Capítulo 6

Programación orientada a objetos en java

6.1. Resultados de aprendizaje

4. Desarrolla programas organizados en clases, analizando y aplicando los principios de la programación orientada a objetos.

a) Se han reconocido la sintaxis, estructura y componentes típicos de una clase.

b) Se han definido clases.

c) Se han definido propiedades y métodos.

d) Se han creado constructores.

e) Se han desarrollado programas que instancien y utilicen objetos de las clases creadas anteriormente.

f) Se han utilizado mecanismos para controlar la visibilidad de las clases y de sus miembros.

g) Se han definido y utilizado clases heredadas.

h) Se han creado y utilizado métodos estáticos.

i) Se han definido y utilizado interfaces.

j) Se han creado y utilizado conjuntos y librerías de clases.

7. Desarrolla programas aplicando características avanzadas de los lenguajes orientados a objetos y del entorno de programación.

a) Se han identificado los conceptos de herencia, superclase y subclase.

b) Se han utilizado modificadores para bloquear y forzar la herencia de clases y métodos.

c) Se ha reconocido la incidencia de los constructores en la herencia.

d) Se han creado clases heredadas que sobrescriban la implementación de métodos de la superclase.

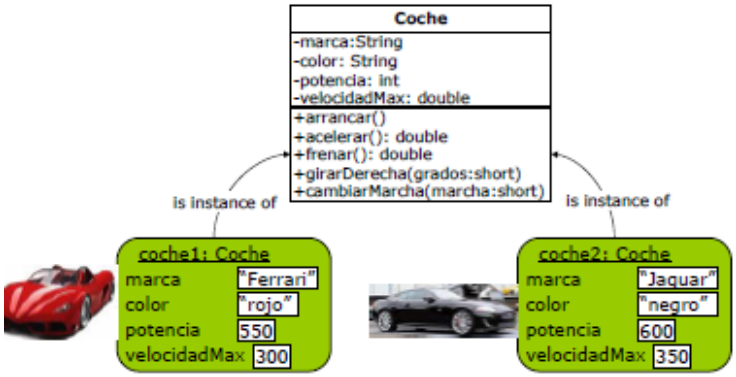
e) Se han diseñado y aplicado jerarquías de clases.

f) Se han probado y depurado las jerarquías de clases.

g) Se han realizado programas que implementen y utilicen jerarquías de clases.

h) Se ha comentado y documentado el código.


Programación Orientado a Objetos
Objetos
objeto1
color rojo
tipo deportivo
matricula 1234 ABC
velocidadMax 300
objeto2
color azul
tipo policiaco
matricula 0000 PM
velocidadMax 250
objeto3
color amarillo
tipo elegante
matricula 9876 DEF
velocidadMax 150
objeto4
color rojo
tipo utilitario
matricula 9235 FCB
velocidadMax 200


Coche
-marca:String
-color: String
-potencia: int
-velocidadMax: double
+arrancar()
+acelerar(): double
+frenar(): double
+girarDerecha(grados:short)
+cambiarMarcha(marcha:short)
is instance of
is instance of
coche1: Coche
marca "Ferrari"
color "rojo"
potencia 550
velocidadMax 300
coche2: Coche
marca "Jaguar"
color "negro"
potencia 600
velocidadMax 350

6.2. ¿ Cómo crear una clase en Java?

Para crear una clase en Java, debemos especificar el tipo y el nombre de los atributos y debemos especificar,si existen, los métodos o funciones, el tipo de dato que retornan, el nombre y los parámetros que reciben dichos métodos.

```
1
2  //Le damos un nombre "MiClase" a la clase
3  public class MiClase
4  {
5    //Atributos de la clase
6    private String atributo1;
7    private int atributo 2;
8    private float atributo 3;
9
10   //Constructor con el mismo nombre de la clase
11   public MiClase(){}
12
13   //Métodos de la clase
14   public void metodo1()
15   {
16     //Método vacio
17   }
18
19   public String metodo2()
20   {
21     return "metodo2";
22   }
23 }
```

En el ejemplo anterior hemos creado una clase en Java llamada MiClase la cual posee un total de tres atributos (todos ellos privados) y son de tipo String, int y float respectivamente. Adicionalmente esta

clase tiene un constructor (que siempre por norma, debe tener el mismo nombre de la clase) el cual no recibe ningún parámetro, aunque pueden recibir todos los parámetros que nosotros deseemos, también tiene un método llamado metodo1 que no retorna valor alguno (es de tipo void) y otro método llamado metodo2 que retorna una cadena de caracteres (es de tipo String) con el valor metodo2. Cabe resaltar que una clase en Java puede tener o no métodos y atributos, sin embargo lo más normal en la mayoría de los casos es que tenga tanto métodos como atributos que la caractericen.

