Clases, Objetos Atributos y Métodos Modificadores de acceso

Concepto de Clase.

Casi todo lo que nos rodea o que conocemos puede formar parte de una **Clase**. Una **Clase** agrupa una serie de características y acciones comunes de un determinado elemento u objeto de aplicación. El molde para hacer jarrones seria una **Clase** y un jarrón hecho con ese molde sería un **objeto**. Un tipo de marca de coche seria la **clase** y un coche de esa marca seria el **objeto**. Cuando hablemos de los objetos, se entenderá mejor el concepto de clase, aún así es un concepto que no entraña demasiada dificultad de entender.

Definición de una clase.

Para **definir una clase** se antepone la palabra reservada class seguida del nombre de la misma que por convención a de tener la primera letra del nombre en Mayúscula. Una clase se define atendiendo al siguiente modelo:

```
class MiPrimeraClase {
    // Variables miembro de la Clase.(Variables, constantes, etc)
    // Método o métodos constructores.
    // Otros métodos
}
```

En esta definición se puede apreciar que una clase engloba todos sus componentes y miembros entre corchetes. En la declaración o encabezado de la clase se ha omitido el tipo de **modificador de acceso** que antepone a la palabra **class**, así como otros conceptos que explicaremos de forma breve en el siguiente apartado. Como adelanto decir que Java asigna el modificador **public** por defecto cuando este lo hemos omitido en nuestra declaración de la clase.

Componentes de una Clase.

Una clase consta **en su cabecera** de la palabra **class** que indica que se esta definiendo una clase, del **nombre de la clase** a definir atendiendo a las convenciones o nomenclatura anteriormente señalada y tal como hemos adelantado en el apartado anterior también tenemos **modificadores de acceso** que explicaremos a continuación y otras **palabras reservadas** que se utilizan para indicar si la **clase hereda de otra clase** (superclase o clase Madre), si **implementa algún tipo de interfaz**, si es una **clase Abstracta**, etc.

De momento para seguir un sistema de aprendizaje progresivo y fácil de asimilar solo hablaremos de los **tipos de modificadores**, de las **variables miembro de la clase**, de las **variables locales** y de los **métodos de la clase** que podemos encontrar.

Modificadores de acceso: public, private, protected.

Un modificador de acceso tal y como se puede intuir por su propio nombre, regulan el acceso a los distintos campos o miembros de una clase, así como a la misma clase que contenga dicho modificador en su definición. Dicho esto tenemos que:

- El modificador **public** es de ámbito público, es decir se puede acceder a un miembro de la clase (atributo, método) con dicho modificador **desde cualquier clase**.
- El modificador **private** al contrario que el anterior es de ámbito privado, es decir **solo se puede acceder a un miembro de la clase desde la misma clase**.
- El modificador **protected** es de ámbito de paquete, es decir se puede acceder a un miembro de la clase (atributo, método) con dicho modificador **desde cualquier clase del mismo paquete**.

Ejemplo con modificadores:

```
package A;
        /**
         *
         * @author Miguel Angel
         */
    public class Clase_A {

                public String nombre;
                private String telefono;
                protected String ciudad;

            public static void main(String[] args) {

                  System.out.println("Nombre: "+nombre);
                  System.out.println("telefono: "+telefono);
                  System.out.println("Ciudad: "+ciudad);

            }
  }
```

```
package A;
        /**
         *
         * @author Miguel Angel
         */

    public class Clase_B {

            public static void main(String[] args) {

                  System.out.println("Nombre: "+nombre);
```

```
                System.out.println("Ciudad: "+ciudad);

        }



}
```

```
package B;
        /**
         *
         * @author Miguel Angel
         */

    public class Clase_C {

            public static void main(String[] args) {

                    System.out.println("Nombre: "+nombre);

            }

}
```

Según el ejemplo, vemos que en la clase **«Clase_A» del paquete «A»** se declaran los atributos nombre, telefono y ciudad. De acuerdo a las caracteristicas de cada modificador, desde esta clase se puede acceder a todos los atributos.

