

TEMA: [OBJETOS Y CLASES]

Unidad III: Clases y Objetos.

Programación Orientada a Objetos I
Nivel II
Facultad Ciencias de la Ingeniería

Ing. Jorge Vinueza Martínez, MgTI
jvinuezam@unemi.edu.ec



Milagro, Octubre 2018

Agenda

Mayo S5: 2018

Clases y Objetos

- ❑ Un programa utiliza una serie de objetos y, normalmente, muchos son iguales. La descripción (modelo) de un tipo de dichos objetos es una clase. Una aplicación se compone de una serie de clases, produciendo objetos que interactúan entre sí.

Clases y Objetos

Clase

Nombre de la clase



Vehículo

Característica de la clase

Marca
Referencia
Modelo
Placa
Color
Valor

Objetos de la Clase

Objeto 1



Objeto 2



Objeto 3



Cada objeto tiene valores propios para cada una de las características de la Clase.



Los **objetos** que existen en el mundo real pueden ser **conceptuales** o **físicos**. Este es un ejemplo de objetos físicos.

Clases y Objetos

Clase

Nombre de la clase



Cuenta bancaria

Característica de la clase



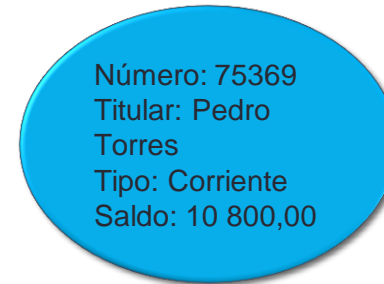
Número
Titular
Tipo
Saldo

Objetos de la Clase

Objeto 1



Objeto 2



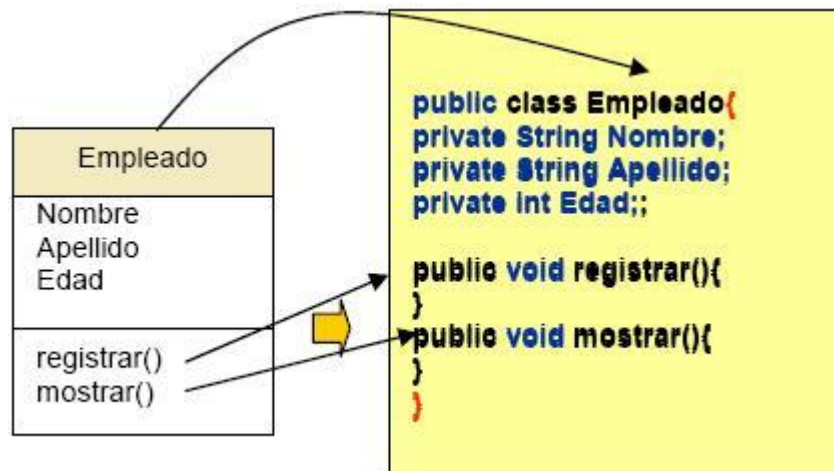
Objeto 3



Los **objetos conceptuales** existen en el mundo real pero no son tangibles. Las **características** de las clases que se presentan en los ejemplos son solo **algunas**, porque pueden tener **muchas más**.

¿Definición de la clase?

- ❑ La primera línea del programa, en la línea cuarta, después del comentario, define una clase que se llama `Hola` . La definición de la clase empieza en el carácter abre llave `{` y termina en el carácter cierra llave `}`. El nombre de la clase la puede elegir el programador. En el siguiente capítulo del libro verá todo lo relativo a la definición y uso de clases. También se suele llamar declaración de la clase.



Una clase en Java

- En el caso de ser una clase distinta de la clase principal, el método `main()` no suele aparecer. Una clase es la descripción (modelo) de un tipo de objetos. Una aplicación, un programa, se compone de un conjunto de clases, que crean objetos que interactúan entre sí. El nombre de la clase se empezará, como convenio de programación, en mayúscula. Una vez que la clase está disponible, el programador puede instanciar, crear, objetos de dicha clase. Por ello, los términos “objeto” e “instancia de clase” o, simplemente, “instancia” se usan como sinónimos.

```
/**
 *
 * Estructura de una clase en java
 */
public class NombreClase {
    //Declaración de atributos de la clase

    //Declaración de los métodos de la clase

    //Método main, que indica donde empieza la ejecución.
    public static void main(String[] args) {
        //Declaración de variables del método

        //Sentencias de ejecución
    }
}
```

Definición de Objeto

- ❑ Un objeto es la referencia e instancia de una clase. Al crear una referencia se asigna un espacio de memoria dinámica al objeto, pero no es utilizable. Al crear la instancia, el objeto es utilizable. La sintaxis de la referencia es la siguiente.

