Clase 9

PROGRAMACIÓN 1

Objetivos del tema

- **Presentar el concepto de objeto, clase, atributo, método e instancia**
- **Interpretar el código fuente de una aplicación Java donde aparecen implementados los conceptos anteriores**
- **Construir una aplicación Java sencilla, convenientemente especificada, que emplee los conceptos anteriores.**

Programación Orientada a Objetos

En la vida real todos los objetos tienen una serie de características y un comportamiento.

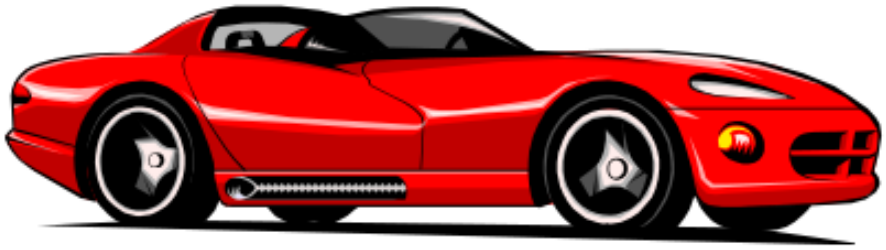
Por ejemplo, una puerta tiene color, forma, dimensiones, material... (características) y puede abrirse, cerrarse... (comportamiento)

En Programación Orientada a Objetos, un objeto es una combinación de datos y rutinas que operan con esos datos.

Objetos

Campos o atributos: almacenan datos. Estos datos pueden ser de tipo primitivo y/o otro tipo de objeto (composición de objetos).

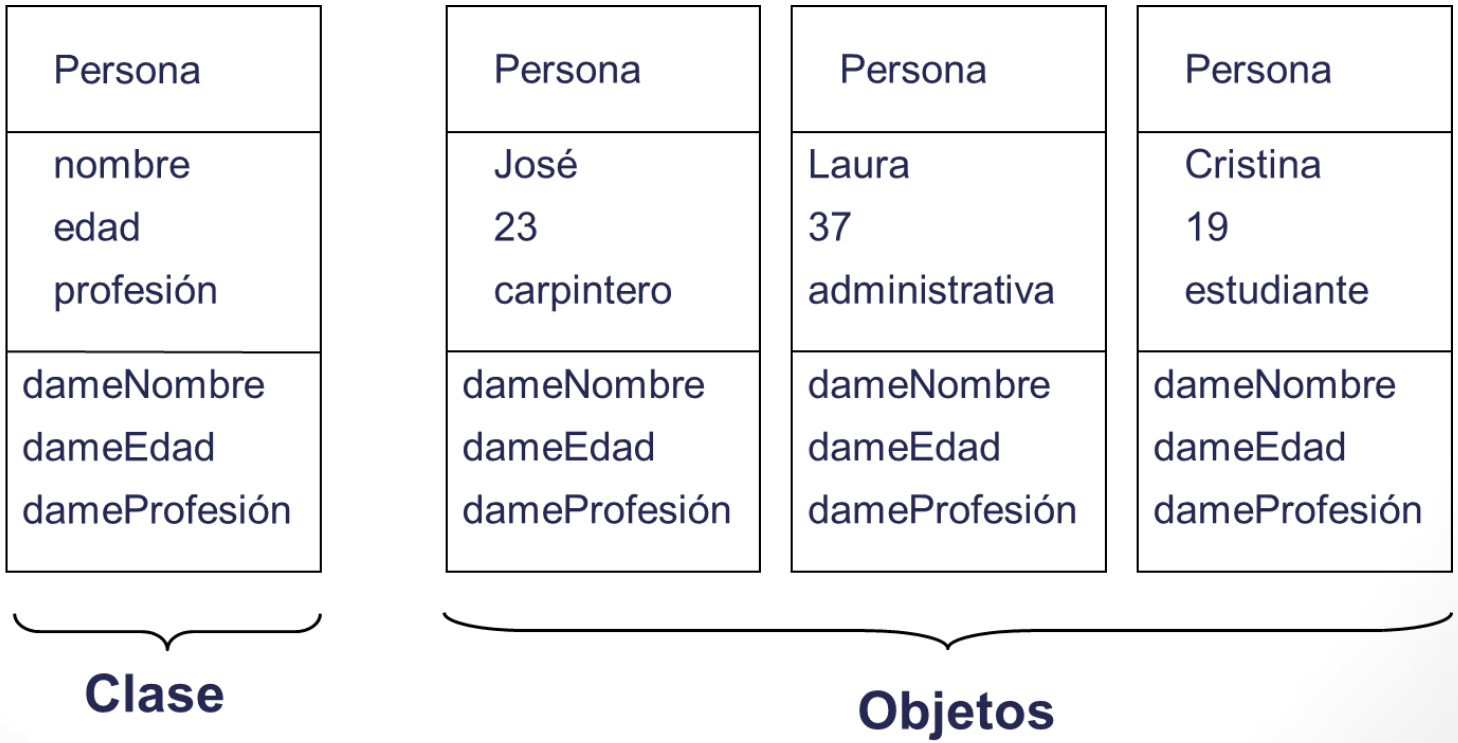
Rutinas o métodos: llevan a cabo una determinada acción o tarea con los atributos.



