Fundamentos de Java

...

Programación Orientada a Objetos

Contenido

- Programación Orientada a Objetos
- Pilares
- Abstracción:
  - Introducción a las clases y objetos
  - Atributos
  - Métodos
  - Diagrama general de una clase
- Encapsulamiento:
  - Modificadores de acceso
  - Paquetes:
    - Import
    - Import estáticos
    - Paquetes más importantes en Java
- Herencia
- Polimorfismo

Programación Orientada a Objetos

Ventajas:

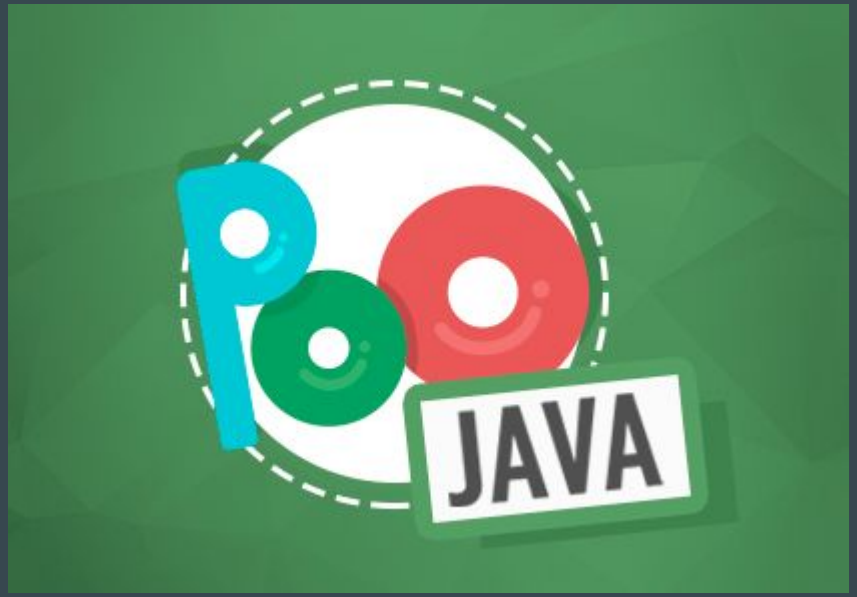
- Reusable
- Mantenible
- Facilidad de entendimiento
- Menos esfuerzo en pruebas de algoritmo
- Programas más sencillos y rápido
- Aumento de productividad

Desventajas:

- Limitaciones del programador
- No hay una forma única de resolver los problemas.
- Se requiere una documentación amplia para determinar la solución planteada.

Pilares de la POO

1. Abstracción
2. Encapsulamiento
3. Herencia
4. Polimorfismo
5. Modularidad
6. Principio de Ocultamiento
7. Recolección de Basura



Abstracción

Resaltar la parte más representativa de algo, ignorando detalles para centrarse en lo principal.

Introducción a las Clases y Objetos

Clases:

- Es una plantilla:
  - Nombre
  - Atributos
  - Métodos:
    - Arranque
    - Operaciones
  - Constructor
  - Destructor
- El nombre del archivo debe terminar con extensión **.java**
- Es una abstracción de algún hecho o ente del mundo real.

Objetos:

- Es una instancia de una clase.
- Se compone de dos elementos:
  - Atributos
  - Métodos

Atributos

Características:

- Son variables
- Guardan datos
- Deben ser declaradas, inicializadas y usadas tal como una variable o constante

Diferencias con variables:

- Conceptual
- Establece todas las características del objeto
- Define los parámetros para que los desarrolladores identifiquen el objetivo de la clase

