

3. Entendiendo las clases

Programación Orientada a Objetos

Objetivos

- Programar una clase
- Utilizar atributos y métodos
- Describir y utilizar modificadores de acceso
- Describir reglas de sintaxis y convenciones
- Describir la estructura de un programa
- Describir tipos de datos primitivos
- Describir palabras reservadas
- Describir variables
- Utilizar conversión entre tipos de datos
- Utilizar operaciones y operadores
- Describir y programar métodos
- Crear objetos



Clases

- Una clase es un prototipo o plantilla para crear objetos.
- En Java, cada clase pública se almacena en un archivo independiente con el mismo nombre, comenzando con una letra mayúscula.
- Al ser un prototipo, es en la implementación de la clase donde se programa el comportamiento de los futuros objetos de la clase.



Estructura

```
public class nombre_clase  
{
```

cuerpo de la clase

```
}
```

La “cáscara” de
la clase

El contenido de la
clase

```
public final class Persona {  
    // ATRIBUTOS  
    // MÉTODOS  
}
```

Persona.java

modificador de acceso

modificador

nombre de la clase

Atributos

- Los atributos almacenan valores para un objeto
- También se les llama variables de instancia
- Los atributos definen el estado de un objeto

```
public class TicketMachine
{
    private int price;
    private int balance;
    private int total;

    //Constructor y métodos omitidos
}
```

modificador de acceso

tipo

nombre atributo

private int price;

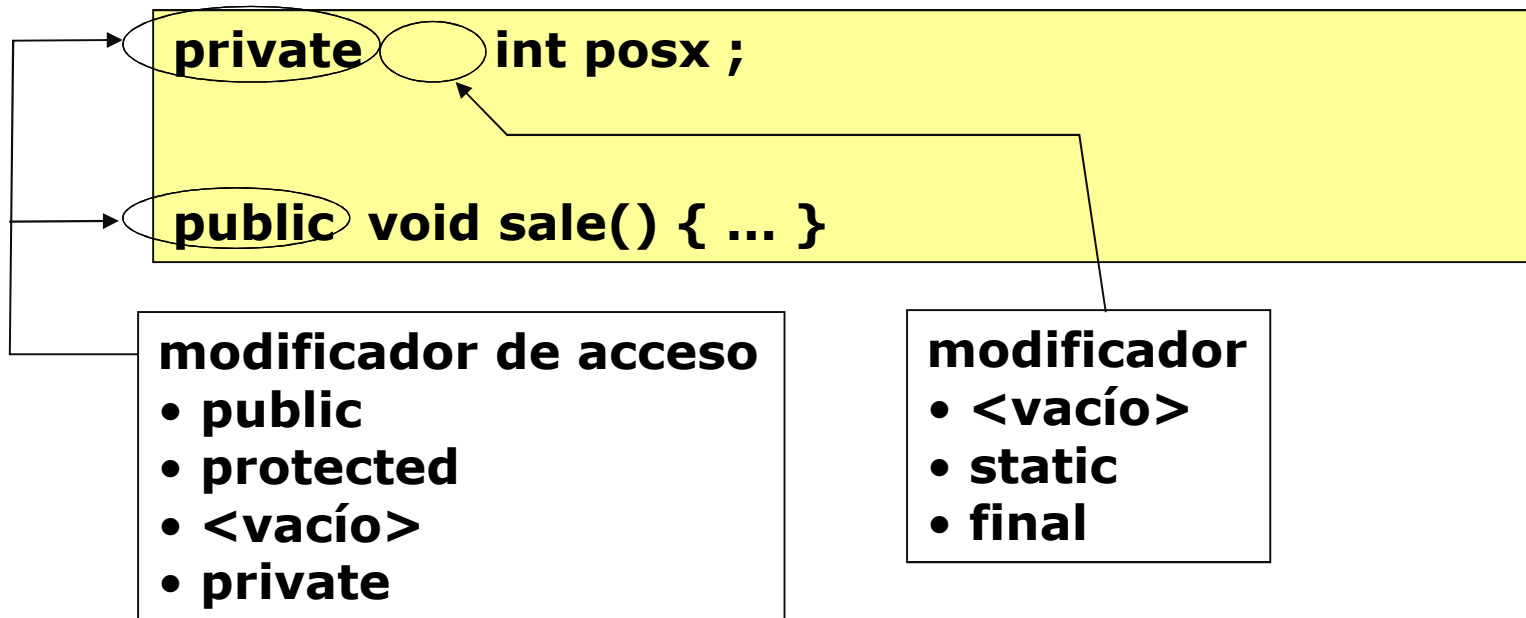
Atributos y Métodos

```
public class Perro {  
    // ATRIBUTOS  
    int tarea;  
    int posx, posy;  
  
    // MÉTODOS  
    void sit(){  
        tarea=1;  
    }  
  
    void ven(int px,int py){  
        posx=px;  
        posy=py;  
        tarea=0;  
    }  
}
```

```
    int vertarea(){  
        return tarea;  
    }  
  
    void sale(){  
        posx = posx + 10;  
        posy = posy + 20;  
    }  
}
```

