Programación Orientada a Objetos (POO II)

Objetos

Los objetos tienen tres características fundamentales:

- **Comportamiento:** ¿qué puede hacer este objeto? O lo que es lo mismo: ¿qué métodos se le pueden aplicar?
- **Estado del objeto:** ¿cómo reacciona el objeto cuando le aplicamos estos métodos?
- **Identidad del objeto:** ¿cómo distinguimos este objeto de otros que tengan el mismo comportamiento y estado?

Todos los objetos que son instancias o ejemplares de la misma clase tienen un parecido familiar, al admitir el mismo comportamiento. **El comportamiento de un objeto se define mediante los métodos** que se puedan aplicar al mismo.

El objeto almacena información relativa al aspecto que tiene en ese momento. Es lo que se denomina estado del objeto. El estado del objeto puede cambiar con el tiempo, pero siempre obedeciendo a llamadas de sus métodos.

El estado de un objeto puede influir sobre su comportamiento.

Objetos y variables objeto

Para trabajar con los objetos, lo primero es construirlos y especificar su estado inicial. Después se les aplican métodos a los objetos.

En Java se utilizan constructores para crear nuevos objetos. Un constructor es un método especial cuya finalidad es construir objetos y darles valores iniciales. Los constructores tienen el mismo nombre que la clase.

Pongamos un ejemplo. para construir un constructor de la clase **Date** se combina el nombre de la clase con la palabra reservada **new**.

Ejemplo:

new Date() // Esta expresión construye un nuevo objeto. El objeto recibe como valor inicial la fecha y hora actuales.

Se puede pasar el objeto a un método:

Ejemplo:

System.out.println(***new Date()***);

Ejemplo con la clase GregorianCalendar

La clase GregorianCalendar permite manejar fechas en Java y constituye un buen ejemplo para ver cómo construir un objeto, darle estado inicial y manipular sus propiedades. La expresión **new GregorianCalendar()** construye un nuevo objeto que representa la fecha y la hora en la que se ha construido el objeto.

Es posible construir un objeto calendario para la medianoche de una determinada fecha, proporcionando el año, mes y día correspondientes:

new GregorianCalendar(1999,11,31); // con esta expresión estamos llamando al constructor de la clase y dando un estado inicial al objeto (en este caso el objeto sería la fecha) concretamente 31 de Diciembre de 1999.

Los meses se cuentan a partir de 0 por lo que Diciembre será el mes 11. Existen constantes del tipo Calendar.DECEMBER por ejemplo y así para todos los meses: **new GregorianCalendar(1999, Calendar.DECEMBER, 31);**

También se puede especificar la hora:

New GregorianCalendar (1999, Calendar.DECEMBER, 31, 23, 59, 59); // La clase **GregorianCalendar** cuenta con sobrecarga de constructores. Dependiendo del estado inicial que queramos dar al objeto, utilizaremos un constructor u otro.

Si necesitamos almacenar el objeto fecha en una variable objeto, el código sería el siguiente:

GregorianCalendar mifecha= new GregorianCalendar(……); // donde **"mifecha"** es el nombre que se ha decidido dar al nuevo objeto.

Constructor Summary

Constructors

Constructor and Description
`GregorianCalendar()` Constructs a default `GregorianCalendar` using the current time in the default time zone with the default locale.
`GregorianCalendar(int year, int month, int dayOfMonth)` Constructs a `GregorianCalendar` with the given date set in the default time zone with the default locale.
`GregorianCalendar(int year, int month, int dayOfMonth, int hourOfDay, int minute)` Constructs a `GregorianCalendar` with the given date and time set for the default time zone with the default locale.
`GregorianCalendar(int year, int month, int dayOfMonth, int hourOfDay, int minute, int second)` Constructs a GregorianCalendar with the given date and time set for the default time zone with the default locale.
`GregorianCalendar(Locale aLocale)` Constructs a `GregorianCalendar` based on the current time in the default time zone with the given locale.
`GregorianCalendar(TimeZone zone)` Constructs a `GregorianCalendar` based on the current time in the given time zone with the default locale.
`GregorianCalendar(TimeZone zone, Locale aLocale)` Constructs a `GregorianCalendar` based on the current time in the given time zone with the given locale.

