élogos

CLASES Y OBJETOS. ATRIBUTOS, MÉTODOS Y MIEMBROS ESTÁTICOS

élogos

ÍNDICE

CLASES Y OBJETOS. ATRIBUTOS, MÉTODOS Y MIEMBROS ESTÁTICOS

1. Definición formal de clase y objeto. Creación de objetos a partir de clases 3
2. Implementación de métodos y atributos de una clase 4
3. Modificadores de acceso 6

Clases y objetos. Atributos, métodos y miembros estáticos

1. Definición formal de clase y objeto. Creación de objetos a partir de clases

Definición de clase

Una clase la podríamos definir como una plantilla que contiene aquellas propiedades y métodos comunes a todos los objetos que pertenezcan a dicha clase.

En el caso de que quisiéramos implementar la clase Coche, pensaríamos mediante este proceso de abstracción en propiedades comunes a todos los coches, por ejemplo, color, número de puertas, tipo de motor, etc. y también en aquellos métodos comunes a todos los coches, tales como, acelerar, frenar, etc.

Al crear un objeto, dicho objeto adquirirá las propiedades y métodos de la clase a la cual pertenece y es aquí, en dicho objeto, donde las propiedades tomarán valor.

Operador New

Si queremos crear un nuevo objeto de cualquier clase, hemos de hacer uso de el operador new. La forma de utilizarlo es la siguiente:

```
NombreClase NombreNuevoObjeto = new NombreClase();
```

Ejemplos

String str = new String(); Random r = new Random();

Date fecha = new Date(27,4,2003);

Constructor

Constructor es un método propio de cada clase que nos permite crear un nuevo objeto de esa clase. Este método se llama exactamente igual que la clase, no retorna ningún valor.

La misión de un constructor es inicializar las variables o atributos para un objeto dado, por ello, los constructores pueden recibir parámetros:

```
public class Numero{

    private int x;
```

```
    public Numero(int a){
        x=a; //se inicializa el atributo x
    }
    :
}
```

La llamada a este constructor se produciría al crear un objeto de la clase Numero, utilizando una instrucción como la siguiente:

```
Numero num = new Numero (4);
```

Toda clase Java debe incluir al menos un constructor. Si no se define ninguno, el compilador incluirá un constructor por defecto dentro de la clase

Sobrecarga de métodos y constructores

La sobrecarga de métodos consiste en definir dentro una misma clase varios métodos con el mismo nombre, si bien, estos deberán diferenciarse en el número o tipo de parámetros recibidos. La sobrecarga también es aplicable a constructores.

2. Implementación de métodos y atributos de una clase

Acceso a las variables

A las variables que se definen dentro de una clase, fuera de los métodos, se les denomina atributos.

Desde el interior de cualquier de los métodos la clase se puede acceder a estas variables utilizando directamente su nombre. Para acceder desde fuera, se emplea la notación:

objeto.variable;

Clases y objetos. Atributos, métodos y miembros estáticos

Tipos de variables

Variables de instancia

Son aquellas que van dentro de la clase y también se denominan propiedades o atributos. No hace falta inicializarlas. Se necesita un objeto para acceder a ellas.

Variables de clase

Se conocen también como variables estáticas, pues su valor no depende de ningún objeto particular de la clase. Se definen con el modificador static:

static int colores;

Variables locales

Son las que van dentro de los métodos. Es obligatorio inicializarlas antes de usarlas.

void pintar()

{

String color=”Rojo”; System.out.println(color);

}

Acceso a los métodos de un objeto

Se accede también haciendo uso de la dot (notación “punto”) notation.

Objeto.metodo();

- Todos los métodos llevan los paréntesis asociados, con o sin parámetros.

- Igualmente como ocurría con las variables, también hay métodos de instancia y métodos de clase, con propiedades similares.

- Es decir, si un método lleva delante el modificador static significa que no es necesario tener un objeto para poder acceder a él.

Declaración completa de un método

La declaración completa de un método sería:

[Modificador] tipo_devuelto Nombre_método ([parámetros formales])

{

[return valor;]

}

Llamar a un método desde un programa

Para llamar a un método desde otro de su misma clase, se indicará simplemente el nombre del método .

Si queremos llamarlo desde otra clase, se utilizará la notación:

objeto.metodo(..);

O en el caso de que sea static:

NombreClase.metodo(..);

En las clases del API de Java, son muchos los métodos que están declarados como static, aquellos cuya ejecución no dependa de ningún objeto particular de a clase.

Asignar el valor de un objeto a otro

Cuando asignamos el valor de una variable objeto a otra variable del mismo tipo, no creamos una copia del objeto, sino que tenemos dos variables apuntando al mismo objeto. El motivo es que las variables objeto, como indicamos al principio del curso, no contienen el objeto en si, sino una referencia al mismo.

3. Modificadores de acceso

En las clases: public

Las clases declaradas como public son accesibles desde otras clases, bien sea directamente o bien mediante herencia, desde clases declaradas fuera del paquete que contiene a esas clases públicas debido a que, por defecto, las clases solamente son accesibles por otras clases declaradas dentro del mismo paquete en el que se han creado.

Para acceder desde otros paquetes, primero tienen que ser importadas. La sintaxis sería:

```
import nombre_paquete.NombreClase;
```

clases: Abstract, Final, Synchronizable

Ahora revisaremos las clases Abstract, Final, Synchronizable

Abstract:

Una clase abstract tiene al menos un método abstracto. Una clase abstracta no se instancia (esto significa que no pueden crearse objetos de dichas clases), sino que se utiliza como clase base para la herencia. Es el equivalente al prototipo de una función en C++. Un método abstracto es un método que no se puede redefinir.

Final:

Clase final es aquella que ya no puede tener clases derivadas por debajo de ella, es decir, no podría tener clases hijas, es lo contrario a una clase abstracta. Nadie puede heredar de una clase final. Aunque sería posible declarar clases con varias combinaciones de public, abstract y final, la declaración de una clase abstracta y a la vez final no tiene sentido, y el compilador no permitirá que se declare una clase con esos dos modificadores juntos.

Synchronizable:

Este modificador indica que todos los métodos definidos en la clase están sincronizados, es decir, que no se puede acceder al mismo tiempo a ellos desde distintos threads; el sistema se encarga de colocar los avisos o warnings necesarios para evitarlo. Este mecanismo hace que desde hilos diferentes se puedan modificar las mismas variables sin que haya problemas de que se sobrescriban.

En los atributos: propiedades de la clase y métodos

Los modificadores de acceso son palabras reservadas del lenguaje que nos permiten restringir la visibilidad y acceso a los elementos de nuestra solución de software. Existen 4 tipos de modificadores:

default: ámbito de paquete

public: Los elementos de tipo públicos son visibles desde cualquier parte, es decir se pueden leer o modificar desde cualquier clase que conforme la solución de software.

protected: Los elementos protegidos serán visibles para la misma clase y las clases que heredan de ésta.

private: Los elementos privados sólo son visibles dentro de la clase que fueron declarados