Modificadores de Acceso de Atributos y Métodos

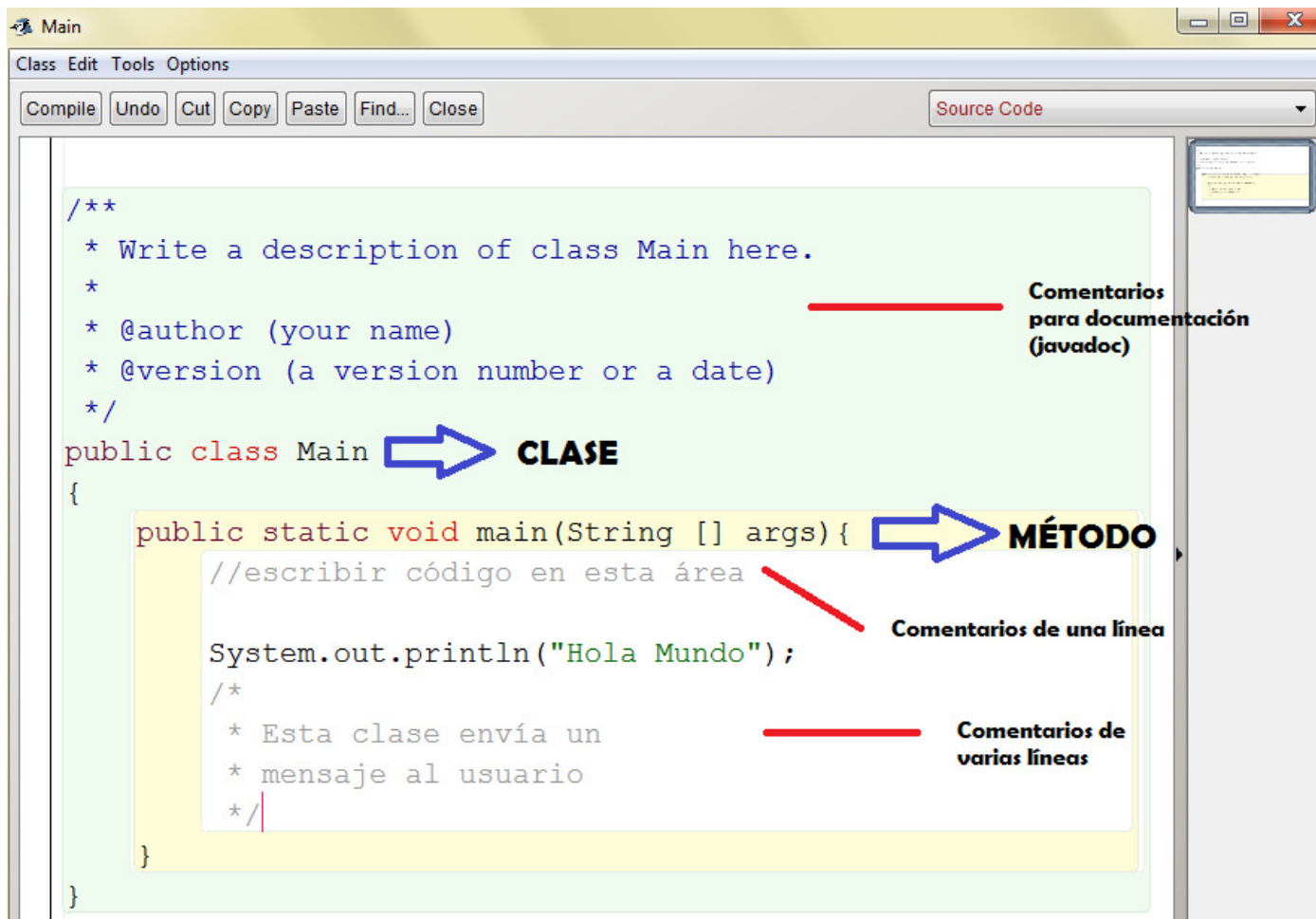
- Los atributos y métodos de una clase también pueden incluir modificadores:



Modificador de Acceso

Accesibilidad	public	protected	<vacío>	private
Desde la misma clase	sí	sí	sí	sí
Desde cualquier clase o subclase del mismo paquete	sí	sí	sí	no
Desde una subclase en otro paquete	sí	sí, usando herencia	no	no
Desde cualquier clase en cualquier paquete	sí	no	no	no

Reglas de Sintaxis



Convenciones

- Nombres de clases empiezan con mayúscula.
- Nombres de métodos empiezan con minúscula.
- Nombres de atributos con minúscula.
- Cada clase se guarda en un archivo separado que lleva el mismo nombre de la clase.