- Atributos:
  - color
  - velocidad
  - ruedas
  - motor
- Métodos:
  - arranca()
  - frena()
  - dobla()

¿Qué es una clase?

- Una clase representa al conjunto de objetos que comparten una estructura y un comportamiento.
- Una clase es una combinación específica de atributos y métodos y puede considerarse un tipo de dato de cualquier tipo no primitivo.
- Ejemplo: Representación mediante objetos de 3 personas.



Observaciones

- **En una aplicación se crean una serie de instancias (objetos) que interactúan entre sí mandándose mensajes (es decir, llamando a los métodos de los objetos).**

Clase principal{

- Definir constantes;
- Dentro de main()
  - Crear objetos
  - Procesar;

}

//Definir todas las clases involucradas, cada una en un archivo diferente

Clase *nombre*{

- Definir constantes;
- Definir atributos;
- Definir métodos;

}

Modificadores de acceso

Para modificar atributos o consultar su valor utilizamos métodos.

La base de la ***encapsulación*** consiste en ***hacer visible los atributos o métodos que sean realmente necesarios***.

Para controlar esto se utilizan los modificadores de acceso:

- **private** (Acceso solo dentro de la clase)
- **protected** (Acceso desde la clase y sus hijos "herencia")
- **public** (Acceso público desde cualquier lugar)

Estos modificadores de acceso se colocan justo delante del valor de retorno de un método, o del tipo de un atributo.

Para esta materia solo utilizaremos **private** y **public**.

Palabras clave static y final

static

Los atributos de una clase pueden ser de clase o de instancia. ***Son de clase si usa la palabra clave static***, y será única para todas las instancias (objetos) de la clase (***ocupa un único lugar en memoria***).

Si no usa static, el sistema crea un lugar nuevo para esa variable con cada instancia (***la variable es diferente para cada objeto***).

En el caso de una constante se usa **static**.

final

Es una variable de tipo constante y no admitirá cambios después de su declaración y asignación de valor.

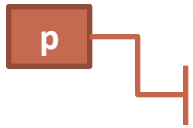
El método main como se viene usando es un método de clase, y va ser el único con esa característica.

Ejemplo

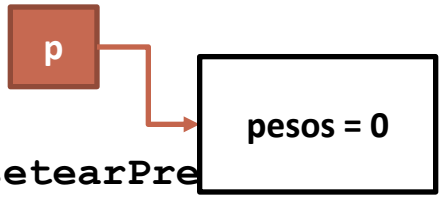
```
public class Programa {
 public static void main (String [] args ) {
 Precio p; // Crea una referencia de la clase Precio
 p = new Precio(); // Crea el objeto de la clase Precio
 p.setearPrecio(56.8); // Llamada al método que asigna 56.8 a pesos
 System.out.println("Valor = " + p.obtenerPrecio());
// Llamada al método que devuelve el valor de pesos
 Precio q = new Precio(); // Crea una referencia y el objeto
 q.pesos=75.6; // También asigna 75.6 al atributo pesos
 System.out.println("Valor = " + q.pesos);
 }
}

public class Precio {
// Atributo o variable miembro
 public double pesos;//Este atributo debería se private
 // Métodos de acceso a atributos
 public double obtenerPrecio() { return pesos; }
 public void setearPrecio(double x) { pesos=x; }
}
```

