SOBRE CARGA DE MÉTODOS

Agosto, 2018
Guadalaiara. Ialisco. México

La sobre carga de métodos es el primer tema que estudiaremos de la programación orientada a objetos, porque nos permite tener en un mismo método variantes de ese comportamiento. Se le llama sobre carga de métodos porque en una misma clase se van a definir cada una de las variantes que queramos que el método tenga.

Las características para exista sobre carga de métodos en un programa son las siguientes:

- Los métodos sobre cargados se definen dentro de una misma clase
- Cada definición debe tener parámetros diferentes
- Deberán tener el mismo identificador
- Y por supuesto, deberán tener diferente código

En el lenguaje de java existen muchas clases que los creadores del lenguaje programaron que tienen sobre carga de métodos. Si revisas las clases que se encuentran en el API de java encontraras por cada clase métodos con las características descritas arriba.

La sobre carga de métodos también se puede aplicar en el constructor de la clase, esto sucede cuando queremos tener variantes en la inicialización de los atributos del objeto. Recordemos que la principal finalidad de un objeto es inicializar los valores del mismo. Por lo tanto por cada variante o por cada constructor sobre cargado estaríamos creando un objeto con un espacio en memoria diferente.

¿Cómo saber que en una clase existe sobre carga de métodos o sobre carga de constructores? Pues es sencillo tenemos que hacer referencia a sus cuatro características. Si el mismo identificador del método o constructor se encuentra definido varias veces dentro de la misma clase pero con diferentes parámetros y diferente código, entonces ese es un método o constructor sobre cargado.

Es necesario que tengan diferentes parámetros porque de esta manera el compilador puede diferenciar un método de otro ya que deben tener el mismo identificador.

SOBRE CARGA DE MÉTODOS

EJEMPLO SOBRE CARGA DE MÉTODOS

Ahora vamos a estudiar cómo implementar la sobre carga de métodos. Supongamos que de acuerdo al análisis del problema tenemos como resultado el siguiente esquema preliminar de clases:

SumaGenerica
- num1 : entero - num2 : entero
+ suma ( ) : void + suma (real n1) : void + suma (real n2, n3) : void + capturaDatos ( ) : void

Hagamos el código en base al diseño en el lenguaje de java

Archivo: SumaGenerica.java

```
class SumaGenerica { //DEFINICIÓN CLASE
        //DECLARACIÓN ATRIBUTOS
        private int num1;
        private int num2;
        //DEFINICIÓN METODOS SOBRECARGADOS
        public void suma( ){
                System.out.println("Suma enteros: “ + (num1+num2) );
        }
        public void suma(double n1){
                System.out.println("Suma enteros y reales: “ +
(num1+num2+n1) );
        }
        public void suma(double n2, double n3){
                System.out.println("Suma reales: “ + (n2+n3) );
        }
        //DEFINICIÓN DE METODOS
        public void capturaDatos( ){
                System.out.println(“Escribe un numero entero: “);
                num1 = DatosEntrada.entero ( );
System.out.println(“Escribe otro numero entero: “);
                num2 = DatosEntrada.entero ( );
        }
```

Archivo: AppSumaGenerica.java

```
class AppSumaGenerica {
        //DEFINICIÓN FUNCIÓN PRINCIPAL
        public static void main(String[] args) {

        //INSTANCIA: declaración de la variable de referencia y creación del objeto
                SumaGenerica op=new SumaGenerica( );

                System.out.println("Sumas Genericas\n");
                op.capturaDatos ( ); //LLAMADA METODO
                //LLAMADA METODO SOBRECARGADO
                op.suma();
                op.suma(7.3);
                op.suma(3.6, 14.7);
        }
}
```

NOTA IMPORTANTE: Recuerda que para utilizar el método **DatosEntrada.entero ( )** debes tener la clase Datos Entrada con el método estático entero ( )

En el archivo U2-a0 DatosDeEntrada.pdf de la carpeta "recursos" dentro de la Unidad 2 de la plataforma podrás encontrar en la diapositiva 21 el ejemplo del archivo: DatosEntrada.java

Este archivo lo debes tener en la misma carpeta donde estás guardando tus archivos para los programas del problema en particular de esta manera puedes hacer uso de **DatosEntrada.entero ( )** sin que te marque error o que no lo encuentre al momento de compilar