Estructura de un Programa

- Un programa contiene paquetes, un paquete contiene clases y una clase contiene métodos.
- Método main
 - Al ejecutar un programa en Java, se comienza ejecutando las instrucciones del método main.
 - Todo programa debe tener UN método main.

Tipos de Datos Primitivos

Tipo	Descripción	Largo
byte	Entero de un byte	8 bits
short	Entero corto	16 bits
int	Entero	32 bits
long	Entero largo	64 bits
float	Punto flotante de precisión simple	32 bits
double	Punto flotante de precisión doble	64 bits
char	Caracter	16 bits
boolean	Booleano	(true y false)

Palabras Reservadas

abstract	continue	finally	int	public	throw
assert	default	float	interface	return	throws
boolean	do	for	long	short	transient
break	double	goto	native	static	true
byte	else	if	new	strictfp	try
case	enum	implements	null	super	void
catch	extends	import	package	switch	volatile
class	false	inner	private	synchronized	
const	final	instanceof	protected	this	while

Variables

- Una variable representa un espacio en memoria para almacenar un valor de un determinado tipo.
- El valor de una variable, a diferencia de una constante, puede cambiar durante la ejecución del programa.
- Para utilizarla en un programa hay que declararla.
- Se enuncia y se le asigna un tipo:
 tipo identificador;

- Ejemplos:

```
short dia, mes, año;  
int contador = 0;  
String nombre = "";
```

Asignación

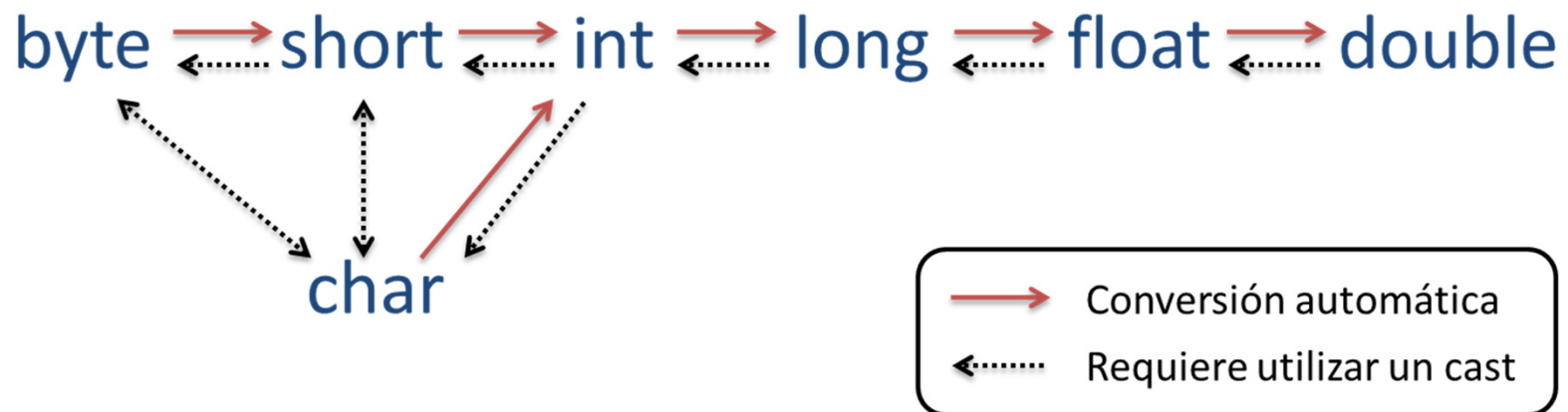
- Los valores se almacenan en los campos (y en otras variables) vía sentencias de asignación:
 `variable = expresión;`
 `price = ticketCost;`
- Una variable almacena un sólo valor, por lo que cualquier valor previo se pierde.

Conversión entre Tipos de Datos

- Cuando en una expresión intervienen operandos con tipos de datos diferentes, hay que realizar su conversión.
- Se convierten los operandos al tipo del operando cuya precisión sea más alta.
- Si es una asignación se convierte el valor de la derecha al tipo de la variable de la izquierda siempre que no haya pérdida de la información.
- En otro caso, hay que realizar la conversión explícita.

