

PRÁCTICA DE N°5 DE SIS103

Nombre: Gonzales Suyo Franz Reinaldo

C.U. 111-500

Carrera: Ing. en Ciencias de la Computación

1: Ejercicio de Estado de un objeto

Class Automóvil:

```
/**
 * Esta clase define objetos de tipo Automóvil con una marca, modelo,
 * motor, tipo de combustible, tipo de automóvil, número de puertas,
 * cantidad de asientos, velocidad máxima, color y velocidad actual.
 * @version 1.2/2020
 */
public class Automovil {
    // Atributo que define la marca de un automóvil

    String marca;
    // Atributo que define el modelo de un automóvil
    int modelo;
    // Atributo que define el motor de un automóvil
    int motor;
    // Tipo de combustible como un valor enumerado
    enum tipoCom {GASOLINA, BIOETANOL, DIESEL, BIODIESEL, GAS_NATURAL}
    // Atributo que define el tipo de combustible
    tipoCom tipoCombustible;
    // Tipo de automóvil como un valor enumerado
    enum tipoA {CIUDAD, SUBCOMPACTO, COMPACTO, FAMILIAR, EJECUTIVO, SUV}
    // Atributo que define el tipo de automóvil
    tipoA tipoAutomóvil;
    // Atributo que define el número de puertas de un automóvil
    int numeroPuertas;
    // Atributo que define la cantidad de asientos de un automóvil
    int cantidadAsientos;
    // Atributo que define la velocidad máxima de un automóvil
    int velocidadMaxima;
    // Color del automóvil como un valor enumerado
    enum tipoColor {BLANCO, NEGRO, ROJO, NARANJA, AMARILLO, VERDE, AZUL,
VIOLETA}
    // Atributo que define el color de un automóvil
    tipoColor color;
    // Atributo que define la velocidad de un automóvil
    int velocidadActual = 0;

    /*
    * Constructor de la clase Automóvil
```

```

* @param marca Parámetro que define la marca de un automóvil
* @param modelo Parámetro que define el modelo (año de
*fabricación) de un automóvil
* @param motor Parámetro que define el volumen del cilindraje del
* motor (puede ser gasolina, bioetanol, diésel, biodiesel o gas natural)
* @param tipoAutomóvil Parámetro que define el tipo de automóvil
* (puede ser Carro de ciudad, Subcompacto, Compacto, Familiar,
* Ejecutivo o SUV)
* @param numeroPuertas Parámetro que define el número de
* puertas de un automóvil

* @param cantidadAsientos Parámetro que define la cantidad de
* asientos que tiene el automóvil
* @param velocidadMaxima Parámetro que define la velocidad
* máxima permitida al automóvil
* @param color Parámetro que define el color del automóvil (puede
* ser Blanco, Negro, Rojo, Naranja, Amarillo, Verde, Azul o Violeta)
*/

```

```

Automovil(String marca, int modelo, int motor, tipoCom tipoCombustible,
tipoA tipoAutomóvil, int numeroPuertas, int
    cantidadAsientos, int velocidadMaxima, tipoColor color) {
    this.marca = marca;
    this.modelo = modelo;
    this.motor = motor;
    this.tipoCombustible = tipoCombustible;
    this.tipoAutomóvil = tipoAutomóvil;
    this.numeroPuertas = numeroPuertas;
    this.cantidadAsientos = cantidadAsientos;
    this.velocidadMaxima = velocidadMaxima;
    this.color = color;
}

```

```

/*
* Método que devuelve la marca de un automóvil
* @return La marca de un automóvil
*/

```

```

String getMarca() {
    return marca;
}

```

```

/*
* Método que devuelve el modelo de un automóvil
* @return El modelo de un automóvil
*/