En este otro ejemplo hemos creado para este ejemplo tiene como nombre Animal, pertenece al paquete misClases y posee los atributos raza, nombre y edad; tenemos un constructor que recibe un nombre y se lo asigna al animal y tres métodos encargados de obtener y establecer la edad del animal y el restante para obtener el nombre.

```
1  package misClases; //Se le declara un paquete
2
3  public class Animal
4  {
5    private String raza;
6    private String nombre;
7    private int edad;
8
9    public Animal(String nuevoNombre)
10   {
11     nombre = nuevoNombre; //Se le da un nombre al animal
12   }
13
14   //Método para obtener la edad del animal
15   public int getEdad()
16   {
17     return edad;
18   }
19
20   //Método para establecer la edad del animal
21   public void setEdad(int nuevaEdad)
22   {
23     edad = nuevaEdad;
24   }
25
26   //Método para obtener el nombre del animal
27   public String getNombre()
28   {
29     return nombre;
30   }
31 }
```

6.2.1. ¿ Cómo crear objetos en Java?

Al momento de crear objetos en Java, debemos tener claras dos cosas indispensables, la primera es el nombre de la clase para la cual vamos a crear el objeto y segundo el constructor que dicha clase posee, es decir, si el constructor recibe o no parámetros.

Para crear objetos en Java, el lenguaje nos proporciona el **comando new**, con este comando le decimos a Java que vamos a crear un nuevo objeto de una clase en especifico y le enviamos los parámetros (en caso de ser necesario) según el constructor, veamos un ejemplo.

```
1 MiClase miObjeto; //Declaramos una variable del tipo de la clase
2 miObjeto = new MiClase(); //Aquí ya hemos creado un objeto de MiClase
3
4 MiClase miObjeto = new MiClase();
```

```
1  import misClases.Animal; //Importamos la clase Animal para poder
       usarla
2
3  public class Ejemplo
4  {
5    public static void main(String[] args)
6    {
7      //Creamos un animal cuyo nombré será Falco
8      Animal miAnimal = new Animal("Falco");
9      //Le establecemos 3 años de edad a Falco.
10     miAnimal.setEdad(3);
11     //Mostraremos el nombre del animal por pantalla
12     System.out.println("El nombre es: " + miAnimal.getNombre());
13     //Mostramos la edad del animal por pantalla
14     System.out.println(+ " y tiene " + miAnimal.getEdad() + " años");
15     /Este código deberia imprimir "El nombre es: Falco y tiene 3
           años"
16   }
17 }
```

6.2.2. Sobrecarga de métodos y de constructores

La firma de un método es la combinación del tipo de dato que devuelve, su nombre y su lista de argumentos.

La sobrecarga de métodos es la creación de varios métodos con el mismo nombre pero con diferentes firma. Java utiliza el número y tipo de argumentos para seleccionar cuál definición de método ejecutar.

Java diferencia los métodos sobrecargados con base en el número y tipo de argumentos que tiene el método y no por el tipo que devuelve.

También existe la sobrecarga de constructores: Cuando en una clase existen constructores múltiples, se dice que hay sobrecarga de constructores.