Conversiones Implícitas Permitidas

- Las flechas indican las conversiones implícitas permitidas.
- De menos a más precisos



Conversión Explícita

- Conversión forzada que se realiza mediante una construcción denominada “cast” de la siguiente forma:

(tipo) expresión

- Hay que tener cuidado con la pérdida de precisión.
- Ejemplo:

```
double pesoMax = 55.73;  
  
float limite = pesoMax; //Error. Hay posible pérdida de datos.  
  
//Hay que utilizar el 'casting' o casteo.  
  
limite = (float) pesoMax; //Correcto. El programador se hace  
//'responsable' de la posible pérdida de datos.
```

Operaciones y Operadores

- Una expresión es un conjunto de operandos unidos mediante operadores para especificar una operación determinada.
- Todas las expresiones cuando se evalúan retornan un valor.
- Uno o más valores y/o variables pueden formar parte de operaciones.
- Las operaciones dentro de una expresión se realizan siguiendo reglas de precedencia.

Operadores

(-) Precedencia (+)

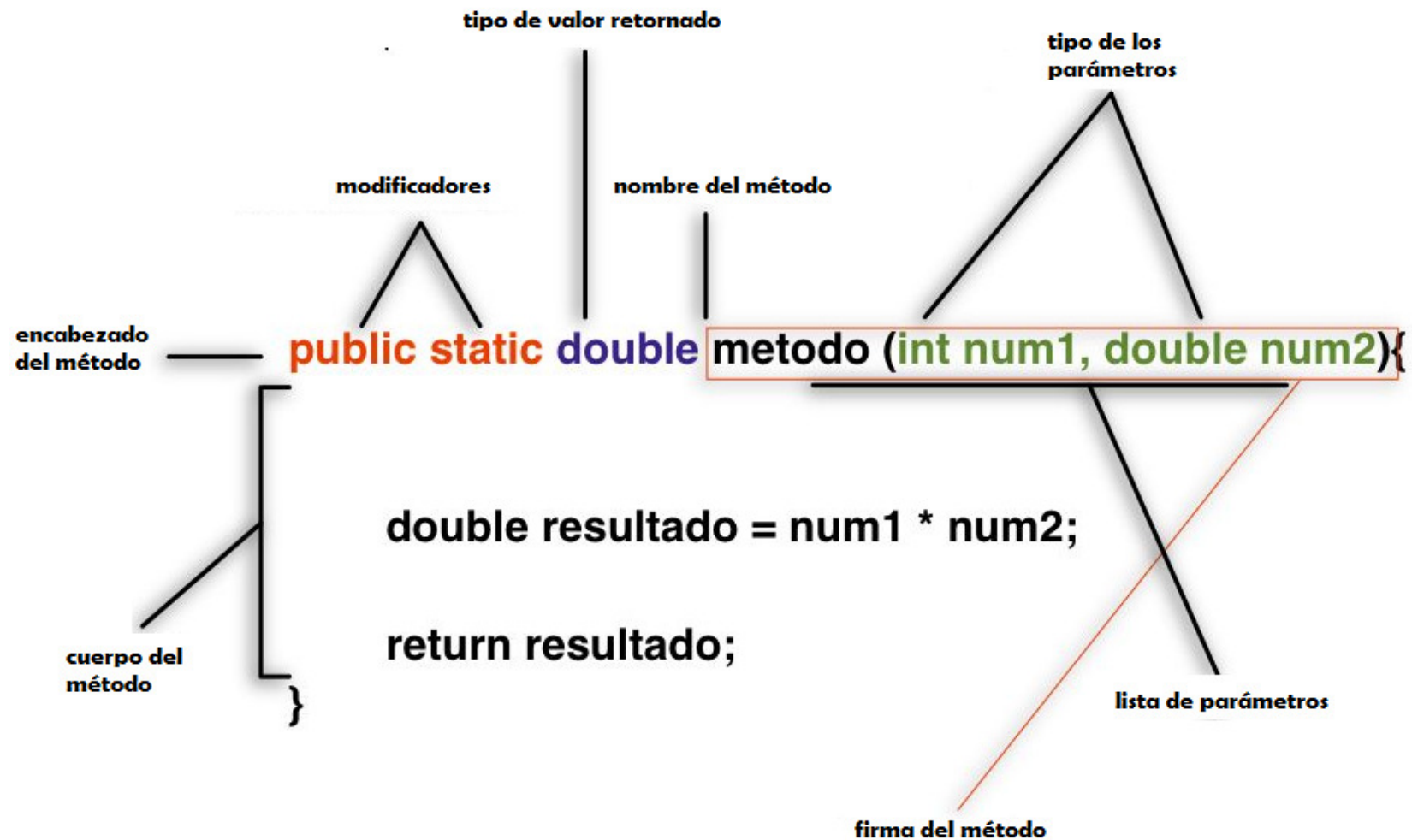
Tipo de operadores	Operadores de este tipo
Operadores posfijos	[] . (parametros) expr++ expr--
Operadores unarios	++expr --expr +expr -expr ~ !
Creación o conversión	new (tipo) expr
Multiplicación	* / %
Suma	+ -
Desplazamiento	<<
Comparación	< <= == instanceof
Igualdad	== !=
AND a nivel de bit	&
OR a nivel de bit	^
XOR a nivel de bit	
AND lógico	&&
OR lógico	
Condicional	? :
Asignación	= += -= *= /= %= &= ^= = <<= ==

