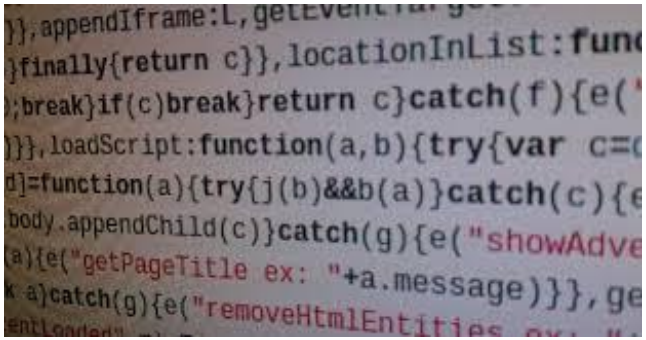


UT 7

INTRODUCCIÓN POO.

Utilización de Objetos

PRIMERA PARTE

IES JUAN DE LA CIERVA
DPTO. INFORMÁTICA

CONTENIDOS DE LA PRESENTACIÓN

Objetivos

Esta unidad propone la utilización de objetos ya definidos en el propio lenguaje, la creación por el usuario de objetos sencillos, así como las propiedades de estos.

Contenidos

1. **Utilización de métodos.**
2. **Utilización de propiedades.**
3. **Parámetros y valores devueltos**.
4. Librerías de objetos.
5. Constructores.
6. Destrucción de objetos y liberación de memoria.

Ejemplo: clase Vehículo

Programa una clase controladora que instanciará objetos de la clase Vehiculo del Tema 5 y en la que deberás:

- **Instanciar 2 objetos (monovolumen y deportivo)**
- **Monovolumen con los valores de los atributos que ya teníamos**
- **Deportivo:**
  - **Marca:Lexus**
  - **Modelo:F Sport**
  - **2 pasajeros**
  - **capacidad del depósito : 66**
  - **Consumo:18,8**

Calcula da autonomía de los 2 vehículos y muestra el resultado.

Método: Concepto

- Subrutina que manipula los datos definidos por la clase.
- Bloque de código al que se hace referencia mediante el nombre y que se puede invocar desde cualquier parte de la propia clase o desde otra.

Método: Concepto

- Subrutina que manipula los datos definidos por la clase.
- Bloque de código al que se hace referencia mediante el nombre y que se puede invocar desde cualquier parte de la propia clase o desde otra.

Métodos: Declaración I

Formato:

```
<modificador> <tipo_dev> <nombre método>(lista de argumentos)  {
    cuerpo del método.
}
```

Hay 4 elementos para definir un método:

Modificador de acceso:

- **Predeterminado Package(default)** : Accesible desde cualquier clase que esté dentro del paquete.
- **Público (public):** Accesibles desde cualquier clase de cualquier paquete.
- **Privado (private):** Sólo accesibles desde dentro de la clase donde se declaran.

  (*) Una clase no puede ser private.
- **Protegido (protected):** Accesibles desde cualquier clase del paquete o subclases en otros paquetes.

Métodos: Declaración II

Formato:

```
<modificador> <tipo_dev> <nombre método>(lista de argumentos)   {
    cuerpo del método.
 }
```

Tipo_dev: tipo de datos que devuelve el método.
Si no devuelve datos será void

Nombre método: Identificador válido no utilizado para otros elementos

Métodos: Declaración III

Formato:

```
<modificador> <tipo_dev> <nombre método>(lista de argumentos) {
    cuerpo del método.
}
```

Lista de argumentos:

Secuencia de pares de tipo e identificador separados por comas.

Argumento (parámetro): variable que recibe el valor de los argumentos pasados al método al invocarlo.