MiClase **m**;

- ❑ Donde m es la referencia del objeto. La sintaxis de la instancia es:

m = **new** **MiClase**();

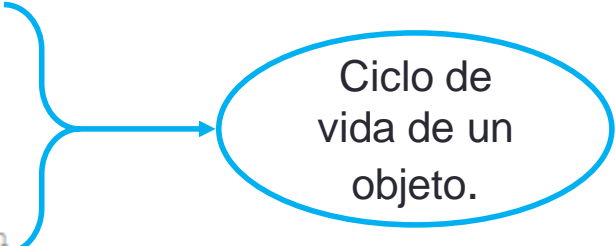
- ❑ Al hacer la instancia se puede acceder a los atributos y métodos públicos y protegidos si aplica, a través del objeto **m**. Otra sintaxis para realizar referencia e instancia en la misma línea de código es:

MiClase **m** = **new** **MiClase**();

Declaración de un objeto

- ❑ Cuando los objetos se quedan sin referencias dejan de ser accesibles por el programa. A partir de ese momento, el sistema puede reclamar sus recursos (la memoria que ocupa). Por ello, se habla del ciclo de vida de un objeto, que consta de las siguientes fases: definición, creación, uso, desaparición.

```
//Método main, que indica donde empieza la ejecución.  
public static void main(String[] args) {  
    NombreClase objeto;           //Definición  
    objeto = new NombreClase();    //Creación  
    objeto.ponGrupo("33", Horario.Tarde); //Uso  
    System.out.println(objeto.nombre); //Uso  
    //...  
}
```



Ciclo de vida de un objeto.

- ❑ El sistema decide cuando desaparecen los objetos que se han quedado sin referencias y decide cuando reclamar los recursos que fueron asignados en la creación del objeto. El programador no tiene control de cuando se liberaran los recurso ocupados por un objeto ni tiene que preocuparse por ello.

Atributos

- ❑ Los atributos son los elementos que definen el estado de un objeto, se definen de la misma forma que las variables, pero en el ámbito del bloque de una clase. De esta forma, cualquier método que sirva para dar un comportamiento al objeto puede acceder a cualquiera de los atributos del mismo.
- ❑ Existen dos tipos de atributos: los atributos de clase y los atributos de instancia (o de objeto).
- ❑ Los atributos de clase existen independientemente de si se han creado objetos de una determinada clase o no. Se identifican por el modificador **static**.
- ❑ Los atributos de instancia, o de objeto, tienen un ciclo de vida asociado al del objeto al que pertenecen
- ❑ Se define de la siguiente forma:

acceso tipo **nombre = valorInicial;**

- En la clase Alumno existen atributos que son tipos primitivos, como el año de nacimiento que es un entero del tipo **int** , y existen atributos que son referencias a objetos, como el nombre que es una referencia a un objeto de la clase **String** . Los atributos de una clase pueden tener asignado un valor inicial, definido junto a la declaración del atributo, como se muestra a continuación:

acceso **tipo** nombre = **valorInicial**;

```
public class NombreClase {  
    //Declaración de atributos de la clase  
    private String nombre = null;  
    String apellido;  
    protected int edad;  
    public Horario horario = Horario.Mañana;  
}
```

Métodos

- ❑ Los métodos son funciones que determinan el comportamiento de los objetos. Un objeto tendrá un comportamiento u otro según los métodos de que disponga. Los métodos se declaran y definen en las clases. Así, cualquier objeto de esa clase tendrá disponibles esos métodos y podrán ser invocados.

```
acceso tipo nombre(parámetros){  
    ...  
    ...  
}
```

Un método tiene dos partes claramente diferenciadas, la cabecera y el cuerpo. La cabecera está formada por:

- ❑ Accesibilidad del método. De momento se declara *public*.
- ❑ Tipo del valor a devolver: si el método devuelve un valor, hay que poner el tipo que se devuelve. Si no devuelve valor, se pone *void*.