```

```

int getModelo() {
    return modelo;
}

```

```

/*

```

```

* Método que devuelve el volumen en litros del cilindraje del motor
* de un automóvil
* @return El volumen en litros del cilindraje del motor de un
* automóvil
*/
int getMotor() {
    return motor;
}

/*
* Método que devuelve el tipo de combustible utilizado por el motor
* de un automóvil
* @return El tipo de combustible utilizado por el motor de un
* automóvil
*/
tipoCom getTipoCombustible() {
    return tipoCombustible;
}

/*
* Método que devuelve el tipo de automóvil
* @return El tipo de automóvil
*/
tipoA getTipoAutomóvil() {
    return tipoAutomóvil;
}

/*
* Método que devuelve el número de puertas de un automóvil
* @return El número de puertas que tiene un automóvil
*/
int getNumeroPuertas() {
    return numeroPuertas;
}

/*
* Método que devuelve la cantidad de asientos de un automóvil
* @return La cantidad de asientos que tiene un automóvil
*/
int getCantidadAsientos() {
    return cantidadAsientos;
}

/*
* Método que devuelve la velocidad máxima de un automóvil
* @return La velocidad máxima de un automóvil
*/
int getVelocidadMaxima() {
    return velocidadMaxima;
}

```

```

/*
 * Método que devuelve el color de un automóvil
 * @return El color de un automóvil
 */
tipoColor getColor() {
    return color;
}

/*
 * Método que devuelve la velocidad actual de un automóvil
 * @return La velocidad actual de un automóvil
 */
int getVelocidadActual() {
    return velocidadActual;
}

/*
 * Método que establece la marca de un automóvil
 * @param marca Parámetro que define la marca de un automóvil
 */
void setMarca(String marca) {
    this.marca = marca;
}

/*
 * Método que establece el modelo de un automóvil
 * @param modelo Parámetro que define el modelo de un automóvil
 */
void setModelo(int modelo) {
    this.modelo = modelo;
}

/*
 * Método que establece el volumen en litros del motor de un automóvil
 * @param motor Parámetro que define el volumen en litros del
 * motor de un automóvil
 */
void setMotor(int motor) {
    this.motor = motor;
}

void setVelocidadMaxima(int velocidadMaxima) {
    this.velocidadMaxima = velocidadMaxima;
}

/*
 * Método que establece el color de un automóvil
 * @param color Parámetro que define el color de un automóvil
 */
void setColor(tipoColor color) {

```

```

        this.color = color;
    }

    /**
     * Método que establece la velocidad de un automóvil
     * @param velocidadActual Parámetro que define la velocidad actual
     * de un automóvil
     */
    void setVelocidadActual(int velocidadActual) {
        this.velocidadActual = velocidadActual;
    }

    /**
     * Método que incrementa la velocidad de un automóvil
     * @param incrementoVelocidad Parámetro que define la cantidad a
     * incrementar en la velocidad actual de un automóvil
     */

    void acelerar(int incrementoVelocidad) {
        if (velocidadActual + incrementoVelocidad < velocidadMaxima) {
            /* Si el incremento de velocidad no supera la velocidad máxima
*/
            velocidadActual = velocidadActual + incrementoVelocidad;
        } else {

            /* De otra manera no se puede incrementar la velocidad y se
genera mensaje */
            System.out.println("No se puede incrementar a una velocidad
superior a la máxima del automóvil.");
        }
    }

    /**
     * Método que decrementa la velocidad de un automóvil
     * @param marca Parámetro que define la cantidad a decrementar en
     * la velocidad actual de un automovil
     */

    void desacelerar(int decrementoVelocidad) {
        /* La velocidad actual no se puede decrementar alcanzando un valor
negativo */
        if ((velocidadActual - decrementoVelocidad) > 0) {
            velocidadActual = velocidadActual - decrementoVelocidad;
        } else {
            /* De otra manera no se puede decrementar la velocidad y se
genera mensaje */
            System.out.println("No se puede decrementar a una velocidad
negativa.");
        }
    }