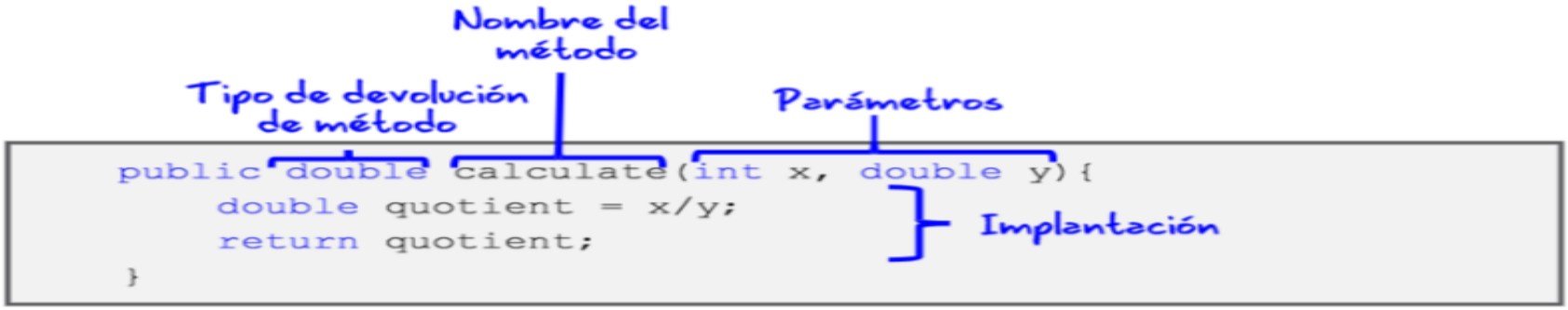
Pueden ser **de cualquier tipo primitivo o tipo de objeto**.

El tipo de parámetro utilizado al invocar al método **debe coincidir con el especificado en la definición** del método.

Si el método **carece** de parámetros se pondrá **( )**

Implementación de un método: Ejemplos

Ejemplos:



Ejercicio . Creación Métodos Clase Vehículo

Crear un método que muestre la autonomía de los vehículos.

Métodos: Uso de parámetros

<tipo_dev> <nombre método>(lista de argumentos) {
cuerpo del método.
}

Se pueden pasar uno o más valores a un método al invocarlo.

- Argumento : valor pasado al método al invocarlo
- Parámetro : variable que recibe el argumento.

¿Cómo se invoca?

objeto.método (lista de argumentos)

¿Cómo se declara en el método?

método (lista de parámetros)

Métodos: Retornar valores

<tipo_dev> <nombre método>(lista de argumentos) {

cuerpo del método.

}

Los valores devueltos por un método tienen diferentes aplicaciones:

- Resultado de un cálculo
- Puede indicar operación satisfecha o fallida.
- Incluir un código de estado.

¿Cómo se devuelve un valor?

return <valor devuelto>;

Ejemplo 4: Creación y uso de Métodos

Crear un método que calcule la autonomía de los vehículos, devolviendo el resultado a la clase controladora.

Ejemplo 5: Creación y uso de Métodos

```
class Numero {

      //devolver True si el número es par
      boolean par(int x) {
            if ((x%2) == 0)   return true;
            else     return false;
      }
}

class ComprobarPar {
       public static void main(String args[ ]) {
              Numero num = new Numero( );
              if (num.par(10) ) System.out.println("El número 10 es par ");
              if (num.par (9) ) System.out.println("El número 9 es par ");
              if (num.par (8) ) System.out.println("El número 8 es par ");
       }
}
```

Ejemplo 6: Creación y uso de Métodos

```
class Numero {

    //devolver True si el número es par
    ‘……

    //devolver True si el primer parámetro es divisor del segundo
        boolean divisor(int a, int b) {
            if ( (b%a) == 0) return true;
            else return false;
        }
    }
class ComprobarDivisor {
      public static void main(String args[ ]) {
            Numero num = new Numero( );
            if (num.Divisor(2,10) ) System.out.println(“El número 2 es divisor de 10 “);
            if (num.Divisor (3,10) ) System.out.println(“El número 3 es divisor de 10“);
    }
}
```

Ejercicio: Algunas preguntas

1) **¿Cuál es la diferencia entre clase y objeto?**
2) **¿Cómo se define una clase?**
3) **Con dos instrucciones diferentes, indique como declarar el objeto *contador* de la clase *MiContador.***
4) **Indica cómo declarar el método miMetodo si tiene un tipo de devolución double y dos parámetros a y b enteros.**
5) **¿Cómo debe darse la instrucción para que un método devuelva un valor?**
6) **¿Cuál es la función de new?**
7) **Si un método no devuelve valores ¿Qué tipo de devolución debe tener?.**

Ejercicio 7: Creación y uso de Métodos

Añadir un método a la clase Vehiculo en el que se calcule los litros de combustible necesarios para recorrer una determinada distancia dada en kilómetros.

La distancia se pasará como argumento en la invocación al método y será de tipo entero. Se pedirá por teclado y deberá ser positiva.

El método devolverá un resultado de tipo double.

Dar como nombre al programa principal *CrearObjetoVehiculo4*.

Dar como nombre al método *combustibleNecesario*

¿Qué tipo de dato puede ser un atributo, parámetro,….?