Ejemplos:

```
1
2  /* Métodos sobrecargados */
3  int calculaSuma(int x, int y, int z){
4      ...
5  }
6  int calculaSuma(double x, double y, double z){
7      ...
8  }
9
10 /* Error: estos métodos no están sobrecargados */
11 int calculaSuma(int x, int y, int z){
12      ...
13 }
14 double calculaSuma(int x, int y, int z){
15     ...
16 }
```

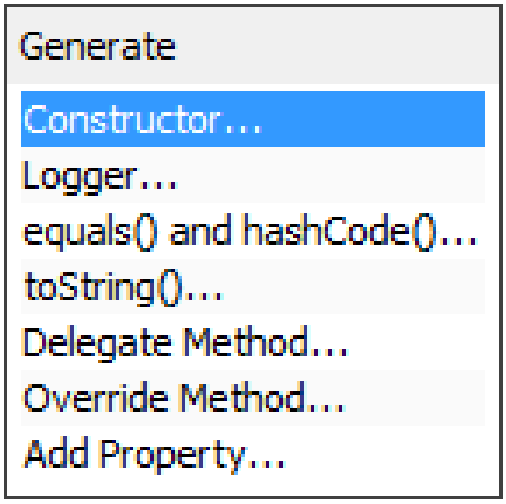
```
1
2  class Usuario4
3  {
4       String nombre;
5       int edad;
6       String direccion;
7
8       /* El constructor de la clase Usuario4 esta sobrecargado */
9       Usuario4( )
10      {
11         nombre = null;
12         edad = 0;
13         direccion = null;
14      }
15
16      Usuario4(String nombre, int edad, String direccion)
17      {
18         this.nombre = nombre;
19         this.edad = edad;
20         this.direccion = direccion;
21      }
22
23      Usuario4(Usuario4 usr)
24      {
25         nombre = usr.getNombre();
26         edad = usr.getEdad();
27         direccion = usr.getDireccion();
28      }
29
30      void setNombre(String n)
31      {
32         nombre = n;
33      }
34
35      String getNombre()
```

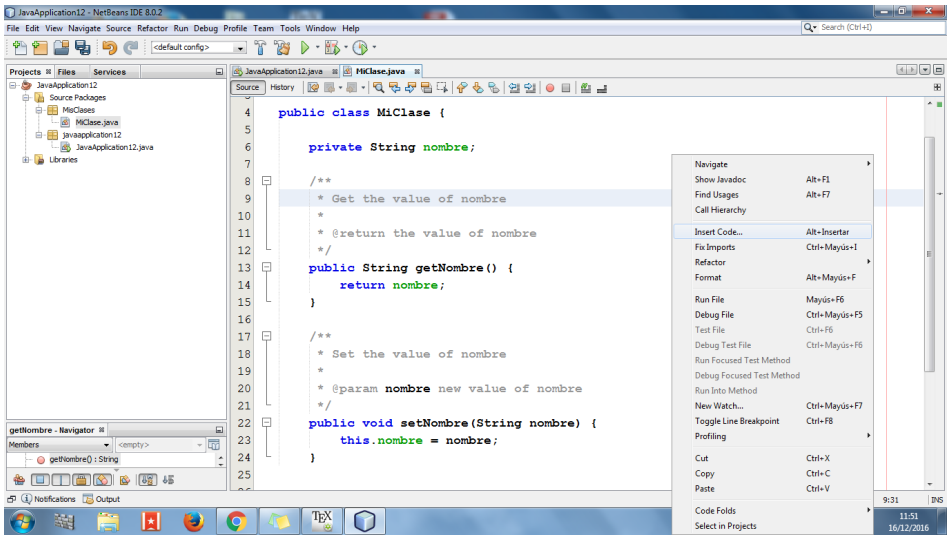
```
36        {
37           return nombre;
38        }
39
40        /* El metodo setEdad() está sobrecargado */
41        void setEdad(int e)
42        {
43           edad = e;
44        }
45
46        void setEdad(float e)
47        {
48           edad = (int)e;
49        }
50
51        int getEdad()
52        {
53           return edad;
54        }
55
56        void setDireccion(String d)
57        {
58           direccion = d;
59        }
60
61        String getDireccion()
62        {
63           return direccion;
64        }
65  }
```

6.2.3. Generación automática de código

Menú que aparece al pulsar el botón derecho del ratón:

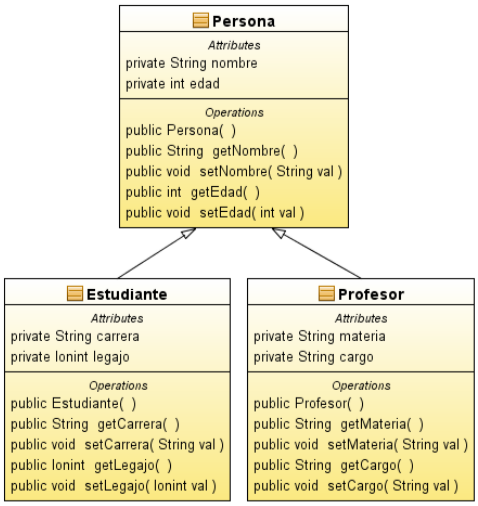
Añadiendo propiedad:


Generate
Constructor...
Logger...
equals() and hashCode()...
toString()...
Delegate Method...
Override Method...
Add Property...


public class MiClase {
private String nombre;
/**
* Get the value of nombre
*
* @return the value of nombre
*/
public String getNombre() {
return nombre;
}
/**
* Set the value of nombre
*
* @param nombre new value of nombre
*/
public void setNombre(String nombre) {
this.nombre = nombre;
}

6.3. Codificando las relaciones

Relación de generalización o herencia:**extends**

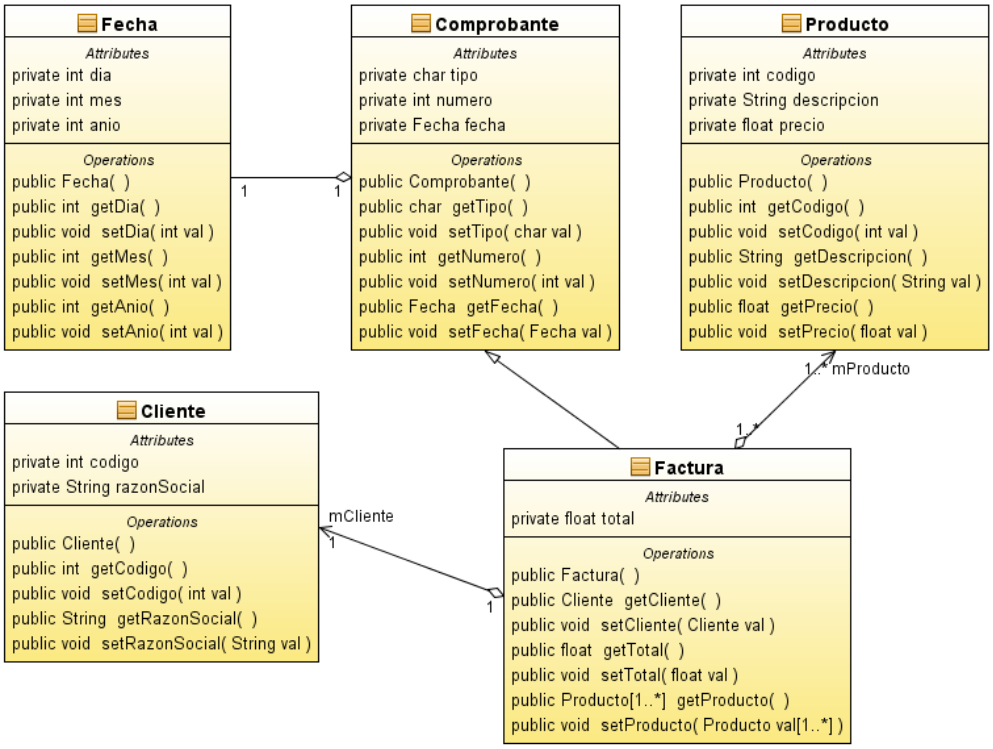


```java
public class Persona {
    private String nombre; // propiedad
    private int edad; // propiedad

    public Persona () { // constructor
    }
    public int getEdad () { // asesor a edad
        return edad;
    }
    public void setEdad (int val) { // mutador de edad
        this.edad = val;
    }
    public String getNombre () { // asesor a nombre
        return nombre;
    }
    public void setNombre (String val) { // mutador de nombre
        this.nombre = val;
    }
}
```

```java
public class Estudiante extends Persona {

    private String carrera; // propiedad
    private int legajo; // propiedad

    public Estudiante () { // constructor
    }
    public String getCarrera () { // asesor a carrera
        return carrera;
    }
    public void setCarrera (String val) { // mutador de carrera
        this.carrera = val;
    }
    public int getLegajo () { // asesor a legajo
        return legajo;
    }
    public void setLegajo (int val) { // mutador de legajo
        this.legajo = val;
    }
}
```

```
public class Profesor extends Persona {

    private String materia; // propiedad
    private String cargo; // propiedad

    public Profesor () { // constructor
    }
    public String getCargo () { // asesor de cargo
        return cargo;
    }
    public void setCargo (String val) { // mutador de cargo
        this.cargo = val;
    }
    public String getMateria () { // asesor de materia
        return materia;
    }
    public void setMateria (String val) { // mutador de materia
        this.materia = val;
    }
}
```

