit. Posteriormente, a través de los métodos asCelsius y asFahrenheit se puede obtener el valor numérico de dicha temperatura en Celsius o Fahrenheit indistintamente.

Implementadlo...

Posibles soluciones

- **Seguramente lo habréis hecho de varias formas distintas**
- **Aquí analizaremos tres posibilidades**
  - Aunque habría muchas más
  - Nótese que no nos estamos centrando en obtener un diseño orientado a objetos fetén para el problema genérico de conversión de cantidades con unidades

    El objetivo del ejercicio es mucho más mundano: aclarar de una vez por todas el papel que desempeñan los atributos en el diseño orientado a objetos.

Primera opción

- Con un único atributo, guardando internamente el valor siempre en una unidad (por ejemplo, grados Celsius) y convirtiéndolo cuando sea necesario

```
public class Temperature
{
     private double celsius;

     public Temperature(double value)
     {
          this.celsius = value;
     }

     public void setCelsius(double value)
     {
          this.celsius = value;
     }

     public void setFahrenheit(double value)
     {
          this.celsius = (value - 32) * 5 / 9;
     }

     public double asCelsius()
     {
          return celsius;
     }

     public double asFahrenheit()
     {
          return celsius * 9 / 5 + 32;
     }
}
```

Primera opción: un único atributo de tipo Celsius (por ejemplo) y convertimos sólo cuando sea necesario: al asignar, evidentemente, un valor ya en Fahrenheit y al llamar al método toCelsius.

¿Qué os parece?

Poca ocupación de memoria, pero lento

- **Ventaja**
  - Ocupación de memoria mínima
- **Inconveniente**
  - Supóngase que se toman 1.000 medidas en Fahrenheit y sólo se usan en Fahrenheit
  - Se estarían realizando todas las conversiones continuamente de forma innecesaria

Segunda opción

- Dos atributos, permanentemente actualizados

```
public class Temperature
{
     private double celsius;
     private double fahrenheit;

     public Temperature(double value)
     {
          setCelsius(value);
     }

     public void setCelsius(double value)
     {
          this.celsius = value;
          this.fahrenheit = celsius * 9 / 5 + 32;
     }

     public void setFahrenheit(double value)
     {
          this.fahrenheit = value;
          this.celsius = (value - 32) * 5 / 9;
     }

     public double asCelsius()
     {
          return celsius;
     }

     public double asFahrenheit()
     {
          return fahrenheit;
     }
}
```

Segunda opción: dos atributos, uno para cada unidad, que están permanente actualizados (sincronizados): cada vez que se cambia la temperatura, se modifican los dos para que representen el mismo valor.

¿Qué os parece?

¿Más rápido?, pero más memoria

- **Ventaja**
  - Velocidad: una vez establecido el valor, ya no se realizan más conversiones
- **Inconvenientes**
  - Más ocupación de memoria: dos atributos
  - Velocidad: si se toman 1.000 temperaturas en Celsius que sólo se van a leer en dicha unidad, se estarían haciendo otras tantas conversiones a Fahrenheit para nada

Tercera opción

- Dos atributos, pero convirtiendo sólo a petición (es decir, cuando sea necesario)

```
public class Temperature
{
     private double celsius;
     private double fahrenheit;

     private boolean celsiusIsValid;
     private boolean fahrenheitIsValid;

     public Temperature(double value)
     {
          setCelsius(value);
     }

     public void setCelsius(double value)
     {
          this.celsius = value;
          celsiusIsValid = true;
          fahrenheitIsValid = false;
     }

     public void setFahrenheit(double value)
     {
          this.fahrenheit = value;
          fahrenheitIsValid = true;
          celsiusIsValid = false;
     }
```

Tercera opción: dos atributos, que se convierten sólo cuando es necesario.

```
     public double asCelsius()
     {
          if (!celsiusIsValid) {
               assert fahrenheitIsValid;
               setCelsius(fahrenheit);
          }
          return celsius;
     }

     public double asFahrenheit()
     {
          if (!fahrenheitIsValid) {
               assert celsiusIsValid;
               setFahrenheit(celsius);
          }
          return fahrenheit;
     }
}
```

¿Qué os parece?

El más rápido, el que más memoria necesita

- **Ventaja**
  - El más eficiente desde el punto de vista de la velocidad: sólo se realizan las conversiones estrictamente necesarias
- **Inconveniente**
  - El que más memoria consume: cuatro atributos

Conclusiones

¿Cuál es mejor?

Depende

- **Lo importante es que el diseño elegido permite cambiar la implementación a lo que sea mejor en cada caso… ¡sin que tengamos que tocar en veinte sitios distintos!**
  - Podemos empezar por la primera (la más sencilla y rápida) y cambiarla cuando sepamos que tenemos problema o queremos más eficiencia en un aspecto u otro (memoria, velocidad…)
- **¿Cuándo sería necesario tocar el resto del programa?**
  - Cuando hagamos cambios en la signatura de alguno de los métodos de `Temperatura`, que son los que utilizan otras clases (de ahí la importancia de pensarlos bien)

De eso trata esta asignatura: de determinar las clases y métodos adecuados (no su implementación)

Conclusiones

- **En esto consiste diseñar: decidir los métodos y parámetros adecuados para que sea flexible (y no haya que cambiarlos)**
- **La implementación (programar, lo que hacíais otros años) siempre se podrá cambiar y mejorar**
- **En nuestro ejemplo, tenemos tres implementaciones y no hemos tenido que tocar el** main

Conclusiones

- **Un buen programador puede obtener un programa eficiente, pero si no es buen diseñador será difícil adaptarse a los cambios**
  - (que los habrá)
- **Un buen diseñador permitirá llegar a un programa eficiente**
  - Además, permitirá un prototipado rápido

¿Y los atributos?

- **Cada versión los cambiaba**
- **Tienen un papel secundario en el diseño: están supeditados a la implementación**
  - Serán aquellos que necesite cada una, son irrelevantes para los clientes de la clase
- **¿Cómo será el diseño resultante si al empezar a diseñar una clase se piensa primero en sus atributos?**

Los atributos están supeditados a la implementación. Nunca se debe diseñar una clase pensando en sus atributos, sino en los métodos que debería tener y sus parámetros.