CLEAN CODE

Entornos de Desarrollo

IES Alonso de Avellaneda
DAM 1 diurno

Martín Tadeo, María

Contenido

Código Original

Main

```java
package cleancodeev1;

public class CleanCodeEv1 {
    public static void main(String[] args) {
        Coche coche = new Coche();
        Coche coche1 = new Coche("BMW", "negro", 430.47f, 0.84f, 6.5f);
        Coche coche2 = new Coche("Mercedes", "gris", 150.95f, 1.02f,6.5f);
        Coche coche3 = new Coche("VW", "blanco", 93.62f, 0.65f, 6.5f);
        Coche coche4 = new Coche("Fiat", "rojo", 694.22f, 0.58f, 6.5f);

        System.out.printf("COCHE0 = Modelo: %-10s Color: %-10s Número de Kilómetros ida: %-10s Precio combustible: %s \n"
                , coche.modelo, coche.color, coche.parametro(coche.nroKilometros), coche.parametro(coche.precioGasolina));
        System.out.printf("Distancia total recorrida: %-10s Precio total de la gasolina: %-10s \n\n"
                , coche.parametro(coche.nroKilometros * 2), coche.parametro(coche.Viaje(coche)));

        System.out.printf("COCHE1 = Modelo: %-10s Color: %-10s Número de Kilómetros ida: %-10s Precio combustible: %s \n"
                , coche1.modelo, coche1.color, coche.parametro(coche1.nroKilometros), coche.parametro(coche1.precioGasolina));
        System.out.printf("Distancia total recorrida: %-10s Precio total de la gasolina: %-10s \n\n"
                , coche.parametro(coche1.nroKilometros * 2), coche1.parametro(coche.Viaje(coche1)));

        System.out.printf("COCHE2 = Modelo: %-10s Color: %-10s Número de Kilómetros ida: %-10s Precio combustible: %s \n"
                , coche2.modelo, coche2.color, coche.parametro(coche2.nroKilometros), coche.parametro(coche2.precioGasolina));
        System.out.printf("Distancia total recorrida: %-10s Precio total de la gasolina: %-10s \n\n"
                , coche.parametro(coche2.nroKilometros * 2), coche2.parametro(coche.Viaje(coche2)));

        System.out.printf("COCHE3 = Modelo: %-10s Color: %-10s Número de Kilómetros ida: %-10s Precio combustible: %s \n"
                , coche3.modelo, coche3.color, coche.parametro(coche3.nroKilometros), coche.parametro(coche3.precioGasolina));
        System.out.printf("Distancia total recorrida: %-10s Precio total de la gasolina: %-10s \n\n"
                , coche.parametro(coche3.nroKilometros * 2), coche.parametro(coche1.Viaje(coche3)));

        System.out.printf("COCHE4 = Modelo: %-10s Color: %-10s Número de Kilómetros ida: %-10s Precio combustible: %s\n"
                , coche4.modelo, coche4.color, coche.parametro(coche4.nroKilometros), coche.parametro(coche4.precioGasolina));
        System.out.printf("Distancia total recorrida: %-10s Precio total de la gasolina: %-10s "
                , coche.parametro(coche4.nroKilometros * 2), coche.parametro(coche.Viaje(coche4)));
    }
}
```

Clase Coche

```java
package cleancodeev1;

public class Coche {
    String modelo;
    String color;
    float nroKilometros;
    float precioGasolina;
    float consumo;

    public Coche(String modelo, String color, float nroKilometros, float precioGasolina, float consumo) {
        this.modelo = modelo;
        this.color = color;
        this.nroKilometros = nroKilometros;
        this.precioGasolina = precioGasolina;
        this.consumo = consumo;
    }

    public Coche(){
        modelo = "";
        color = "";
        nroKilometros = 0f;
        precioGasolina = 0f;
        consumo = 0f;
    }

    public String getModelo() {
        return modelo;
    }
    public void setModelo(String modelo) {
        this.modelo = modelo;
    }

    public String getColor() {
        return color;
    }
```

```java
public void setColor(String color) {
    this.color = color;
}
public float getNroKilometros() {
    return nroKilometros;
}
public void setNroKilometros(float nroKilometros) {
    this.nroKilometros = nroKilometros;
}
public float getPrecioGasolina() {
    return precioGasolina;
}
public void setPrecioGasolina(float precioGasolina) {
    this.precioGasolina = precioGasolina;
}
public float getConsumo() {
    return consumo;
}
public void setConsumo(float consumo) {
    this.consumo = consumo;
}

public String toString() {
    return "Coche{" + "modelo=" + modelo + ", color=" + color + ", nroKilometros=" + nroKilometros +
            ", precioGasolina=" + precioGasolina + ", consumo=" + consumo + '}';
}

public String parametro(double parametro){
    String resultado = " " + parametro;
    return resultado;
}

public double Viaje(Coche vehiculo){
    float viajeIda = (vehiculo.nroKilometros * 2 / vehiculo.consumo) * vehiculo.precioGasolina;
    return viajeIda;
}
```

