Atributos y Métodos

Material de clase elaborado por Sandra Victoria Hurtado Gil

1. Atributos

Un atributo (o **variable de instancia**) es la definición de una característica de los objetos de una clase. Es decir, un atributo representa una característica que tendrán los objetos. Por ejemplo, algunos atributos de una clase carro pueden ser el color, la placa, la marca, etc.

Para definir un atributo en una clase son indispensables dos elementos:

- *El nombre*: Es la denominación dada a cada atributo, normalmente es un sustantivo. Debe ser un nombre significativo.
- *El tipo:* Cada atributo debe ser de un tipo de dato definido. Estos tipos de datos dependen del lenguaje de programación seleccionado. Por ejemplo, en Java se tienen tipos de datos como *int* (número entero), *double* (número real), *boolean* (valor de verdad o booleano), *String* (cadena de caracteres), entre otros.

Identificación de atributos

Para establecer los atributos de las clases se tienen en cuenta todas las fuentes de información disponibles en el proyecto, ya sea un enunciado, entrevista, documento, diagrama, cuestionario u otro que describa lo que el sistema debe permitir. El sentido común y la experiencia también son factores para tener en cuenta a la hora de establecer los atributos de una clase.

Al definir la clase Persona, por ejemplo, es posible identificar muchas características: *número de identificación*, *nombres*, *apellidos*, *edad*, *sexo*, *grupo sanguíneo*, *etnia* y *número de hijos*. Aunque existen muchas otras características que podrían ser enunciadas, al modelar una clase el analista o desarrollador debe identificar aquellas que son **relevantes** para el **contexto** en el que se construye el sistema. Estas características relevantes serán los atributos de la clase.

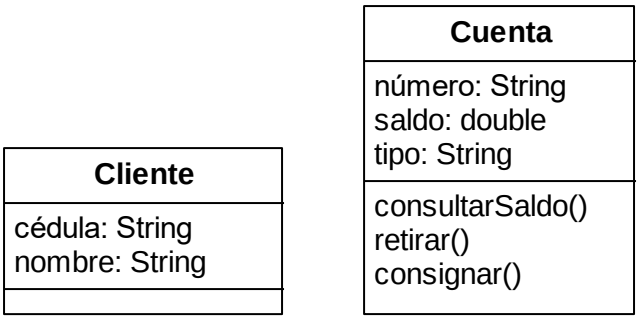
Por ejemplo, para un banco los atributos *identificación*, *nombres y apellidos* sean muy importantes, mientras que la *edad*, *grupo sanguíneo*, *etnia* y *número de hijos* no tengan la misma relevancia. Estas últimas serían muy importantes en otro contexto, por ejemplo, para un sistema médico.

Representación

Recordemos que las clases se representan con un rectángulo dividido en tres partes. La parte intermedia es el lugar dónde se definen los atributos. La forma de representar cada atributo en el diagrama es:

```
nombre: tipo
```

Por ejemplo (mostrando solo los atributos):



Implementación en Java

Los atributos se definen como variables, es decir:

```
tipoDeDato nombre;
```

Se recomienda que el nombre del atributo esté en minúscula, con mayúscula intermedia cuando está formado por varias palabras.

Por ejemplo:

```
/**
 * Una persona que tiene una cuenta en el banco, y por lo tanto
 * se considera cliente del banco.
 * @version 1.0
 */
public class Cliente
{
      String cedula;
      String nombre;
}
```

2. Métodos

Como se ha explicado previamente, los objetos, además de tener características e identidad, tienen comportamiento. El comportamiento se refiere a las acciones que puede hacer un objeto, o las acciones que se pueden hacer con él, incluyendo responder “preguntas” sobre sus características.

Cuando se modela un grupo de objetos mediante una clase, el comportamiento común que se desee representar de estos objetos se establece en los métodos. Por ejemplo, en un carro (automóvil) sus métodos son: encender, acelerar, frenar, cambiar una llanta, venderlo, etc.

Un método, por lo tanto, representa **una acción, comportamiento o servicio de los objetos**. También puede entenderse un método como una pequeña función o procedimiento que actúa sobre el objeto y sus atributos.

Para definir un método, los elementos son:

- *El nombre*: Por lo general es un verbo que explica la acción que se puede realizar.
- *Los parámetros*: Corresponden a información adicional que se envía al objeto para que pueda realizar la acción. No siempre es necesario enviar información adicional al objeto para usar un método, depende de lo que se desee pedir. Un método puede tener cero, uno o más parámetros.
- *El tipo de retorno*: Corresponde al tipo de información que se espera que devuelva (o "retorne") un objeto cuando se llame al método. También es posible que el método ejecute alguna acción y no retorne nada. Un método solo puede tener **un** valor de retorno (o ninguno).

Modelado de métodos

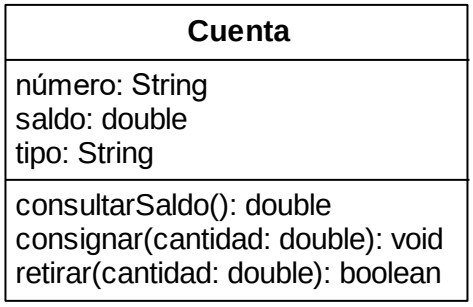
Para establecer los métodos de las clases se tienen en cuenta todas las fuentes de información disponibles en el proyecto para determinar qué acciones realiza cada objeto o que acciones se pueden realizar con ese objeto en el programa. Los métodos por lo general se relacionan con los atributos que tiene cada objeto, permitiendo consultarlos, modificarlos o hacer operaciones con ellos.