Explicación del ejemplo

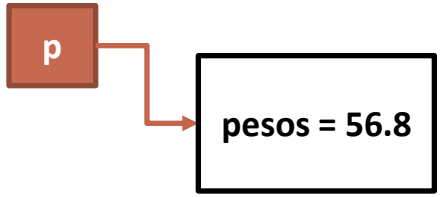
```
//Define una instancia de la clase Precio
Precio p;
```



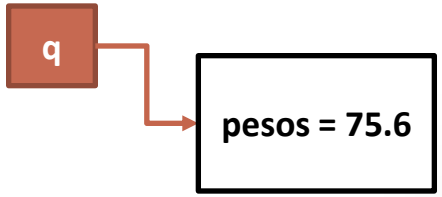

```
//Se asigna espacio a la instancia p de Precio
p = new Precio();
```



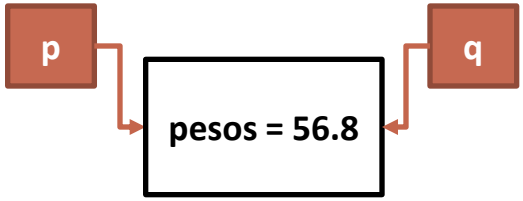

```
//Se asgina valor a pesos usando setearPre
p.setearPrecio(56.8);
```




```
Precio q = new Precio();
q.pesos=75.6;
```




```
q = p;
```



Constructor

- Los constructores son métodos pertenecientes a la clase. Se utilizan para construir o instanciar una clase. Puede haber ***varios constructores***, de acuerdo a las necesidades del usuario.

Estructura de una clase

Clase *nombre*{

- Definir constantes;
- Definir atributos;
- Definir constructores;
- Definir métodos;

}

Ejemplo

```
public class Programa {
 public static void main (String [] args ) {
 Precio p; // Crea una referencia de la clase Precio
 p = new Precio(0.0); // Crea el objeto de la clase Precio con pesos
  en 0
 p.setearPrecio(56.8); // Llamada al método que asigna 56.8 a pesos
 System.out.println("Valor = " + p.obtenerPrecio());
// Llamada al método que devuelve el valor de pesos
 Precio q = new Precio(75.6); // Crea el objeto y asigna 75.6 a pesos
 System.out.println("Valor = " + q.obtenerPrecio());
 }
}
public class Precio {
// Atributo o variable miembro
 private double pesos;//solo se accede por métodos
//Constructor de precio
 Precio(double x) { pesos=x; }
// Métodos de acceso a atributos
 public double obtenerPrecio() { return pesos; }
 public void setearPrecio(double x) { pesos=x; }
}
```

Ejercicios

Definir una clase Punto representada por sus coordenadas x e y. Agregar métodos de acceso.

```
public class Punto {
  private float x, y;
  Punto(float _x,float _y){x = _x; y = _y;}
  public void setearXY(float _x,float _y){x = _x; y = _y;}
  public float obtenerX(){ return x;}
  public float obtenerY(){ return y;}
}
```

Agregar un método que devuelva el cuadrante del punto.

```
public int cuadrantePunto(){
 if((x>=0)&&(y>=0)) return 1;
 else if((x<0)&&(y>=0)) return 2;
 else if((x<0)&&(y<0)) return 3;
 else return 4;
}
```

Ejercicios

- Crear un objeto nuevo punto en una clase Principal (que contiene el método main) y mostrando por pantalla sus coordenadas. Modificar alguna de sus coordenadas accediendo a través de sus métodos y volver a mostrarlo por pantalla.