BLOQUE 1: NOMBRES

Con significado

Los nombres que utilizamos son importantes para orientarnos de la utilidad de las funciones, atributos, objetos, métodos… y es por eso que debemos realizar ciertos cambios en el código para cumplir así con este requisito.

```java
public double Viaje(Coche vehiculo){
    float viajeIda = (vehiculo.nroKilometros * 2 / vehiculo.consumo) * vehiculo.precioGasolina;
    return viajeIda;
}

public double gastosGasolinaViaje(Coche vehiculo){
    float costeTotal= (vehiculo.nroKilometros * 2 / vehiculo.consumo) * vehiculo.precioGasolina;
    return costeTotal;
}
```

Clase Coche

```java
Coche coche = new Coche();
Coche coche1 = new Coche("BMW", "negro", 430.47f, 0.84f, 6.5f);
Coche coche2 = new Coche("Mercedes", "gris", 150.95f, 1.02f,6.5f);
Coche coche3 = new Coche("VW", "blanco", 93.62f, 0.65f, 6.5f);
Coche coche4 = new Coche("Fiat", "rojo", 694.22f, 0.58f, 6.5f);
```

```
Coche cocheVacio = new Coche();
Coche primerCoche = new Coche("BMW", "negro", 430.47f, 0.84f, 6.5f);
Coche segundoCoche = new Coche("Mercedes", "gris", 150.95f, 1.02f, 6.5f);
Coche tercerCoche = new Coche("VW", "blanco", 93.62f, 0.65f, 6.5f);
Coche cuartoCoche = new Coche("Fiat", "rojo", 694.22f, 0.58f, 6.5f);
```

Main

Fáciles de pronunciar

En nuestro caso los nombres tanto de los atributos como de las funciones de todo el código están escritas en castellano ya que así nos facilita a nivel personal su pronunciación y su entendimiento. Esta característica debe adaptarse también dependiendo del idioma en el que trabajemos y desarrollemos el programa al igual que en el entorno profesional en el que nos encontremos (tecnicismos, jerga…)

Elige una sola palabra por concepto

Algunos nombres de atributos o funciones deben ser compuestos para una mayor aclaración y mejor entendimiento ya que con una única clave no serían suficientemente descriptivos como para poder entender fácilmente de que se tratan o cuál es su cometido. Ej: `cocheVacio`, `precioGasilina`…

```
Coche cocheVacio = new Coche();
```

Main

```
float precioGasolina;
```

Clase Coche

Bloque 2: Funciones

Las funciones deben ser pequeñas Haz una única cosa

Las funciones que hay en el código únicamente realizan una función por lo que automáticamente al no ser muy complejas son pequeñas y no tienen una excesiva cantidad de código. Incluso se han incluido nuevas para así simplificar el código haciéndolo más claro, pudiéndolo reutilizar evitando a su vez la repetición de código innecesario economizando y agilizando su creación. Varios ejemplos claros con la función **`imprimirCoche`** y **`gastosGasolinaViaje`**.

Don't repeat Yourself (No te repitas)

Uno de los ejemplos de mejora de nuestro código original y que así se asemeje más a la estructura del clean code es evitar la redundancia. Se puede ver claramente en nuestra función **`parametro`** que se encuentra en nuestra clase **`Coche`**

```
public String parametro(double parametro){
    String resultado = " " + parametro;
    return resultado;
}
```

Clase Coche

```
System.out.printf("COCHE1 = Modelo: %-10s Color: %-10s Número de Kilómetros ida: %-10s Precio combustible: %s \n"
        , coche1.modelo, coche1.color, coche.parametro(coche1.nroKilometros), coche.parametro(coche1.precioGasolina));
System.out.printf("Distancia total recorrida: %-10s Precio total de la gasolina: %-10s \n\n"
        , coche.parametro(coche1.nroKilometros * 2), coche1.parametro(coche.Viaje(coche1)));
```

Main

Esta función únicamente transforma un parámetro, del tipo que sea, a tipo `String` lo cual no aporta gran funcionalidad al programa, sino que genera una mayor complicación a la hora de ser utilizado en el `main` pudiéndose asi simplificar e incluso aportando claridad al código prescindiendo de él.