Al igual que con los atributos, solo se deben establecer los métodos que sean **relevantes** para el sistema que se esté modelando. Por ejemplo, en un almacén donde vendan carros, no son importantes los métodos acelerar o frenar del carro; sino otros métodos como definir el precio del carro, venderlo o asignarle la placa (si es nuevo).

Representación

En el tercer compartimiento del rectángulo que representa a una clase van los métodos, es decir, los servicios que tendrán los objetos de esa clase.

Por ejemplo:



La forma de representar cada método es:

```
nombre (parámetros): retorno
```

Un **parámetro** es una **variable** que contiene información adicional que necesita recibir el método para poder cumplir con el servicio que le están pidiendo. Un parámetro es un valor de "entrada" para el método.

Cada parámetro tiene un nombre y un tipo de dato, así:

```
nombre: tipo
```

Además, dependiendo del servicio que se le pida al objeto, él **retornará** algún valor o no.
En Java, si el método retorna algún valor se debe definir de qué tipo es, y si no retorna nada se escribe ***void***.

Implementación en Java

Después de tener identificados todos los métodos en el diagrama de clases, se puede elaborar el código correspondiente en un lenguaje de programación como Java.

Los métodos en el lenguaje Java tienen dos partes:

- Un encabezado (también llamado firma o interfaz del método): tiene la información general del método, que es igual a la que está en el diagrama, solo que escrito con la notación del lenguaje de programación.
- Un cuerpo: es donde se escribe el código, que se ejecutará cada vez que llamen al método. Es decir, es el "corazón" del método, donde se define su comportamiento, las acciones que realiza con el objeto y sus atributos.

- **El encabezado, firma o interfaz del método**

El encabezado del método es donde se indica su nombre, los parámetros que recibe y el tipo de dato que retorna. La forma de definirlo es:

```
/**
 * Descripción del método
 * @param   nombreParámetro   explicación
 * @return  explicación
 */
tipoRetorno nombreMétodo(parámetros)
{

}
```

- Lo primero que debe ir es el comentario que explica en qué consiste el método, los parámetros que recibe y lo que retorna.
    - Para cada parámetro se usa un *@param*, seguido del nombre del parámetro y luego una explicación del mismo.
    - Para lo que retorna se usa *@return*, seguido de una explicación de lo que retorna.
- Luego va la línea de encabezado del método, también llamada la interfaz o la firma del método, así:
    - Primero va el tipo de retorno (como *int*, *double* o *void*).

- Luego sigue el nombre del método, con la primera letra en minúscula.
- Luego van los parámetros entre paréntesis.
  - Si no recibe parámetros se colocan los paréntesis vacíos.
  - Si van varios parámetros se separan por comas.
  - Para cada parámetro va primero el tipo de dato y luego el nombre.

- **El cuerpo**

El cuerpo de cada método tiene las instrucciones Java para hacer las acciones que se requieren. El cuerpo de los métodos siempre va enmarcado o encerrado por llaves; esto permite diferenciar los diferentes métodos que tenga una clase.

Por ejemplo, los métodos “consignar” y “retirar”:

```
/**
 * Consigna o adiciona una cantidad de dinero en la cuenta,
 * lo cual incrementa el saldo.
 * @param  cantidad    la cantidad de dinero que se desea
 *                      consignar, en pesos
 */
void consignar(double cantidad) {
      saldo = saldo + cantidad;
}
```

```
/**
 * Retira o saca una cantidad de dinero de la cuenta,
 * si hay saldo suficiente.
 * @param  cantidad  la cantidad que se desea retirar, en pesos.
 * @return si se pudo hacer el retiro porque tenía dinero
 *          suficiente o no (true o false)
 */
boolean retirar(double cantidad) {
      if (saldo >= cantidad) {
            saldo = saldo - cantidad;
            return true;
      }
      else {
            return false;
      }
}
```

Es importante tener en cuenta:

- Si un método tiene en su encabezado algún tipo de retorno, siempre debe tener una instrucción *return*.
- La instrucción *return* termina la ejecución del método, incluso si está dentro de un ciclo.

A continuación, se presenta el código completo de la clase Cuenta, como referencia:

```java
/**
 * Una cuenta bancaria, es decir, un registro en
 * una entidad bancaria que tiene un saldo (cantidad de dinero)
 * que se puede modificar mediante retiros o consignaciones.
 * @version 1.5
 */
public class Cuenta {
	String numero;
	double saldo;
	String tipo;

	/**
	 * Consulta el saldo actual de la cuenta (el dinero que tiene).
	 * @return  el valor del saldo, en pesos.
	 */
	double consultarSaldo() {
		return saldo;
	}

	/**
	 * Retira o saca una cantidad de dinero de la cuenta,
	 * si hay saldo suficiente.
	 * @param  cantidad  la cantidad que se desea retirar, en pesos.
	 * @return si se pudo hacer el retiro porque tenía dinero
	 *          suficiente o no (true o false)
	 */
	boolean retirar(double cantidad)  {
		if (saldo >= cantidad) {
			saldo = saldo - cantidad;
			return true;
		}
		else {
			return false;
		}
	}

	/**
	 * Consigna o adiciona una cantidad de dinero en la cuenta,
	 * lo cual incrementa el saldo.
	 * @param  cantidad    la cantidad de dinero que se desea
	 *                      consignar, en pesos
	 */
	void consignar(double cantidad) {
		saldo = saldo + cantidad;
	}
} // fin clase Cuenta
```