Desde la clase **«Class_B» del mismo paquete «A»** solo podemos acceder al atributo public «nombre» y al atributo protected «ciudad».

Y por último desde la clase **«Class_C» del paquete «B»** tan solo tendriamos acceso al atributo público «nombre», siempre y cuando hubiésemos importado la clase «Clase_A» al paquete B utilizando la sentencia **import** que veremos cuando realicemos algún programa en el entorno de desarrollo elegido ya sea NETBEANS o ECLIPSE.

Atributos o miembros de Clase.

En el cuerpo de una Clase podemos encontrar principalmente atributos, campos o variables de clase, también llamadas estas últimas variables globales. Sobre las variables ya comentamos bastante en la primera parte del curso Java básico y que podéis repasar en estas dos páginas tipos

primitivos y declaración de variables. Solamente vuelvo a recordar la diferencia entre variables globales o de clase y las variables locales cuyo ámbito seria el cuerpo de un método en particular.

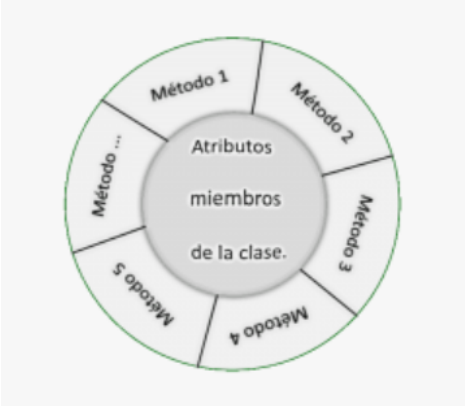
```
package A;
        /**
         *
         * @author Miguel Angel
         */

       public class Clase_A {

            public String nombre; //Atributo Global


      public static void main(String[] args) {

           String nombre; //Atributo local.
                         // Al ser local no se añade modificador.
           System.out.println("Nombre: "+nombre);

     }
}
```

Aparte de variables globales y locales una clase puede tener declaración o instancias de clase (Objetos), que en definitiva es la pieza principal del lenguaje Java. Sobre los Objetos hablaremos a continuación.

¿Qué es un objeto?

Un **objeto** (Instancia de clase) como ya podéis haber intuido si habéis leído el texto precedente es el elemento principal en la programación en Java. Se podría decir que es un cómputo de todas las características de la clase que lo crea, es decir, **un objeto tiene un estado inicial** regulado por los atributos de la clase y una actuación o acciones que se adquieren de la clase que le ha creado.

Una representación gráfica de un objeto bien podría ser la siguiente:



En el dibujo anterior se representa un objeto donde se aprecia un núcleo o círculo interno que

contiene los **atributos o miembros de la clase** y en el anillo externo se sitúan los métodos, tanto de acceso a los atributos del objeto, como otros métodos que den funcionalidad al objeto. Este dibujo solo es una mera representación simbólica de un objeto, que explica de forma gráfica el concepto de **encapsulación** de los atributos de un objeto.

Declarar y crear un objeto.

La declaración de un objeto se hace como cualquier otra variable, pero el tipo de dato en este caso sería una Clase:

```
ClaseDelObjeto objeto1;
```

Esta declaración solo tiene sentido si es para usarla en un contexto global, aunque lo más común es encontrar directamente la declaración y creación del objeto en una misma sentencia:

```
ClaseDelObjeto objeto1 = new ClaseDelObjeto()
```

Si hemos seguido el curso de javal esta sentencia nos resultará familiar, ya que la vimos con anterioridad en la creación de tablas o arrays. El motivo de esta coincidencia, es que **una tabla en si misma es un objeto** también, que puede almacenar otras variables incluso objetos. En esta sentencia de izquierda a derecha nos encontramos con una referencia a la clase «ClaseDelObjeto» seguida del nombre que queramos del objeto, que sigue la misma nomenclatura que la del nombre de las clases, con la diferencia que la primera letra obligatoriamente debe ser en letra minúscula. Luego nos encontramos con la palabra reservada «new» que crea propiamente dicho un nuevo objeto y por último nos encontramos con el constructor de la clase «ClaseDelObjeto()» que explicaremos en siguientes apartados y que es el que se encarga de inicializar dicho objeto ya sea con parámetros o sin ellos.