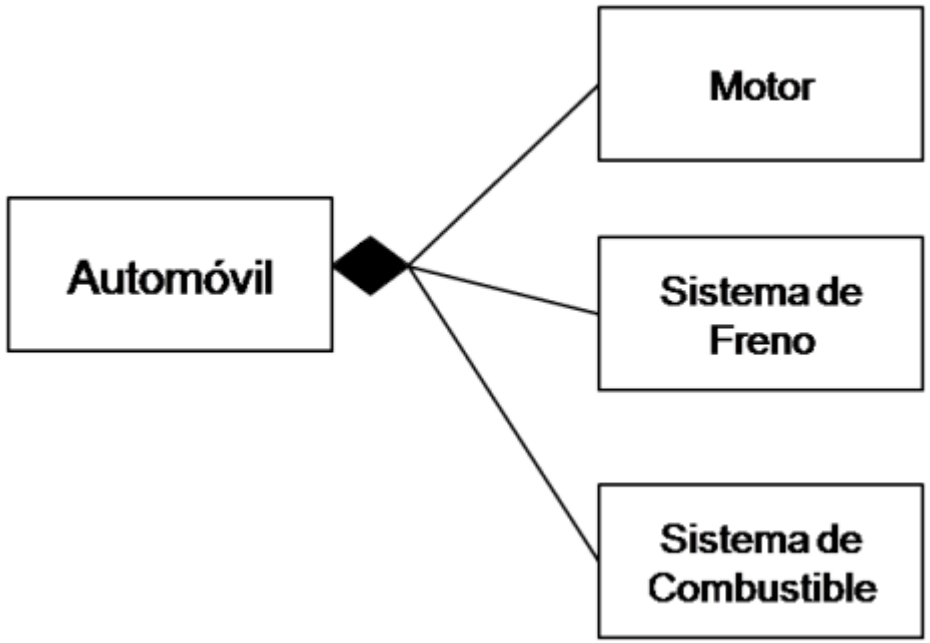
- Agregar a la clase punto un método que calcule su distancia al origen de coordenadas (**Math.sqrt(a)**, **Math.pow(a,b)**).

- Agregar un método a la clase punto que modifique el estado del punto, trasladándolo al primer cuadrante (si el punto ya está en el primer cuadrante su estado no cambia).

Composición

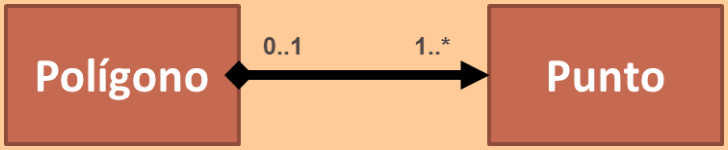
Mecanismo para crear clases complejas agregando objetos de clases ya existentes

Ejemplo: la nueva clase (Coche) tiene un objeto miembro (un atributo) de la clase Motor



Ejemplo

Se tiene un polígono formado por puntos bidimensionales, hasta un máximo de 10 puntos. Modelar las clases del problema con todos los métodos necesarios para la carga y acceso de los datos.




```
public class Punto {
  private float x, y;
  Punto(float _x,float _y){
       x = _x; y = _y;
  }
  public void setearXY(float _x,float _y){
       x = _x; y = _y;
  }
  public float obtenerX(){ return x;}
  public float obtenerY(){ return y;}
}
```

Ejemplo

```
public class Poligono {
  final static int MAXPUNTOS = 10;
  private Punto[] puntos = new Punto[MAXPUNTOS];
  private int cantPuntos = 0;
  public void agregarPunto(Punto xy){
       if (cantPuntos<MAXPUNTOS-1){
              puntos[cantPuntos] = xy;
              cantPuntos++;
       }
  }

  public Punto obtenerPunto(int pos){ return
  puntos[pos];}//accede a un punto en una pos
  public void imprimirPuntos(){
       for (int i = 0;i<cantPuntos;i++)
              System.out.println("Punto "+i+" =
       ("+obtenerPunto(i).obtenerX()+","+obtenerPunto(i).obtenerY
  ()+")");
  }
}
```

Ejemplo

Para probar el sistema desarrollado se realiza un programa donde se simula la creación de 3 puntos, que luego se agregan al polígono. Finalmente se imprime el contenido del polígono.

```
public class Programa {
  public static void main(String[] args) {
        Punto p1 = new Punto(1.0f, 1.0f);
        Punto p2 = new Punto(2.0f, 1.0f);
        Punto p3 = new Punto(1.0f, 3.0f);
        Poligono Pol = new Poligono();
        Pol.agregarPunto(p1);
        Pol.agregarPunto(p2);
        Pol.agregarPunto(p3);
        Pol.imprimirPuntos();
  }
}
```