Asociaciones varias


Fecha
Attributes
private int dia
private int mes
private int anio
Operations
public Fecha( )
public int getDia( )
public void setDia(int val)
public int getMes( )
public void setMes(int val)
public int getAnio( )
public void setAnio(int val)
1
1
Comprobante
Attributes
private char tipo
private int numero
private Fecha fecha
Operations
public Comprobante( )
public char getTipo( )
public void setTipo(char val)
public int getNumero( )
public void setNumero(int val)
public Fecha getFecha( )
public void setFecha(Fecha val)
Producto
Attributes
private int codigo
private String descripcion
private float precio
Operations
public Producto( )
public int getCodigo( )
public void setCodigo(int val)
public String getDescripcion( )
public void setDescripcion(String val)
public float getPrecio( )
public void setPrecio(float val)
1..*
mProducto
1..*
Cliente
Attributes
private int codigo
private String razonSocial
Operations
public Cliente( )
public int getCodigo( )
public void setCodigo(int val)
public String getRazonSocial( )
public void setRazonSocial(String val)
mCliente
1
1
Factura
Attributes
private float total
Operations
public Factura( )
public Cliente getCliente( )
public void setCliente(Cliente val)
public float getTotal( )
public void setTotal(float val)
public Producto[1..*] getProducto( )
public void setProducto(Producto val[1..*])

```
public class Fecha {

    private int dia;
    private int mes;
    private int anio;

    public Fecha () {
    }
    public int getAnio () {
        return anio;
    }
    public void setAnio (int val) {
        this.anio = val;
    }
    public int getDia () {
        return dia;
    }
    public void setDia (int val) {
        this.dia = val;
    }
    public int getMes () {
        return mes;
    }
    public void setMes (int val) {
        this.mes = val;
    }
}
```

```
public class Cliente {

    private int codigo;
    private String razonSocial;

    public Cliente () {
    }
    public int getCodigo () {
        return codigo;
    }
    public void setCodigo (int val) {
        this.codigo = val;
    }
    public String getRazonSocial () {
        return razonSocial;
    }
    public void setRazonSocial (String val) {
        this.razonSocial = val;
    }
}
```

```
public class Comprobante {

    private char tipo;
    private int numero;
    private Fecha fecha;

    public Comprobante () {
    }
    public Fecha getFecha () {
        return fecha;
    }
    public void setFecha (Fecha val) {
        this.fecha = val;
    }
    public int getNumero () {
        return numero;
    }
    public void setNumero (int val) {
        this.numero = val;
    }
    public char getTipo () {
        return tipo;
    }
    public void setTipo (char val) {
        this.tipo = val;
    }
}
```