Métodos

- Definen las operaciones que se pueden realizar con los atributos de una clase.
- Un método siempre está contenido dentro del cuerpo de una clase.
- Es una colección de sentencias que ejecutan una tarea específica, implementan el comportamiento de los objetos.
- Un método no puede contener otro método, es decir, no permite métodos anidados.



Definición



Variables

- **Variables:** Son locales y sólo son accesibles dentro del método.
- **Modificador:** Palabra clave que modifica el nivel de protección del método.
- **Tipo de retorno:** Valor que retorna el método, ejemplo: int, String.
- Si no retorna nada se utiliza la palabra reservada void.
- Sintaxis de retorno:
return expresión;

Parámetros

- Lista de Parámetros: Variables que reciben los valores de los argumentos especificados cuando se invoca al mismo.
- Lista de cero, uno o más identificadores con sus tipos, separados por comas, ejemplo:

(int num1, double num2, String nombre)

Ejemplos

RETORNO
↓
`String hablar();`
↑
PARÁMETROS DE ENTRADA

`void escuchar(mensaje);`
↑
NO TIENE RETORNO
↑
PARÁMETROS DE ENTRADA

`String escucharyHablar(String msg);`
↑
TIENE RETORNO
↑
PARÁMETROS DE ENTRADA

Métodos

- ✓ **Constructores**
 - sin parámetros* `public Alumno()`
 - con parámetros* `public Alumno(String nombre, int edad)`
- ✓ **Accesadores**
 - `public String getNombre()`
- ✓ **Mutadores**
 - `public void setNombre(String nombre)`
- ✓ **Impresión**
 - `public void printAlumno()`
- ✓ **Customer**
 - `public double calcularPromedio(double notaP, double notaE)`
- ✓ **Main**
 - `public static void main(String[] args)`

Método Main

- Toda aplicación java tiene un método main y sólo uno.
- Es el punto de entrada y de salida de la aplicación.
- Definición:

```
public static void main(String[] args) {  
    //cuerpo del método  
}
```

- Es público, estático, no devuelve nada y tiene un argumento del tipo String.

Método Main

- public: que puede ser accesado desde el programa (desde fuera de la clase).
- static: que pertenece a la clase y no al objeto.
- void: que no retorna nada.
- argumentos: es un arreglo de lista de caracteres que se llama args.

Método Constructor

Es un método especial de una clase que es llamado automáticamente siempre que se crea un objeto de esa clase.

Función  Iniciar Objeto

Un objeto de una clase se crea de la siguiente forma:

```
Persona persona1 = new Persona ();
```

Diagram illustrating the object creation process:

- Persona** (circled in yellow) is labeled as **objeto** (object).
- persona1** (circled in purple) is labeled as **objeto** (object).
- Persona** (circled in yellow) is labeled as **clase** (class).