ESTRUCTURA
- Plástico
- 4 ruedas
- 1 volante
- …

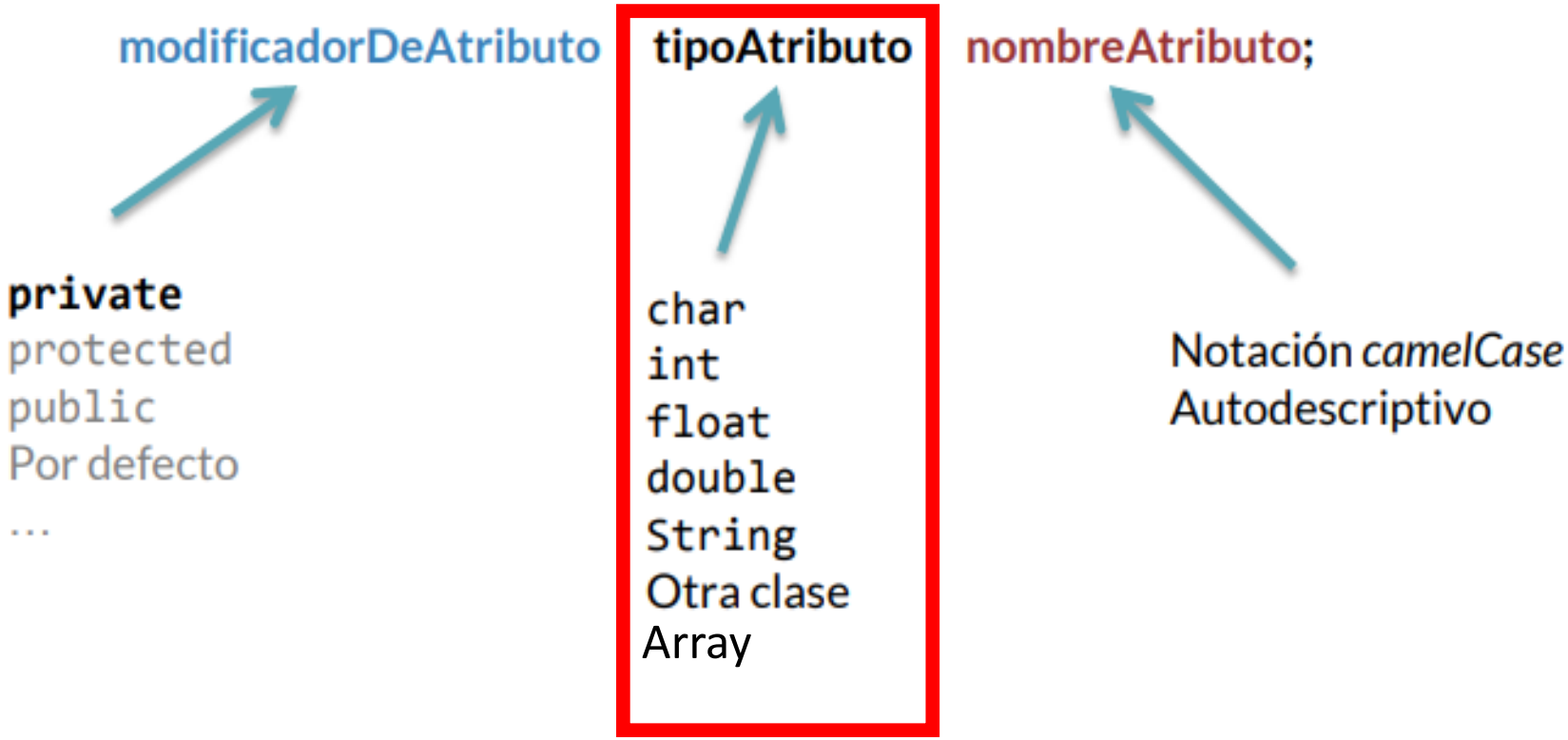
ATRIBUTOS

COMPORTAMIENTO
- Mover adelante
- Mover atrás
- …

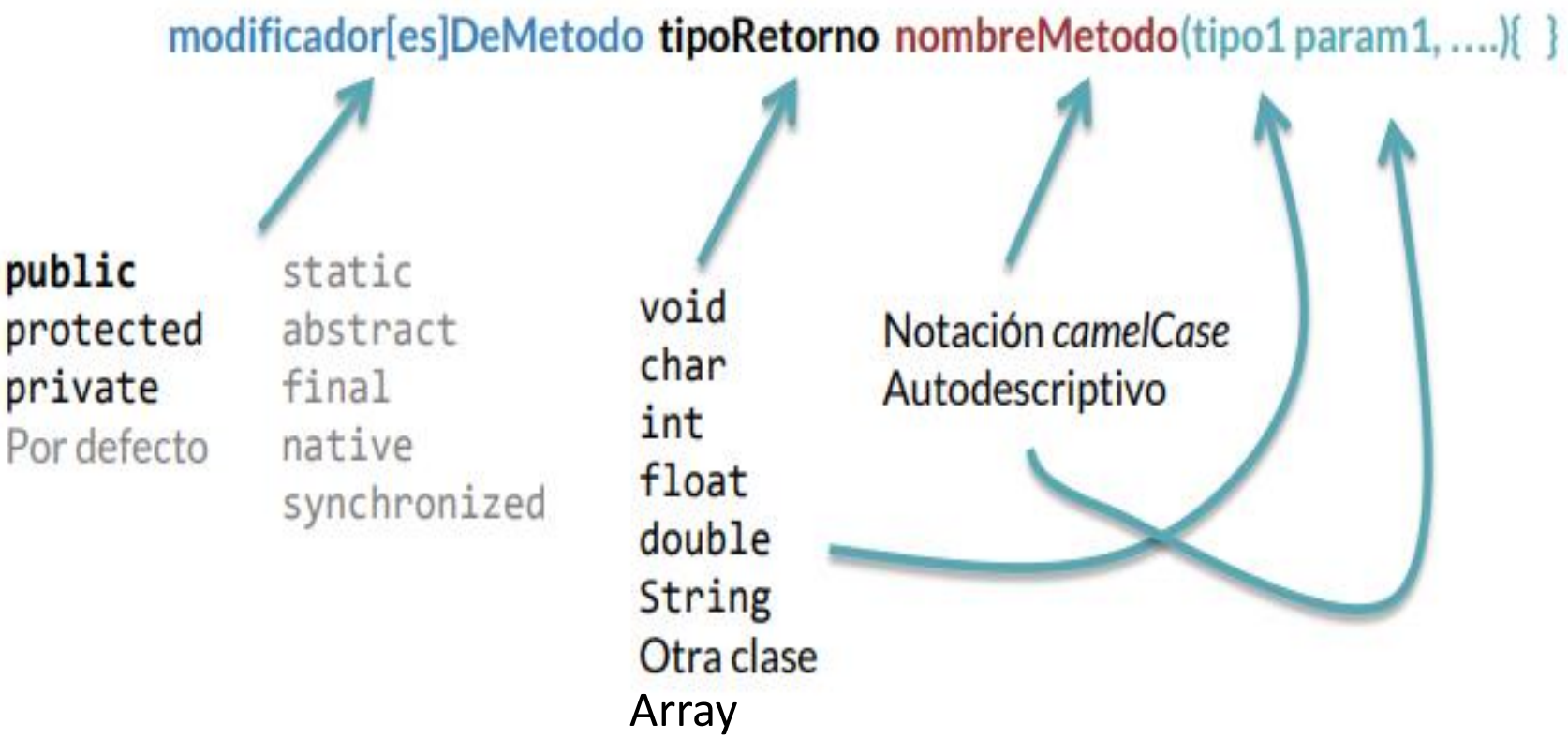
MÉTODOS

```
public class Coche {

    private String color;
    private String numRuedas;
    //…


    public void adelante() {
        //…
    }

    public void atrás() {
        //…
    }
    //…
}
```

¿Qué tipo de dato puede ser un atributo, parámetro,….?



¿Qué tipo de dato puede ser un atributo, parámetro,….?



Ejercicio

- Seguimos trabajando con la clase Vehículo:
- Le añadiremos un atributo que será un vector para guardar los colores que tenemos disponibles en esa marca y modelo de vehículo. Sera un vector de String de tamaño 7.
- Después de instanciar los dos vehículos, les daremos valores a estos vectores.

Declaración de un atributo de tipo array

```
class  Persona {


    // Estructura, conocida como propiedades o atributos
    String nombre;
    String apellidos;
    int edad;
    int altura;
    float peso ;
    String [] aficiones = new String[5];


public class EjemploPersona {

    public  static  void  main ( String [] args ) {

            Persona persona =  new  Persona ();

            persona.edad=15;
            persona.aficiones[0]="senderismo";
```

Práctica 7.1.

- Clase Rectángulo.
- Creación de clases, objetos y métodos.