```

```

    }

    /**
     * Método que coloca la velocidad actual de un automóvil en cero
     */
    void frenar() {
        velocidadActual = 0;
    }

    /**
     * Método que calcula el tiempo que tarda un automóvil en recorrer
     * cierta distancia
     * @param distancia Parámetro que define la distancia a recorrer por
     * el automóvil (en kilómetros)
     */
    double calcularTiempoLlegada(int distancia) {
        return distancia/velocidadActual;
    }

    /**
     * Método que imprime en pantalla los valores de los atributos de un
     * automóvil
     */
    void imprimir() {
        System.out.println("Marca = " + marca);
        System.out.println("Modelo = " + modelo);
        System.out.println("Motor = " + motor);
        System.out.println("Tipo de combustible = " + tipoCombustible);
        System.out.println("Tipo de automóvil = " + tipoAutomóvil);
        System.out.println("Número de puertas = " + numeroPuertas);

        System.out.println("Cantidad de asientos = " +
cantidadAsientos);
        System.out.println("Velocida máxima = " + velocidadMaxima);
        System.out.println("Color = " + color);
    }

    /**
     * Método main que crea un automóvil, imprime sus datos en
     * pantalla y realiza varios cambios en su velocidad
     */
    public static void main(String args[]) {

        Automovil auto1 =
new Automovil("Ford",2018,3,tipoCom.DIESEL,tipoA.EJECUTIVO,5,6,250,tipoColo
r.NEGRO);
        auto1.imprimir();
        auto1.setVelocidadActual(100);
        System.out.println("Velocidad actual = " + auto1.velocidadActual);
        auto1.acelearar(20);
    }
}

```

```

        System.out.println("Velocidad actual = " + auto1.velocidadActual);
        auto1.desacelerar(50);
        System.out.println("Velocidad actual = " + auto1.velocidadActual);
        auto1.frenar();
        System.out.println("Velocidad actual = " + auto1.velocidadActual);
        auto1.desacelerar(20);
    }
}

```

Prueba de ejecución:

```

C:\Windows\System32> cd C:\GONZALES\SIS103\Clases_Objeto\Estado_objeto

C:\GONZALES\SIS103\Clases_Objeto\Estado_objeto> javac Automovil.java

C:\GONZALES\SIS103\Clases_Objeto\Estado_objeto> java Automovil
Marca = Ford
Modelo = 2018
Motor = 3
Tipo de combustible = DIESEL
Tipo de automóvil = EJECUTIVO
Número de puertas = 5
Cantidad de asientos = 6
Velocidad máxima = 250
Color = NEGRO
Velocidad actual = 100
Velocidad actual = 120
Velocidad actual = 70
Velocidad actual = 0
No se puede decrementar a una velocidad negativa.

C:\GONZALES\SIS103\Clases_Objeto\Estado_objeto>

```

2: Ejercicios propuestos del ejercicio

Ejercicios propuestos

- Agregar a la clase Automóvil, un atributo para determinar si el vehículo es automático o no. Agregar los métodos get y set para dicho atributo.
- Modificar el constructor para inicializar dicho atributo. u Modificar el método acelerar para que si la velocidad máxima se sobrepase se genere una multa. Dicha multa se puede incrementar cada vez que el vehículo intenta superar la velocidad máxima permitida.