Otra de las grandes modificaciones para evitar también la repetición es la creación de un método llamado **`imprimirCoche`** que evita así escribir constantemente todos y cada uno de los atributos con sus formatos correspondientes de las clases creadas para visualizarlas por pantalla.

```
public void imprimirCoche(Coche coche){
    System.out.printf("CARACTERISTICAS: \n\tModelo: %26s \n\tColor: %27s \n\tNumero de Kilometros ida: %8s \n\tPrecio combustible: %14s \n"
            , coche.modelo, coche.color, String.valueOf(coche.nroKilometros), String.valueOf(coche.precioGasolina));
}
```

Clase Coche

```
cocheVacio.imprimirCoche(primerCoche);
```

Main

También se puede ver con el método **`gastosGasolinaViaje`** el cual se utiliza múltiples veces a lo largo del código.

```
public double gastosGasolinaViaje(Coche vehiculo){
    float costeTotal= (vehiculo.nroKilometros * 2 / vehiculo.consumo) * vehiculo.precioGasolina;
    return costeTotal;
}
```

Automáticamente disponer de estas nuevas funciones cambia por completo la extensión de nuestro código en el archivo `main` mejorando notablemente su aspecto y eficacia.

```
cocheVacio.imprimirCoche(primerCoche);
System.out.printf("Gato total del viaje: %20.2f \n", cocheVacio.gastosGasolinaViaje(primerCoche));

cocheVacio.imprimirCoche(segundoCoche);
System.out.printf("Gato total del viaje: %20.2f \n", cocheVacio.gastosGasolinaViaje(segundoCoche));

cocheVacio.imprimirCoche(tercerCoche);
System.out.printf("Gato total del viaje: %20.2f \n", cocheVacio.gastosGasolinaViaje(tercerCoche));

cocheVacio.imprimirCoche(cuartoCoche);
System.out.printf("Gato total del viaje: %20.2f \n", cocheVacio.gastosGasolinaViaje(cuartoCoche));
```

Main

Bloque 3: Comentarios

Usa código autoexplicativo

A lo largo de todo el código procuramos dejar lo más claro posible qué es lo que ocurre en cada momento ahorrándonos así algunos comentarios innecesarios, mientras que hay veces que es necesario recurrir a ellos para evitar así una posible confusión o duda.

A veces los cometarios son necesarios

Hay casos en los que los comentarios no son necesarios como hemos dicho anteriormente, pero en otros si que se requieren como apoyo y aclarar cualquier duda posible. A continuación, se mostrará un ejemplo donde se ven los dos casos, pero a lo largo del programa ha sido necesario completar el código con comentarios como aclaración.

```
public String toString() {
    return "Coche{" + "modelo=" + modelo + ", color=" + color + ", nroKilometros=" + nroKilometros +
            ", precioGasolina=" + precioGasolina + ", consumo=" + consumo + '}';
}
//Método para calcular el coste total de la gasolina de un coche
public double gastosGasolinaViaje(Coche vehiculo){
    float costeTotal= (vehiculo.nroKilometros * 2 / vehiculo.consumo) * vehiculo.precioGasolina;
    return costeTotal;
}
//Método para imprimir los atributos de un Objeto de la clase
public void imprimirCoche(Coche coche){
    System.out.printf("CARACTERISTICAS: \n\tModelo: %26s \n\tColor: %27s \n\tNumero de Kilometros ida: %8s \n\tPrecio combustible: %14s \n"
            , coche.modelo, coche.color, String.valueOf(coche.nroKilometros), String.valueOf(coche.precioGasolina));
}
```

Clase Coche

Los comentarios dicen qué hace el código, no como lo hacen

Otro ejemplo en el que podemos ver la utilidad de los comentarios es en este caso en el que únicamente se nombra la función de las líneas de código, pero no su funcionamiento interno ya que esa no es la finalidad principal al hacer uso de este recurso.

```
//Cosntructor de la clase
public Coche(String modelo, String color, float nroKilometros, float precioGasolina, float consumo) {
    this.modelo = modelo;
    this.color = color;
    this.nroKilometros = nroKilometros;
    this.precioGasolina = precioGasolina;
    this.consumo = consumo;
}
//Cosntructor vacío de la clase
public Coche(){
    modelo = "";
    color = "";
    nroKilometros = 0f;
    precioGasolina = 0f;
    consumo = 0f;
}
```