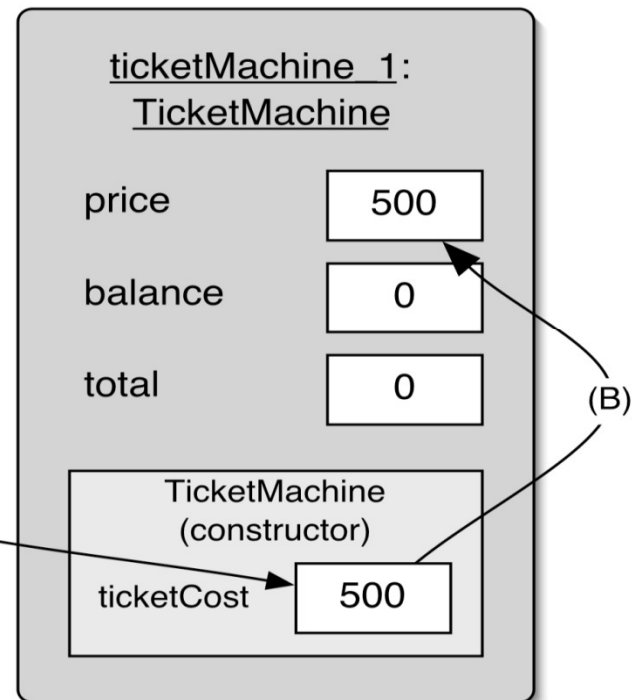
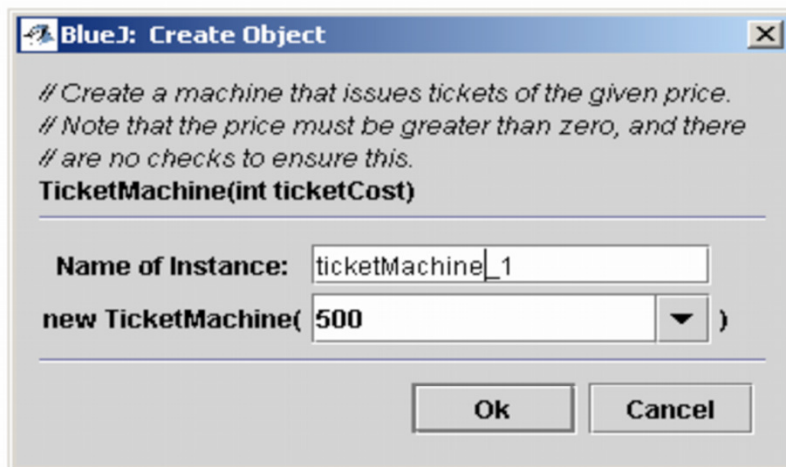
Cuando se crea un objeto

- Asigna memoria para el objeto con el operador new.
- Inicia los atributos de ese objeto.
- Llama al constructor de la clase.
- El método constructor permite parámetros como cualquier otro método. Sino se especifica, se crea uno público por omisión sin parámetros.
- Su nombre es el mismo de la clase a la que pertenece.

Ejemplo: ¿Cómo se llamará el constructor para la clase Persona?

Pasar información vía parámetros

```
public TicketMachine(int ticketCost)
{
    price = ticketCost;
    balance = 0;
    total = 0;
}
```



Pasar información vía parámetros

```
public class CFecha
{
    //atributos
    int dia, mes, año;

    //métodos
    public CFecha() //constructor
    {
        dia = 1;
        mes = 1;
        año = 2000;
    }
}
```


Llamada a un constructor...

- Ejemplo: Llamada a un constructor sin parámetros

```
public static void main (String [] args)
{
    CFecha fecha = new CFecha ();
}
```

- Ejemplo: Llamada a un constructor con parámetros

```
public static void main (String [] args)
{
    CFecha fecha = new CFecha (1,3,2002);
}
```

Sobrecarga del constructor

- Se pueden definir múltiples constructores con el mismo nombre y diferentes parámetros con el fin de iniciar un objeto de una clase de diferentes formas.
- Ejemplo:

```
public CFecha()  
{  
}
```

Constructor
sin parámetros

```
public CFecha(int dd)  
{  
}
```

Constructor
con un parámetro



Sobrecarga del constructor

```
public CFecha(int dd, int mm)
{
}
```

Constructor
con dos parámetros

```
public CFecha(int dd, int mm, int aaaa)
{
}
```

Constructor
con tres parámetros

- ¿Cómo se puede invocar al constructor CFecha con 0, 1, 2 ó 3 parámetros?



Sobrecarga del constructor

CFecha fecha1 = new CFecha();

Constructor
sin parámetros

CFecha fecha2 = new CFecha(3);

Constructor
con un parámetro

CFecha fecha3 = new CFecha(3,12);

Constructor
con dos parámetros

CFecha fecha4 = new CFecha(3,12,2014);

Constructor
con tres parámetros

Sobrecarga del constructor

Operador this

```
public class Perro {  
    int tarea;  
    int posx, posy;  
  
    public Perro(int posx, int posy){  
        this.posx = posx;  
        posy = posy;  
        tarea = 0;  
    }  
  
    public Perro(){  
        tarea = 0; posx = 0; posy = 0;  
    }  
    ...  
}
```

Sobrecarga

Métodos Accesadores y Mutadores

- La clase incluye la idea de ocultación de datos, es decir, no se puede acceder directamente a sus atributos, sino que hay que hacerlo a través de los métodos de la clase.
- Los métodos utilizados para obtener información de un objeto se llaman métodos accesadores.
- Los métodos utilizados para modificar información de un objeto se llaman métodos mutadores.