- Agregar un método para determinar si un vehículo tiene multas y otro método para determinar el valor total de multas de un vehículo.

```
/**
 * Esta clase define objetos de tipo Automóvil con una marca, modelo,
 * motor, tipo de combustible, tipo de automóvil, número de puertas,
 * cantidad de asientos, velocidad máxima, color y velocidad actual.
 * @version 1.2/2020
 */
public class Automovil {
    // Atributo que define la marca de un automóvil

    String marca;
    // Atributo que define el modelo de un automóvil
    int modelo;
    // Atributo que define el motor de un automóvil
    int motor;
    // Tipo de combustible como un valor enumerado
    enum tipoCom {GASOLINA, BIOETANOL, DIESEL, BIODIESEL, GAS_NATURAL}
    // Atributo que define el tipo de combustible
    tipoCom tipoCombustible;
    // Tipo de automóvil como un valor enumerado
    enum tipoA {CIUDAD, SUBCOMPACTO, COMPACTO, FAMILIAR, EJECUTIVO, SUV}
    // Atributo que define el tipo de automóvil
    tipoA tipoAutomóvil;
    // Atributo que define el número de puertas de un automóvil
    int numeroPuertas;
    // Atributo que define la cantidad de asientos de un automóvil
    int cantidadAsientos;
    // Atributo que define la velocidad máxima de un automóvil
    int velocidadMaxima;
    // Color del automóvil como un valor enumerado
    enum tipoColor {BLANCO, NEGRO, ROJO, NARANJA, AMARILLO, VERDE, AZUL,
VIOLETA}
    // Atributo que define el color de un automóvil
    tipoColor color;
    // Atributo que define la velocidad de un automóvil
    int velocidadActual = 0;

    // Agregamos un nuevo atributo si el vehículo es automatico o no
    boolean esAutomatico = false;

    int numero_multas = 0;

    int multas = 0;

    /*
     * Constructor de la clase Automóvil
     * @param marca Parámetro que define la marca de un automóvil
     * @param modelo Parámetro que define el modelo (año de
     * fabricación) de un automóvil
     * @param motor Parámetro que define el volumen del cilindraje del
     * motor (puede ser gasolina, bioetanol, diésel, biodiesel o gas natural)
     * @param tipoAutomóvil Parámetro que define el tipo de automóvil
     * (puede ser Carro de ciudad, Subcompacto, Compacto, Familiar,
     * Ejecutivo o SUV)
     * @param numeroPuertas Parámetro que define el número de
```



```

> * puertas de un automóvil
>
> * @param cantidadAsientos Parámetro que define la cantidad de
> * asientos que tiene el automóvil
> * @param velocidadMaxima Parámetro que define la velocidad
> * máxima permitida al automóvil
> * @param color Parámetro que define el color del automóvil (puede
> * ser Blanco, Negro, Rojo, Naranja, Amarillo, Verde, Azul o Violeta)
> */
>
> Automovil(String marca, int modelo, int motor, tipoCom tipoCombustible,
> tipoA tipoAutomóvil, int numeroPuertas, int
> cantidadAsientos, int velocidadMaxima, tipoColor color, boolean
> esAutomatico) {
>     this.marca = marca;
>     this.modelo = modelo;
>     this.motor = motor;
>     this.tipoCombustible = tipoCombustible;
>     this.tipoAutomóvil = tipoAutomóvil;
>     this.numeroPuertas = numeroPuertas;
>     this.cantidadAsientos = cantidadAsientos;
>     this.velocidadMaxima = velocidadMaxima;
>     this.color = color;
>     this.esAutomatico = esAutomatico;
> }
>
> /*
> * Método que devuelve la marca de un automóvil
> * @return La marca de un automóvil
> */
> String getMarca() {
>     return marca;
> }
>
> /*
> * Método que devuelve el modelo de un automóvil
> * @return El modelo de un automóvil
> */
> int getModelo() {
>     return modelo;
> }
>
> /*
> * Método que devuelve el volumen en litros del cilindraje del motor
> * de un automóvil
> * @return El volumen en litros del cilindraje del motor de un
> * automóvil
> */
> int getMotor() {
>     return motor;
> }
>
> /*
> * Método que devuelve el tipo de combustible utilizado por el motor
> * de un automóvil
> * @return El tipo de combustible utilizado por el motor de un
> * automóvil