Sobrecarga de constructores en GregorianCalendar

Métodos de consulta y modificación

¿Cómo conseguir averiguar el día, mes o año de un objeto de tipo GregorianCalendar? ¿Cómo cambiamos esos valores?

La forma de conseguir estos propósitos es consultar lo documentación de la API en línea para ver que métodos hay disponibles para la clase GregorianCalendar.

Para consultar los valores de un objeto GregorianCalendar se utiliza normalmente el método get. Para seleccionar el elemento que queremos averiguar, utilizamos una de las constantes definidas en la clase Calendar como por ejemplo Calendar.MONTH o Calendar.DAY_OF_WEEK.

- **GregorianCalendar ahora=new GregorianCalendar(); //** creación del objeto
- **int mes =ahora.get(Calendar.MONTH); //** obtención del mes accediendo a método
- **int diaSemana=ahora.get(Calendar.DAY_OF_WEEK); //** obtención del día de la semana accediendo a método.

La API enumera todas las constantes que se pueden utilizar:

static int	JANUARY Value of the MONTH field indicating the first month of the year in the Gregorian and Julian calendars.
static int	JULY Value of the MONTH field indicating the seventh month of the year in the Gregorian and Julian calendars.
static int	JUNE Value of the MONTH field indicating the sixth month of the year in the Gregorian and Julian calendars.
static int	LONG A style specifier for getDisplayName and getDisplayNames indicating a long name, such as "January".
static int	MARCH Value of the MONTH field indicating the third month of the year in the Gregorian and Julian calendars.
static int	MAY Value of the MONTH field indicating the fifth month of the year in the Gregorian and Julian calendars.
static int	MILLISECOND Field number for get and set indicating the millisecond within the second.
static int	MINUTE Field number for get and set indicating the minute within the hour.
static int	MONDAY Value of the DAY_OF_WEEK field indicating Monday.
static int	MONTH Field number for get and set indicating the month.
static int	NOVEMBER Value of the MONTH field indicating the eleventh month of the year in the Gregorian and Julian calendars.
static int	OCTOBER Value of the MONTH field indicating the tenth month of the year in the Gregorian and Julian calendars.
static int	PM Value of the AM_PM field indicating the period of the day from noon to just before midnight.
static int	SATURDAY Value of the DAY_OF_WEEK field indicating Saturday.
static int	SECOND Field number for get and set indicating the second within the minute.
static int	SEPTEMBER Value of the MONTH field indicating the ninth month of the year in the Gregorian and Julian calendars.

Algunos de los campos de la clase Calendar

El estado se modifica con el método set, por ejemplo:

- **fechalimite.set(Calendar.YEAR,2001);**
- **fechalimite.set(Calendar.MONTH,Calendar.APRIL);**
- **fechalimite.set(Calendar.DAY_OF_MONTH, 15);**

Existe un método para fijar el año, mes y día con una sola llamada:

- **fechalimite.set (2001, Calendar.APRIL, 15);**

Otro método permite añadir un número determinado de días, semanas, meses a un determinado objeto de calendario:

- **fechalimite.add(Calendar.MONTH, 3);** // se desplaza tres días. Si se añade un número negativo, se desplaza al calendario hacia atrás.

El método get solo busca el estado del objeto y lo notifica.

Los métodos set y add modifican el estado del objeto.

Existe una convención para anteponer a los métodos de acceso el prefijo **get** y anteponer a los de modificación el prefijo **set**, como por ejemplo **getTime** y **setTime**.

Ejemplo:

- **Date lahora=calendario.getTime(); //** este tipo de métodos que comienzan con el prefijo "get" son conocidos como métodos **"getters".**

- **Calendario.setTime(laHora); //** este tipo de métodos que comienzan con el prefijo "set" son conocidos como métodos **"setters".**