GET

Accesadores

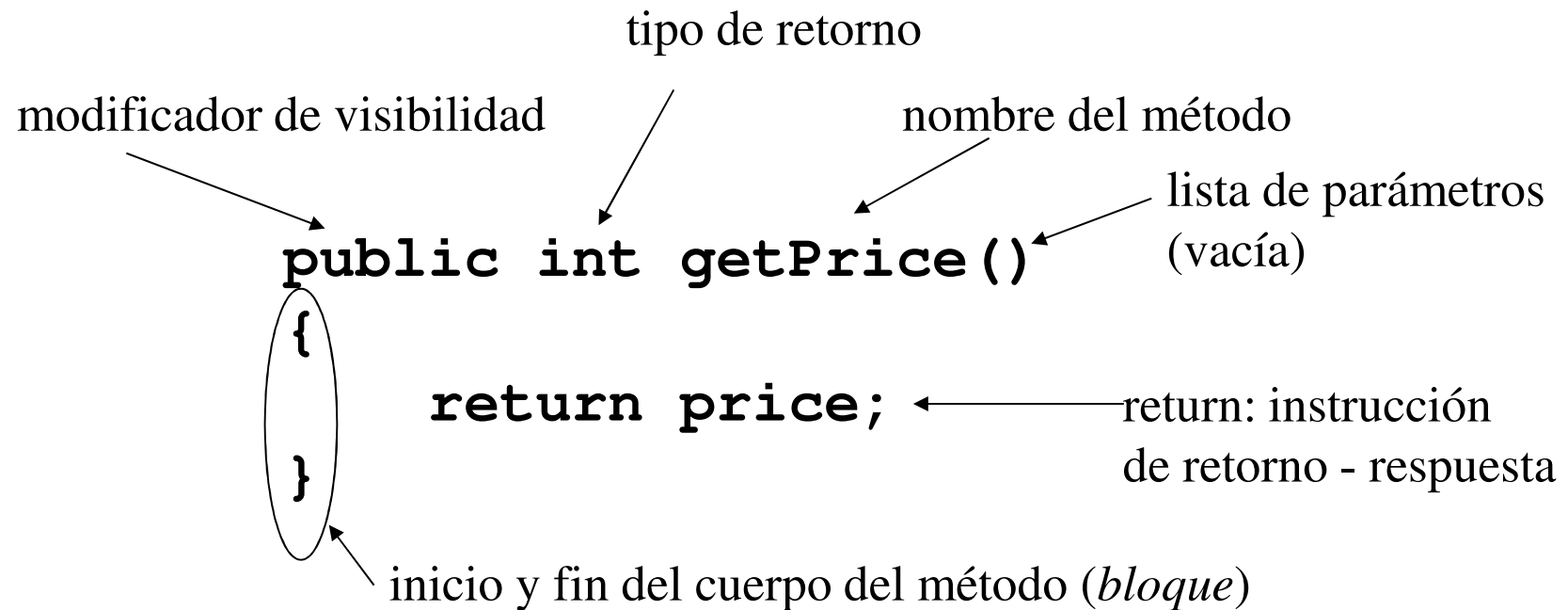
- El método accesador tiene como objetivo mostrar el contenido de un campo o atributo a quien lo solicite.
- Este método utiliza la sentencia return, que hace dos cosas:
 - Entrega el valor de respuesta del método.
 - Provoca el fin de la ejecución del método inmediatamente.

Sintaxis

```
/**  
 * [Modificador de Acceso] tipo de retorno getNombreCampo()  
 * {  
 *     return nombreCampo;  
 * }  
 */  
public int getNumerador()  
{  
    return numerador;  
}
```

Devolver
valor del
atributo
numerador

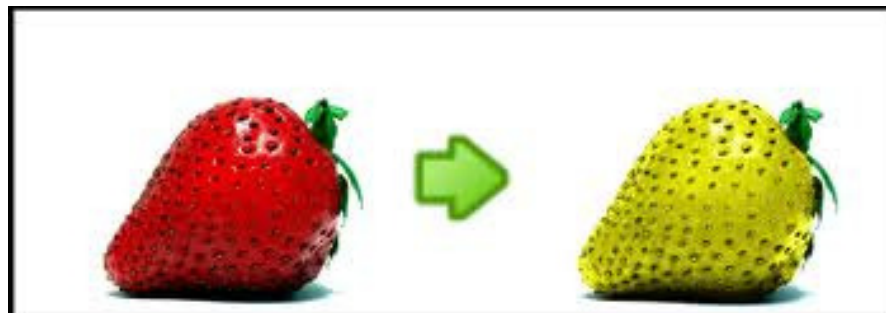
Sintaxis



SET

Mutadores

- El método mutador tiene como objetivo o responsabilidad permitir el cambio de valor para un atributo en particular, esto a través de un parámetro que recibe el nuevo valor y la asignación que permite almacenar el nuevo valor en la referencia indicada.