```

```

*/
tipoCom getTipoCombustible() {
    return tipoCombustible;
}

/*
 * Método que devuelve el tipo de automóvil
 * @return El tipo de automóvil
 */
tipoA getTipoAutomóvil() {
    return tipoAutomóvil;
}

/*
 * Método que devuelve el número de puertas de un automóvil
 * @return El número de puertas que tiene un automóvil
 */
int getNumeroPuertas() {
    return numeroPuertas;
}

/*
 * Método que devuelve la cantidad de asientos de un automóvil
 * @return La cantidad de asientos que tiene un automóvil
 */
int getCantidadAsientos() {
    return cantidadAsientos;
}

/*
 * Método que devuelve la velocidad máxima de un automóvil
 * @return La velocidad máxima de un automóvil
 */
int getVelocidadMaxima() {
    return velocidadMaxima;
}

/*
 * Método que devuelve el color de un automóvil
 * @return El color de un automóvil
 */
tipoColor getColor() {
    return color;
}

/*
 * Método que devuelve la velocidad actual de un automóvil
 * @return La velocidad actual de un automóvil
 */
int getVelocidadActual() {
    return velocidadActual;
}

// Metodo que devuelve false o true de que el automovil es automatico o no
boolean getEsAutomatico(){
    return esAutomatico;
}

```

```

/*
 * Método que establece la marca de un automóvil
 * @param marca Parámetro que define la marca de un automóvil
 */
void setMarca(String marca) {
    this.marca = marca;
}

/*
 * Método que establece el modelo de un automóvil
 * @param modelo Parámetro que define el modelo de un automóvil
 */
void setModelo(int modelo) {
    this.modelo = modelo;
}

/*
 * Método que establece el volumen en litros del motor de un automóvil
 * @param motor Parámetro que define el volumen en litros del
 * motor de un automóvil
 */
void setMotor(int motor) {
    this.motor = motor;
}

void setVelocidadMaxima(int velocidadMaxima) {
    this.velocidadMaxima = velocidadMaxima;
}

/*
 * Método que establece el color de un automóvil
 * @param color Parámetro que define el color de un automóvil
 */
void setColor(tipoColor color) {
    this.color = color;
}

/*
 * Método que establece la velocidad de un automóvil
 * @param velocidadActual Parámetro que define la velocidad actual
 * de un automóvil
 */
void setVelocidadActual(int velocidadActual) {
    this.velocidadActual = velocidadActual;
}

/*
 * Método que incrementa la velocidad de un automóvil
 * @param incrementoVelocidad Parámetro que define la cantidad a
 * incrementar en la velocidad actual de un automóvil
 */

// Metodo que estable si el aotumovil es ono automatico
void setEsAutomatico(boolean esAutomatico){

```

```

>         this.esAutomatico = esAutomatico;
>     }
>
>     void acelerar(int incrementoVelocidad) {
>         if (velocidadActual + incrementoVelocidad < velocidadMaxima) {
>             /* Si el incremento de velocidad no supera la velocidad máxima */
>             velocidadActual = velocidadActual + incrementoVelocidad;
>         } else {
>
>             /* De otra manera no se puede incrementar la velocidad y se genera
mensaje */
>             System.out.println("No se puede incrementar a una velocidad
superior a la máxima del automóvil.");
>         }
>     }
>
>     /*
>     * Método que decrementa la velocidad de un automóvil
>     * @param marca Parámetro que define la cantidad a decrementar en
>     * la velocidad actual de un automovil
>     */
>
>     void desacelerar(int decrementoVelocidad) {
>
>         /* La velocidad actual no se puede decrementar alcanzando un valor
negativo */
>         if ((velocidadActual - decrementoVelocidad) > 0) {
>             velocidadActual = velocidadActual - decrementoVelocidad;
>         } else {
>             /* De otra manera no se puede decrementar la velocidad y se
genera mensaje */
>             multas += 50;
>             numero_multas ++;
>             System.out.println("No se puede decrementar a una velocidad
negativa.");
>             System.out.println("Se generó una multa por sobrepasar la
vecilidad maxima");
>         }
>     }
>
>     /*
>     * Método que coloca la velocidad actual de un automóvil en cero
>     */
>     void frenar() {
>         velocidadActual = 0;
>     }
>
>     /*
>     * Método que calcula el tiempo que tarda un automóvil en recorrer
>     * cierta distancia
>     * @param distancia Parámetro que define la distancia a recorrer por
>     * el automóvil (en kilómetros)
>     */
>     double calcularTiempoLlegada(int distancia) {
>         return distancia / velocidadActual;
>     }