SALIDA

```
Punto 0 = (1.0,1.0)
Punto 1 = (2.0,1.0)
Punto 2 = (1.0,3.0)
```

Ejercicios

- Se quiere crear una clase Cuenta la cual se caracteriza por tener asociado un número de cuenta y un saldo disponible. Además, se puede consultar el saldo disponible en cualquier momento, recibir ingresos y retirar monto.
- Crear una clase Persona, que se caracteriza por un DNI y una cuenta bancaria. Agregar un método donde crea conveniente que devuelva si la persona es morosa o tiene saldo negativo.
- Crear la clase Libro, cuyos atributos son el titulo, el autor, el número de páginas y la calificación que le damos entre 0 y 10. Crear los métodos típicos para poder modificar y obtener valores de atributos.
- Crear una clase ConjuntoLibros, que almacena un conjunto de libros (con un array de un tamaño fijo). Se pueden añadir libros que no existan (siempre que haya espacio), eliminar libros por título o autor, y mostrar por pantalla los libros con la mayor y menor calificación dada. (**NOTA:** para comparar dos variables String a y b, no se debe utilizar “a == b” sino “a.equals(b)” que devuelve true si son iguales).

Ejercicios

- Se desean modelar los contactos de un celular. Para cada contacto se guarda su nombre y apellido, fecha de nacimiento, número de teléfono, dirección y dirección de mail. El celular puede mostrar los contactos con su apellido y nombre, su edad y su número de teléfono. También muestra la ciudad a la que pertenece el contacto. El celular muestra información a modo de resumen donde se lista la totalidad de contactos y el promedio de edad de los contactos.
- Una serie está formada por un conjunto de episodios. Cada episodio posee un título, una descripción, un indicación de si el usuario ya vio el episodio y una calificación (con valores de 1 a 10). Las series, además, tienen un título, una descripción, un creador y un género.

  Se debe poder:

  - Ingresar la calificación de un episodio. Si el valor ingresado no es correcto imprimir un mensaje por pantalla y no cambiar el valor anterior.
  - Obtener el total de episodios vistos de la serie.
  - Obtener el promedio de las calificaciones dadas por el usuario.
  - Determinar si el usuario ya vio todos los episodios de la serie.

Ejercicios

- Implementar una clase “SimularBanco” que permita generar objetos o instancias de cuenta y persona, y simular un ejemplo.
- Implementar un simulador de “ConjuntoLibros”. Utilizando solo los métodos definidos para las clases involucradas se pide:
  - Definir una instancia de conjuntolibros.
  - Definir 4 instancias de libros.
  - Añadir dichas instancias al conjunto.
  - Eliminar del conjunto un libro según su autor.
  - Mostrar todos los datos de los libros del conjunto.
  - Eliminar del conjunto un libro según su titulo.
  - Mostrar todos los datos de los libros del conjunto.
- Implementar un simulador de una serie donde se muestre que el sistema funciona correctamente para todos los métodos implementados. Finalmente considere el ingreso de un número incorrecto de episodio junto con una calificación, y luego visualice todos los datos de la serie y sus episodios.

Simulador de banco

```
import java.io.BufferedReader;
import java.io.InputStreamReader;

public class SimularBanco {
public static void main(String[] args) {
String opcion;
BufferedReader entrada = new BufferedReader(new InputStreamReader(System.in));
try{
                System.out.println ("Ingrese string para iniciar primera parte: ");
                opcion = entrada.readLine();
                /*
                Crear persona S1 con dni 22;
                Crear persona S2 con dni 33;
                Crear cuenta C1 con número 12;
                Crear cuenta C2 con número 24;
                Asignar la cuenta C1 a la persona S1;
                Asignar la cuenta C2 a la persona S2;
                Realizar un ingreso de 50 en la cuenta S1;
                Realizar un ingreso de 10 en la cuenta S2;
                Lista estado datos propios de S1 junto con su estado de cuenta;
                Realizar un retiro de 20 en la cuenta S2;
                Lista estado datos propios de S1 junto con su estado de cuenta;
                Lista estado datos propios de S2 junto con su estado de cuenta;
                */
}
catch (Exception exc ) {
                System.out.println( exc );
}
}}
```