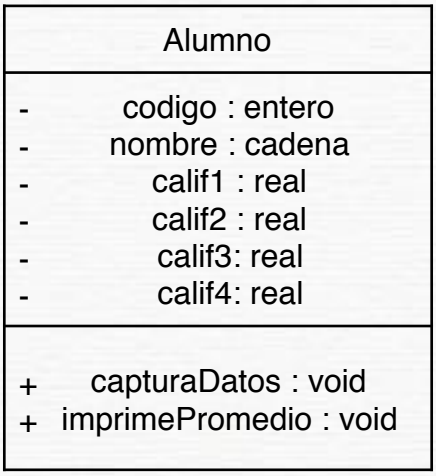
Otro punto importante es que si copias y pegas en el editor para java el archivo DatosEntrada.java tendrás que revisar que donde te marque error al compilar vuelvas a escribir pues la tipografía de la fuente no coincidirá. Se puede resolver escribiendo nuevamente la línea donde te marco el error. Pero, lo más recomendable es que crees un archivo y que tú escribas dentro del editor las líneas del archivo **DatosEntrada.java** que esta en el ejemplo del archivo **U2-a0 DatosDeEntrada.pdf** en la **diapositiva 21**

También puedes usar la clase Scanner con sus respectivos métodos para recibir datos de entrada en lugar de los mencionados en azul en este ejemplo. Para mayor referencia de cómo usar la clase Scanner puedes buscar a partir de la diapositiva 8 en el mismo archivo U2-a0 DatosDe Entrada.pdf

EJEMPLO SOBRE CARGA DE CONSTRUCTORES

Veamos otro ejemplo, ahora aplicando la sobrecarga de constructores.

Para este ejemplo también suponemos que ya hicimos el análisis del planteamiento del cual nos resulta el siguiente esquema preliminar de clases



En el esquema preliminar de clases no se ven reflejados los constructores pero estamos suponiendo que como un requerimiento del problema se encuentra que necesitamos un alumno con dos calificaciones, otro alumno con tres calificaciones y un alumno más con cuatro calificaciones.

Hagamos el código en base al diseño en el lenguaje de java

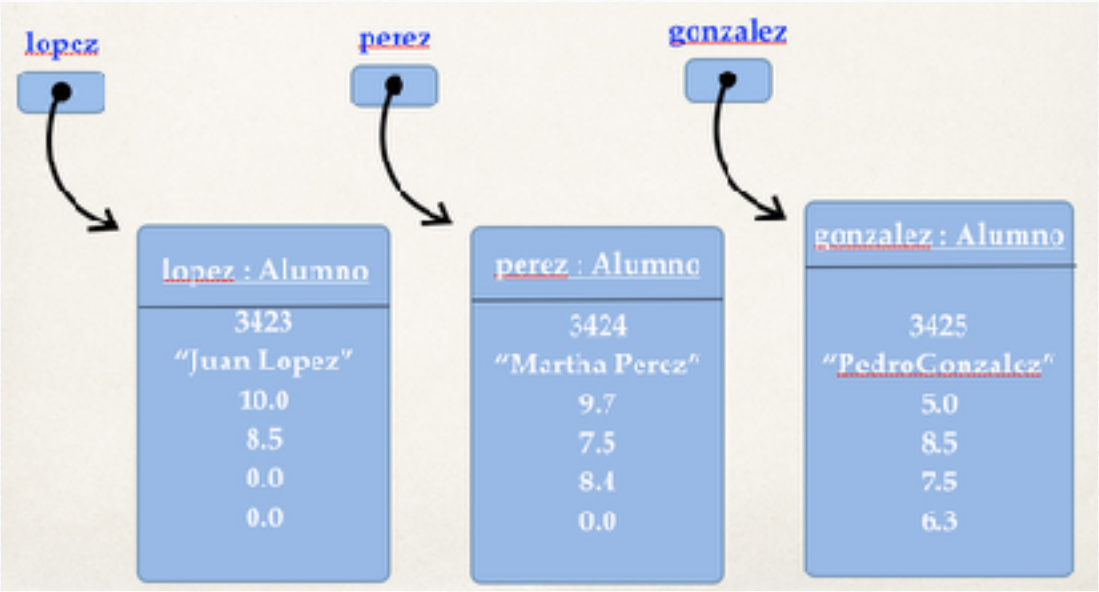
Archivo: Alumno.java

```
public class Alumno {
        private int codigo;
        private String nombre;
        private double c1=0.0, c2=0.0, c3=0.0, c4=0.0;
    //Constructores sobrecargados
        public Alumno(double c1, double c2){
                this.c1=c1;
                this.c2=c2;
        }
        public Alumno(double c1, double c2, double c3){
                this.c1=c1;
                this.c2=c2;
                this.c3=c3;
        }
        public Alumno(double c1, double c2, double c3, double c4 ) {
                this.c1=c1;
                this.c2=c2;
                this.c3=c3;
                this.c4=c4;
        }
        public void capturaDatos ( ) {
                System.out.println(“Escribe tu codigo: “);
                codigo = DatosEntrada.entero( );
                System.out.println(“Escribe tu nombre: ”);
                nombre = DatosEntrada.cadena( );
        }
        public void imprimirPromedio(String titulo){
                System.out.println(titulo);
                System.out.println("Codigo: "+codigo);
                System.out.println("Nombre: "+nombre);
                if(c3==0 && c4==0 )
                        System.out.println("Promedio: "+(c1+c2)/2);
                else if(c4==0)
                        System.out.println("Promedio: "+(c1+c2+c3)/3);
                else
                  System.out.println("Promedio: "+(c1+c2+c3+c4)/4);
        }
}
```