```

```

// Agragar una funcion para determinar un automovio tiene multas
int tieneMultas() {
    return numero_multas;
}

int valorTotalMultas(){
    return multas;
}

/*
 * Método que imprime en pantalla los valores de los atributos de un
 * automóvil
 */
void imprimir() {
    System.out.println("Marca = " + marca);
    System.out.println("Modelo = " + modelo);
    System.out.println("Motor = " + motor);
    System.out.println("Tipo de combustible = " + tipoCombustible);
    System.out.println("Tipo de automóvil = " + tipoAutomóvil);
    System.out.println("Número de puertas = " + numeroPuertas);

    System.out.println("Cantidad de asientos = " + cantidadAsientos);
    System.out.println("Velocida máxima = " + velocidadMaxima);
    System.out.println("Color = " + color);
    System.out.print("Es automatico = " + esAutomatico);
    System.out.println();
}

/*
 * Método main que crea un automóvil, imprime sus datos en
 * pantalla y realiza varios cambios en su velocidad
 */
public static void main(String args[]) {

    Automovil auto1 =
new Automovil("Ford",2018,3,tipoCom.DIESEL,tipoA.EJECUTIVO,5,6,250,tipoColor.
NEGRO, true);
    auto1.imprimir();
    auto1.setVelocidadActual(100);
    System.out.println("Velocidad actual = " + auto1.velocidadActual);
    auto1.acelerar(20);
    System.out.println("Velocidad actual = " + auto1.velocidadActual);
    auto1.desacelerar(50);
    System.out.println("Velocidad actual = " + auto1.velocidadActual);
    auto1.frenar();
    System.out.println("Velocidad actual = " + auto1.velocidadActual);
    auto1.desacelerar(20);
    auto1.tieneMultas();
    System.out.println("Numero de multas = " + auto1.numero_multas);
    auto1.valorTotalMultas();
    System.out.println("Valor total de multas = " + auto1.multas + "$");
}
}







```

Prueba de ejecución:

```
C:\GONZALES\SIS103\Clases_Objeto\Estado_objeto>javac Automovil.java

C:\GONZALES\SIS103\Clases_Objeto\Estado_objeto>java Automovil
Marca = Ford
Modelo = 2018
Motor = 3
Tipo de combustible = DIESEL
Tipo de automóvil = EJECUTIVO
Número de puertas = 5
Cantidad de asientos = 6
Velocidad máxima = 250
Color = NEGRO
Es automatico = true
Velocidad actual = 100
Velocidad actual = 120
Velocidad actual = 70
Velocidad actual = 0
No se puede decrementar a una velocidad negativa.
Se generó una multa por sobrepasar la velocidad máxima
Número de multas = 1
Valor total de multas = 50$

C:\GONZALES\SIS103\Clases_Objeto\Estado_objeto>_
```

 Automovil\$tipoA.class	20/3/2024 09:55	Archivo CLASS	2 KB
 Automovil\$tipoColor.class	20/3/2024 09:55	Archivo CLASS	2 KB
 Automovil\$tipoCom.class	20/3/2024 09:55	Archivo CLASS	2 KB
 Automovil.class	20/3/2024 09:55	Archivo CLASS	5 KB
 Automovil.java	20/3/2024 09:55	Archivo de origen ...	12 KB
 Automovil.txt	20/3/2024 09:56	Documento de tex...	12 KB