Atributos y métodos estáticos o de clase

**Métodos y Atributos estáticos Java**

Este Tutorial será más breve pero es importante porque aclararemos algo que no tocamos en el Tutorial 2 que es donde explicamos la definición de métodos en la Programación Java, se trata de la palabra reservada de Java *static*. Esta palabra sirve para declarar atributos estáticos y sus métodos estáticos para acceder a ellos y manipularlos.

Ahora la pregunta es ¿Cuándo debo utilizar atributos y métodos estáticos?, Cuando creamos un objeto de una clase, este objeto mantiene una copia de los atributos de la clase para almacenar sus propios datos, de los métodos no porque estos solo son usados por los objetos cuando requieren manipular sus atributos y los métodos hacen exactamente la misma función para todos los objetos, pero va haber ocasiones en las que necesitemos un atributo que tenga el mismo valor para todos los objetos (que sea constante), siendo así no sería muy optimo que cada vez que creemos un objeto se esté creando la copia de ese atributo ya que estaríamos desperdiciando memoria creando un atributo que siempre va tener el mismo valor. Entonces cuando queramos un atributo así tiene que ser declarado de la siguiente forma:

**private static double tipoInteres;**

**Métodos y Atributos estáticos Java**

Ahora para dar valor a este atributo necesitamos un método que igual será estático. Este método puede acceder al atributo “tipoInteres” porque este es estático, si el atributo “tipoInteres” no está declarado como estático nos marcara error.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | public static void asignarTipoInteres(double tipo)  {  if(tipo < 0)  {  System.out.println("Tipo invalido");  return;  }  tipoInteres = tipo;  } |

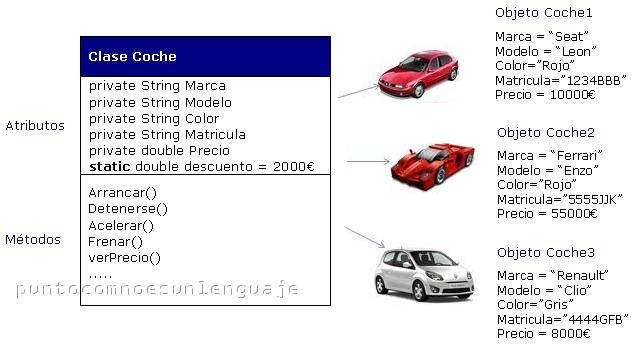
Los atributos tanto como los métodos estáticos deben ser accedidos por medio del nombre de la clase y no del objeto ya que como decíamos antes estos no forman parte de la estructura del objeto. En la siguiente imagen pongo un ejemplo llamando al método “asignarTipoInteres” por medio del nombre de la clase y por medio del nombre del objeto, podemos ver que al usar el nombre del objeto el NetBeans subraya de amarillo el método indicando que ese método esta declarado como estático.

[Métodos y Atributos estáticos Java](http://curso-programacion.com/wp-content/uploads/2012/03/M%C3%A9todos-y-Atributos-est%C3%A1ticos-Java-3.jpg)

Atributos y métodos estáticos o de clase

Supongamos una clase Coche sencilla que se utiliza en un programa de compra-venta de coches usados y de la que se crean 3 objetos de tipo Coche:

La clase contiene 6 atributos: marca, modelo, color, matrícula, precio y descuento. Supongamos que el descuento es una cantidad que se aplica a todos los coches sobre el precio de venta. Como este dato es el mismo para todos los coches y es un valor que se puede modificar en cualquier momento no debe formar parte de cada coche sino que es un dato que deben compartir todos. Esto se consigue declarándolo como **static**.

[](http://2.bp.blogspot.com/-SGYAWr7n3go/US9O9aOIFGI/AAAAAAAAAKE/3IvjBH1ZuCk/s1600/java-static.JPG)

**Entonces un Atributo static:**

-  No es específico de cada objeto. Solo hay una copia del mismo y su valor es compartido por todos los objetos de la clase.