Sintaxis

```
/**  
 * [Modificador de Acceso] void setNombreCampo(tipo dato parametro)  
 * {  
 *     nombreCampo=parametro;  
 * }  
 */  
public void setNumerador(int nNumerador)  
{  
    numerador=nNumerador;  
}
```

Cambiar
valor del
atributo
numerador

Sintaxis

modificador de visibilidad

tipo de retorno (`void`)

nombre del método

parámetro

```
public void setPrice(int nPrice)
{
    price = nPrice;
}
```

← asignación

campo que estamos cambiando

Ejemplo

```
public class Auto{  
    private String patente;
```

```
//accesador
```

```
public String getPatente(){  
    return patente;  
}
```

devolver
patente

patente =
"CBGG31"

```
//mutador
```

```
public void setPatente(String nuevaPatente){  
    patente = nuevaPatente;  
}
```

cambiar
patente

patente antes =
"CBGG31"
patente después=
"DGAB11"

```
}
```

Método de impresión

- Una clase puede tener un método de impresión en el cual se impriman o muestren los valores de sus atributos.
- Para imprimir por pantalla se utiliza la sintaxis:
 - `System.out.println("Hola Mundo");`

- Ejemplo: clase Auto

```
public void printAuto(){  
    System.out.println("La patente es: " + patente);  
}
```

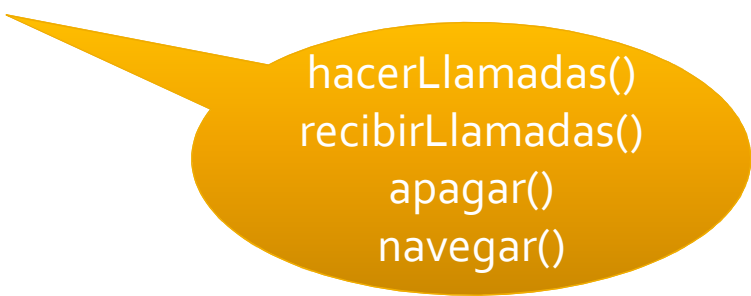


atributo

Método Customer

- Son aquellos métodos de propósito específico o de cliente.
- Los métodos customer son todos aquellos métodos distintos de los constructores, accesadores (getters) y mutadores (setters), son los métodos que se crean a partir de una funcionalidad específica.

Ejemplo: clase Celular



hacerLlamadas()
recibirLlamadas()
apagar()
navegar()

Ejemplos

clase Auto



acelerar()
frenar()

clase Animal



comer()
cazar()

Objetos

- La clase es una plantilla para crear objetos.
- Para crear un objeto de una clase se utiliza el operador new.
- Ejemplo:

```
Persona persona1 = new Persona();
```

persona1 → objeto
Persona → clase

- Cuando se crea un objeto nuevo, se asigna la cantidad de memoria necesaria para ese objeto.
- La memoria se libera cuando el objeto no se utiliza: recolector de basura.

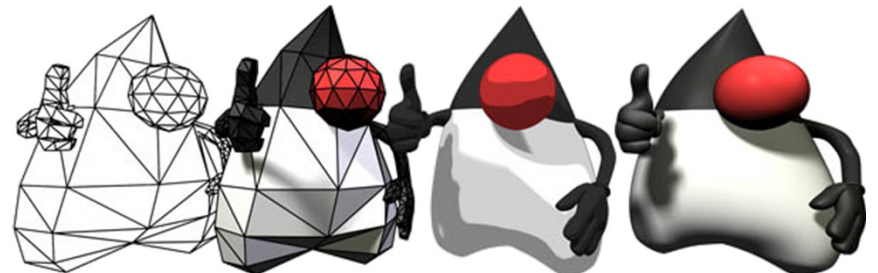
Objetos

- Un objeto de la clase se puede declarar:

```
Persona persona1 = new Persona();
```

o

```
Persona persona1;  
persona1 = new Persona();
```



¿Cuáles son objetos y clases?

```
Auto auto1, toyota, mazda;
```

```
auto1 = new Auto();  
toyota = new Auto();  
mazda = new Auto();
```