Clase coche

Código Final: Clean Code

Main

```java
package cleancodeev1;

public class CleanCodeEv1 {
    public static void main(String[] args) {
        //Creación de un Objeto Coche vacío
        Coche cocheVacio = new Coche();
        //Creación de Objetos Coche con atributos predeterminados
        Coche primerCoche = new Coche("BMW", "negro", 430.47f, 0.84f, 6.5f);
        Coche segundoCoche = new Coche("Mercedes", "gris", 150.95f, 1.02f, 6.5f);
        Coche tercerCoche = new Coche("VW", "blanco", 93.62f, 0.65f, 6.5f);
        Coche cuartoCoche = new Coche("Fiat", "rojo", 694.22f, 0.58f, 6.5f);

        //Impresión de los atributos de cada Objeto coche y el coste del viaje completo de cada uno
            //Primer Orbjeto coche
        cocheVacio.imprimirCoche(primerCoche);
        System.out.printf("Gato total del viaje: %20.2f \n", cocheVacio.gastosGasolinaViaje(primerCoche));
            //Segundo Orbjeto coche
        cocheVacio.imprimirCoche(segundoCoche);
        System.out.printf("Gato total del viaje: %20.2f \n", cocheVacio.gastosGasolinaViaje(segundoCoche));
            //Tercer Orbjeto coche
        cocheVacio.imprimirCoche(tercerCoche);
        System.out.printf("Gato total del viaje: %20.2f \n", cocheVacio.gastosGasolinaViaje(tercerCoche));
            //Cuarto Orbjeto coche
        cocheVacio.imprimirCoche(cuartoCoche);
        System.out.printf("Gato total del viaje: %20.2f \n", cocheVacio.gastosGasolinaViaje(cuartoCoche));

    }
}
```

Clase Coche

```java
package cleancodeev1;

public class Coche {
    //Atributos de la clase
    String modelo;
    String color;
    float nroKilometros;
    float precioGasolina;
    float consumo;

    //Cosntructor de la clase
    public Coche(String modelo, String color, float nroKilometros, float precioGasolina, float consumo) {
        this.modelo = modelo;
        this.color = color;
        this.nroKilometros = nroKilometros;
        this.precioGasolina = precioGasolina;
        this.consumo = consumo;
    }
    //Cosntructor vacío de la clase
    public Coche(){
        modelo = "";
        color = "";
        nroKilometros = 0f;
        precioGasolina = 0f;
        consumo = 0f;
    }
    //Getter y Setter de los atributos de la clase
    public String getModelo() {
        return modelo;
    }
    public void setModelo(String modelo) {
        this.modelo = modelo;
    }
```

```
public String getColor() {
    return color;
}
public void setColor(String color) {
    this.color = color;
}
public float getNroKilometros() {
    return nroKilometros;
}
public void setNroKilometros(float nroKilometros) {
    this.nroKilometros = nroKilometros;
}
public float getPrecioGasolina() {
    return precioGasolina;
}
public void setPrecioGasolina(float precioGasolina) {
    this.precioGasolina = precioGasolina;
}
public float getConsumo() {
    return consumo;
}
public void setConsumo(float consumo) {
    this.consumo = consumo;
}
public String toString() {
    return "Coche{" + "modelo=" + modelo + ", color=" + color + ", nroKilometros=" + nroKilometros +
            ", precioGasolina=" + precioGasolina + ", consumo=" + consumo + '}';
}
//Método para calcular el coste total de la gasolina de un coche
public double gastosGasolinaViaje(Coche vehiculo){
    float costeTotal= (vehiculo.nroKilometros * 2 / vehiculo.consumo) * vehiculo.precioGasolina;
    return costeTotal;
}
```

```
//Método para imprimir los atributos de un Objeto de la clase
public void imprimirCoche(Coche coche){
    System.out.printf("CARACTERISTICAS: \n\tModelo: %26s \n\tColor: %27s \n\tNumero de Kilometros ida: %8s \n\tPrecio combustible: %14s \n"
            , coche.modelo, coche.color, String.valueOf(coche.nroKilometros), String.valueOf(coche.precioGasolina));
}
```