Método sin retorno

```
public class Array {  
    public void imprime() {  
        System.out.println("Este es un metodo sin retorno");  
    }  
}
```

Método con retorno

```
public class Array {  
    public String imprime() {  
        return "Este es un metodo con retorno";  
    }  
}
```

- ❑ Nombre del método. El identificador con el que se invoca o usa el método. Debe ser descriptivo de lo que hace el método, pues facilita en gran medida la legibilidad de los programas.
- ❑ Parámetros: los parámetros que requiere el método para su ejecución. Son valores que se utilizarán en el código que compone el cuerpo del método. Los parámetros aparecen encerrados entre paréntesis.
- ❑ Tipo de excepción: si el método puede lanzar excepciones, hay que indicarlo.

```
public int calcularEdad(int año){  
    ...  
    ...  
}
```

Modificadores de acceso

- ❑ Un modificador es el método que permite asignar valor a un atributo con visibilidad **private** al aplicar el concepto de **encapsulamiento**.

```
/**
 *
 * @author Ariel
 */
public class Cuadrado {
    private int lado; // atributo en privado
    //Constructor Vacío
    public Cuadrado() {
    }
    // Constructor con Sobrecargado
    public Cuadrado(int lado) {
        this.lado = lado;
    }
    // Metodo Consultor
    public int getLado(){
        return this.lado;
    }
    // Metodo Modificador
    public void setLado(int lado) {
        this.lado = lado;
    }
    // Metodos Analizadores
    public int areaCuadrado() {
        int area= this.lado*this.lado;
        return area;
    }
    public int perimetroCuadrado() {
        int perimetro=4*this.lado;
        return perimetro;
    }
}
```

El objeto Array / Vector

- ❑ En muchos programas existe la necesidad básica de disponer de muchas variables u objetos del mismo tipo o de la misma clase. En este caso, es necesario poder llamar a todo el grupo de variables o de objetos con un único nombre para todos ellos.
- ❑ En java un Array se declara de la siguiente forma:

```
int[] nombreArray1;  
int nombreArray2[];
```

- ❑ Ambas declaraciones son equivalentes. La primera línea declara un array de enteros de nombre nombreArray1 . Es decir, nombreArray1 es una referencia a una estructura que puede guardar muchos valores del tipo int . La segunda línea declara que cada elemento de la forma nombreArray2[] es del tipo int . Es decir, que cada uno de los elementos de la estructura nombreArray2 es un entero. Se denomina tipo base del array al tipo de los elementos del mismo. No existe ninguna restricción al tipo base de un array.

A continuación puede ver un programa que declara un array de enteros, asigna los números naturales del uno en adelante a los elementos del array, suma todos los elementos del array y, por último, imprime por pantalla la suma total.

```
public class Array {  
    public static void main(String[] args) {  
        int[] array = new int[50];  
        int suma;  
  
        for(int i=0; i<array.length; i++){  
            array[i] = i + 1;  
        }  
  
        suma = 0;  
        for (int v : array){  
            suma += v;  
        }  
  
        System.out.println("La suma es: "+suma);  
    }  
}
```


Para programar no basta con conocer la sintaxis, también debemos de saber como estructurarlo, como funciona cada método, que papel cumple. Por tal motivo es indispensable poder repasar cada una de las partes aprendidas, porque todas llevan su relación.

CONCLUSIONES



[1] Programación en JAVA (3a. ed.). Madrid, ES: McGraw-Hill España, 2009. ProQuest ebrary. Web. 21 October 2017.

Copyright © 2009. McGraw-Hill España. All rights reserved

Sánchez Allende, Jesús, and Fernández Manjón, Baltasar. Programación en JAVA (3a. ed.). Madrid, ES: McGraw-Hill España, 2009. ProQuest ebrary. Web. 21 October 2017.

Copyright © 2009. McGraw-Hill España. All rights reserved

BIBLIOGRAFÍA



GRACIAS....!

Universidad Estatal de Milagro / FACI Facultad Ciencias de la

Ingeniería

(Haga clic en la flecha cuando se encuentre en el modo P)