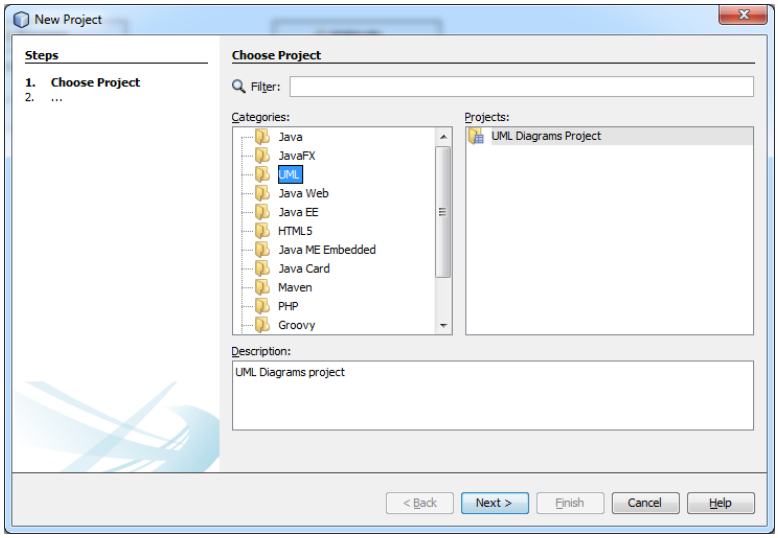
```
public class Producto {

    private int codigo;
    private String descripcion;
    private float precio;

    public Producto () {
    }
    public int getCodigo () {
        return codigo;
    }
    public void setCodigo (int val) {
        this.codigo = val;
    }
    public String getDescripcion () {
        return descripcion;
    }
    public void setDescripcion (String val) {
        this.descripcion = val;
    }
    public float getPrecio () {
        return precio;
    }
    public void setPrecio (float val) {
        this.precio = val;
    }
}
```

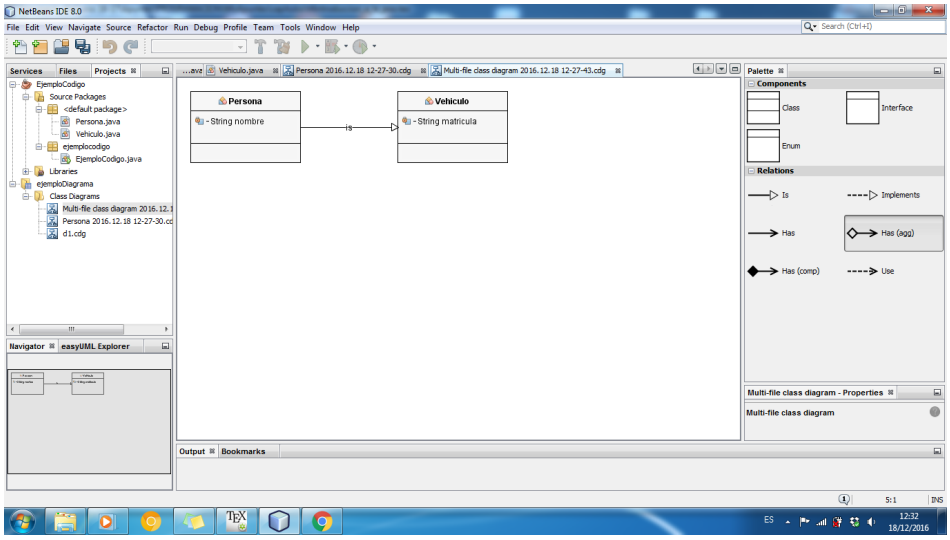
```
import java.util.ArrayList;

public class Factura extends Comprobante {

    private ArrayList<Producto> mProducto;
    private float total;
    private Cliente mCliente;

    public Factura () {
    }
    public Cliente getCliente () {
        return mCliente;
    }
    public void setCliente (Cliente val) {
        this.mCliente = val;
    }
    public float getTotal () {
        return total;
    }
    public void setTotal (float val) {
        this.total = val;
    }
    public ArrayList<Producto> getProducto () {
        return mProducto;
    }
    public void setProducto (ArrayList<Producto> val) {
        this.mProducto = val;
    }
}
```

6.4. UML Java: easyUML plugins para Netbeans

En Netbeans podemos instalar plugins para generar automática código a partir de un diagrama de clases o un diagrama a partir del código.

http://www.jc-mouse.net/ingenieria-de-sistemas/uml-java-easyuml-plugins-para-n




NetBeans IDE 8.0
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Persona
- String nombre
is
Vehiculo
- String matricula
Palette
Components
Class
Interface
Enum
Relations
Is
Implements
Has
Has (agg)
Has (comp)
Use
Multi-file class diagram - Properties
Multi-file class diagram
Output
Bookmarks
Navigator
easyUML Explorer