- Podemos considerarlo como una *variable global* a la que tienen acceso todos los objetos de la clase.

- Existe y puede utilizarse aunque no existan objetos de la clase.

Para acceder a un atributo de clase se escibe:

**NombreClase.atributo**

**Un Método static**:

- Tiene acceso solo a los atributos estáticos de la clase.

- No es necesario instanciar un objeto para poder utilizarlo.

Para acceder a un método de clase se escribe:

**NombreClase.método()**

**Veamos otro Ejemplo:**

Vamos a escribir una clase Persona que contendrá un atributo contadorPersonas que indique cuantos objetos de la clase se han creado.

contadorPersonas debe ser un atributo de clase ya que no es un valor que se deba guardar en cada objeto persona que se crea, por lo tanto se debe declarar static:

**static int contadorPersonas;**

Un ejemplo de uso desde fuera de la clase Persona:

**System.out.println(Persona.contadorPersonas);**

Si lo declaramos como private:

**private static int contadorPersonas;**

Solo podremos acceder al atributo desde fuera de la clase a través de métodos static:

**public static void incrementarContador(){**

**contadorPersonas++;**

**}**

**public static int getContadorPersonas() {**

**return contadorPersonas;**

**}**

En este caso un ejemplo de uso puede ser:

**System.out.println(Persona.getContadorPersonas());**

Cada vez que se crea una persona se incrementará su valor.

Si no es private, desde otra clase podemos hacerlo así:

**Persona.contadorPersonas++;**

Si es private, desde otra clase debemos incrementarlo así:

**Persona.incrementarContador();**

**//Clase Persona**

**public class Persona {**

**private String nombre;**

**private int edad;**

**private static int contadorPersonas;**

**public Persona() {**

**}**

**public Persona(String nombre, int edad) {**

**this.nombre = nombre;**

**this.edad = edad;**

**}**

**public void setNombre(String nom) {**

**nombre = nom;**

**}**

**public String getNombre() {**

**return nombre;**

**}**

**public void setEdad(int ed) {**

**edad = ed;**

**}**

**public int getEdad() {**

**return edad;**

**}**

**public static int getContadorPersonas() {**

**return contadorPersonas;**

**}**

**public static void incrementarContador() {**

**contadorPersonas++;**

**}**

**}**

**//Clase Principal**

**public class Estatico1 {**

**public static void main(String[] args) {**

**Persona p1 = new Persona("Tomás Navarra", 22);**

**Persona.incrementarContador();**

**Persona p3 = new Persona("Jonás Estacio", 23);**

**Persona.incrementarContador();**

**System.out.println("Se han creado: " + Persona.getContadorPersonas() + " personas");**

**}**

**}**

En lugar de utilizar el método incrementarContador() cada vez que se crea un objeto, podemos hacer el incremento de la variable estática directamente en el constructor.

El código de la clase Persona y de la clase principal quedaría ahora así:

**//Clase Persona**

**public class Persona {**

**private String nombre;**

**private int edad;**

**private static int contadorPersonas;**

**public Persona() {**

**contadorPersonas++;**

**}**

**public Persona(String nombre, int edad) {**

**this.nombre = nombre;**

**this.edad = edad;**

**contadorPersonas++;**

**}**

**public void setNombre(String nom) {**

**nombre = nom;**

**}**

**public String getNombre() {**

**return nombre;**

**}**

**public void setEdad(int ed) {**

**edad = ed;**

**}**

**public int getEdad() {**

**return edad;**

**}**

**public static int getContadorPersonas() {**

**return contadorPersonas;**

**}**

**}**

**//Clase Principal**

**public class Estatico1 {**

**public static void main(String[] args) {**

**Persona p1 = new Persona("Tomás Navarra", 22);**

**Persona p3 = new Persona("Jonás Estacio", 23);**

**System.out.println("Se han creado: " + Persona.getContadorPersonas() + " personas");**

**}**

**}**