¿Qué es un método?

También llamado función en otros lenguajes de programación, los métodos son los encargados de dar funcionalidad al objeto y permitir que el mismo pueda invocar o mandar mensajes a otros objetos. Los métodos cumplen con distintas tareas como puede ser la tarea de constructor, el acceso a los atributos miembros de clase del objeto declarados como privados (encapsulamiento) o tareas de distinto índole. Cualquier método tiene esta **estructura básica**:

```
modificador retorno nombreDelMetodo (arg1, arg2, etc) {

    // código del método

}
```

Un claro ejemplo ya conocido es el **método main():**

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

```
    // Código o sentencias.

}
```

Donde vemos que el tipo de modificador es: **public**, el tipo de dato de retorno es: **void** (no devuelve nada), el nombre del método es: **main** y tiene un solo argumento que es un **array** tipo **String.** Los métodos pueden tener uno, varios o ningún argumento o parámetro según convenga y pueden devolver algún tipo de dato u objeto o no devolver nada (void). La nomenclatura de los métodos es prácticamente igual al de cualquier variable, es decir con la primera letra en minúscula y si es compuesto, ejemplo: «crearObjeto», la primera letra de la segunda y siguientes palabras estarían en mayúscula, ejemplo: «borrarObjetoMarca». Otra cuestión a tener en cuenta es que los nombres de los métodos deben reflejar acciones o verbos.

Método constructor de la clase.

Un **método constructor** es el encargado de crear el objeto conforme al molde que le proporciona la clase. Un método constructor no devuelve ningún valor y se nombra con el mismo nombre que tiene la clase:

```
public class miPrimeraClase {

          //Método constructor
          public miPrimeraClase() {

          }

}
```

Esta seria la definición básica de un constructor sin parámetros o argumentos. De forma implícita java proporciona un constructor similar (sin parámetros) en cualquier clase que no lo tenga, pero lo correcto es crear uno de forma explícita que evite posibles errores de compilación. Otro ejemplo de constructor, esta vez con parametros y haciendo referencia a los atributos miembro de la clase:

```
public class miPrimeraClase {

      String nombre;
      int edad;

      //Método constructor
      public miPrimeraClase( String nombre, int edad) {

        this.nombre=nombre;
        this.edad=edad;

     }
}
```

Nota: La palabra **this** se utiliza para hacer referencia al contexto o clase. En esta ocasión se emplea para dejar claro que la asignación de los datos que se pasan como parámetros del método, se hace a los miembros de clase nombre y edad.

También se podría designar los parámetros con otros nombres y no habría necesidad de incluir «this»:

```
public class miPrimeraClase {

          String nombre;
          int edad;

          //Método constructor
          public miPrimeraClase( String nomPersona, int edadPersona) {

             nombre=nomPersona;
             edad=edadPersona;

         }

}
```

Por buenas practicas de programación es mejor la variante del "this."

Sobrecarga de métodos.

java no permite que se creen dos métodos iguales, (con mismo nombre y parámetros), pero si permite crear dos o más métodos con mismo nombre y distinto número o tipo de argumentos. De esta forma el siguiente código seria correcto;

```
public class miPrimeraClase {

       String nombre;
       int edad;

       //Método constructor
       public miPrimeraClase( String nombre, int edad) {

         this.nombre=nombre;
         this.edad=edad;

       }

       //Un método cualquiera
       public void metodo(int edad) {
        ...
       }
```

```
        //El mismo método con distintos parámetros
        public void metodo(String nombre, int edad) {
         ...
        }

}
```

Bibliografía:

- Extraído de <<Curso de Java. Clases, Atributos, Modificadores, Objetos y Métodos.>>. Consultado el 28 de noviembre del 2022