Archivo: AlumnoApp.java

```
class AlumnoApp {

public static void main(String[] args) {

        System.out.println (“Escribe la calificación 1: “);
        double calif1 = DatosEntrada.real ( );
        System.out.println (“Escribe la calificación 2: “);
        double calif2 = DatosEntrada.real ( );
        Alumno lopez=new Alumno(calif1, calif2);

        System.out.println (“Escribe la calificación 1: “);
        double calif1 = DatosEntrada.real ( );
        System.out.println (“Escribe la calificación 2: “);
        double calif2 = DatosEntrada.real ( );
        System.out.println (“Escribe la calificación 3: “);
        double calif3 = DatosEntrada.real ( );
        Alumno perez=new Alumno(calif1, calif2, calif3);

        System.out.println (“Escribe la calificación 1: “);
        double calif1 = DatosEntrada.real ( );
        System.out.println (“Escribe la calificación 2: “);
        double calif2 = DatosEntrada.real ( );
        System.out.println (“Escribe la calificación 3: “);
        double calif3 = DatosEntrada.real ( );
        System.out.println (“Escribe la calificación 4: “);
        double calif4 = DatosEntrada.real ( );
        Alumno gonzalez=new Alumno(calif1, calif2, calif3, calif4);

        //LLAMADAS METODOS
        lopez.capturaDatos ( );
        lopez.imprimirPromedio("Alumno con 2 calificaciones”);
        perez.capturaDatos( );
        perez.imprimirPromedio("Alumno con 3 calificaciones”);
        gonzalez.imrpimeDatos( );
        gonzalez.imprimirPromedio("Alumno con 4 calificaciones");
        }
}
```

Suponiendo también los datos de entrada que el usuario final podría escribir al ejecutar el programa, tendríamos en memoria tres instancias apuntando a tres objetos con las siguientes variantes en la inicialización de sus atributos:



EJERCICIO PARA LA ACTIVIDAD 1

I. PROBLEMÁTICA UNO

Se quiere determinar el ingreso para un jardinero. El método ingreso podrá ser:

1) para un jardin trabajado.
2) para un jardin trabajado y una poda de un árbol
3) para un jardín trabajado, la poda de un árbol y la aplicación de abono

Se sabe que el jardinero cobra diferente por cada actividad.

Los datos que se necesitan leer del jardinero son el nombre del jardinero. La cantidad a cobrar por el jardin y la cantidad a cobrar por el árbol y la cantidad a cobrar por la aplicación de abono se recibirá como parámetro en el método

Imprimir en pantalla el nombre del jardinero, el ingreso cuando trabaje sólo un jardín, el ingreso cuando trabaje un jardín y pode un árbol y el ingreso cuando trabaje un jardín y pode un árbol y aplique abono

Otro requerimiento es encontrar la solución con sobrecarga de métodos

II. PROBLEMÁTICA DOS

Se quiere trabajar con el objeto carro. Se usará la sobrecarga de constructores para obtener los siguientes resultados:

1) El método imprimeReporte imprimirá en pantalla los datos de la placa, el modelo, la marca, la submarca, el precio y el kilometraje si se manda llamar con una instancia creada para un carro usado.

2) Ese mismo método imprimeReporte imprimirá en pantalla los datos de la placa, el modelo, la marca, la submarca y el precio. Pero el dato de kilometraje será 0.0 si se manda llamar con otra instancia creada par aun carro nuevo

Definir dos constructores, uno con 6 parámetros para crear la instancia que corresponda al carro usado; y otro con 5 parámetros para crear la instancia que corresponda al carro nuevo.

Leer los datos desde la función principal para pasar los valores a través de los diferentes constructores al crear las dos instancias requeridas.