Métodos

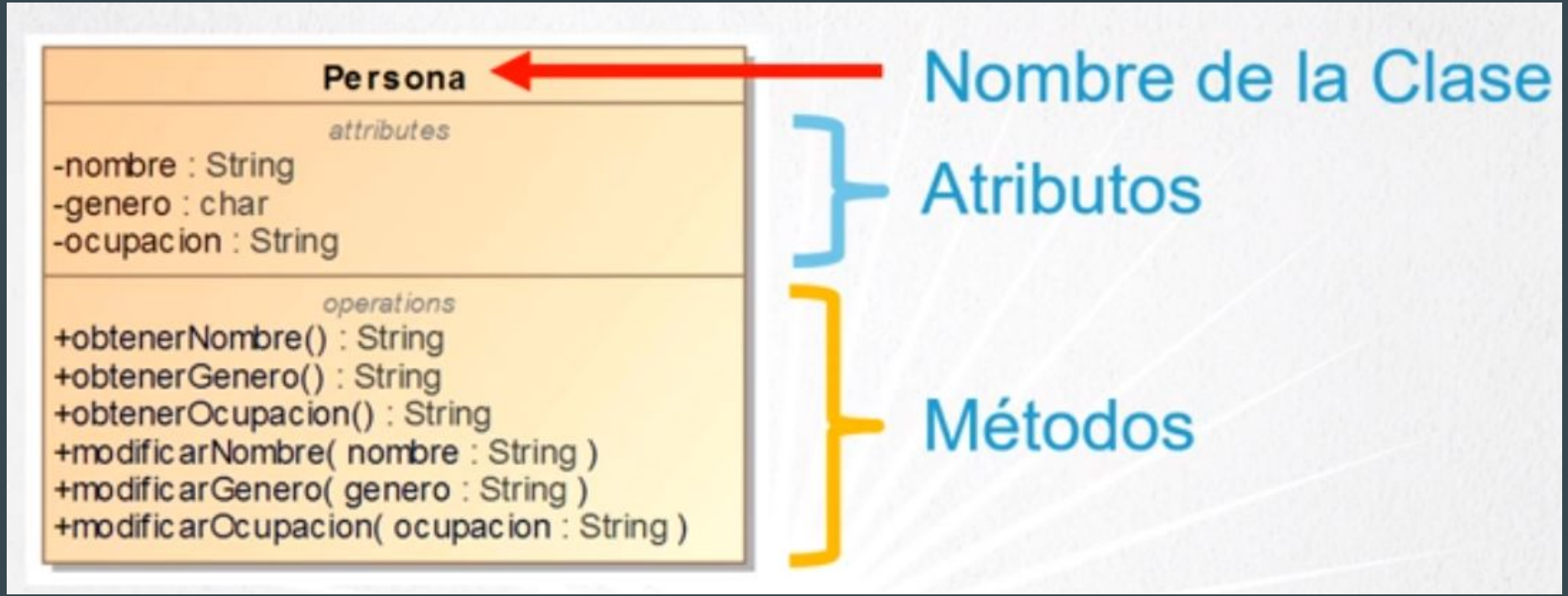
Características:

- Son funciones
- Guardan subrutinas o algoritmos que cumplirán un objetivo específico
- Deben ser declarados y se pueden usar desde cualquier ámbito siempre y cuando la clase se incluya en el mismo

Diferencias con funciones:

- Meramente conceptual
- Los métodos establecen el comportamiento que un objeto debe tener
- Los métodos deben definir los parámetros para que los desarrolladores puedan identificar con exactitud el objetivo o fin de esa clase java

Diagrama General de una Clase



Ejemplo

```java
public class UniversityStudent {
    int id;
    String name;
    String gender;
    String university;
    String career;
    int numSubjects;
    public UniversityStudent(int id, String name, String gender,
        String university, String  career, int numSubjects) {
        this.id = id;
        this.name = name;
        this.gender = gender;
        this.university = university;
        this.career = career;
        this.numSubjects = numSubjects;
    }
    void inscribeSubjects() {
      // TODO: implement
    }
    void cancelSubjects() {
       // TODO: implement
    }
    void consultRatings() {
       // TODO: implement
    }
}
```

Encapsulamiento

Es un mecanismo que consiste en organizar datos y métodos de una estructura, conciliando el modo en que el objeto se implementa, es decir, evitando el acceso a datos por cualquier otro medio distinto a los especificados.

Modificadores de Acceso

Modificador	Descripción
public	Los elementos pueden ser accedidos desde cualquier clase o instancia sin importar el paquete o procedencia de ésta.
private	Los elementos pueden ser accedidos únicamente por la misma clase.
protected	Permite acceso a los elementos desde la misma clase, clases del mismo paquete y clases que hereden de ella (incluso en diferentes paquetes).
default	Permite que tanto la propia clase como las clases del mismo paquete accedan a dichos componentes.

Setters / Getters

```java
public class PruebaEncapsulamiento {

    public static void main(String[] args) {
        //Creamos el objeto
        Persona p1 = new Persona();
        //Modificamos el atributo nombre
        p1.setNombre("Juan");
        //Accedemos al atributo nombre
        System.out.println("Nombre:" + p1.getNombre());
    }
}
```

```java
class Persona {
    //Atributo privado
    private String nombre;
    //Método publico para acceder al atributo nombre
    public String getNombre(){
        return nombre;
    }
    //Método publico para modificar al atributo nombre
    public void setNombre(String nombre){
        this.nombre = nombre;
    }
}
```

Ejemplo

Paquetes

- Organización de clases
- Evitan conflictos entre nombres de clases
- Limitan acceso a sus clases
- Utiliza la palabra reservada **package** seguido del nombre del paquete:
  - Usar minúsculas
  - Usar nombres cortos para uno mejor identificación
- Estructura:
  - paquete.clase / paquete.subpaquete.clase
  - paquete.clase.metodo / paquete.subpaquete.clase.metodo

Import

- Se utiliza la palabra reservada **import** para utilizar la funcionalidad de un paquete externo
- **import** debe ser utilizada sobre la declaración de la clase
- No utiliza espacio en memoria
- Usos:
  - Específico
  - Generico (*)
- Import estático:
  - import static paquete.clase.metodo

Import Estático

```java
package com.gm; //Definición del paquete

public class Utileria {
    public static void imprimir(String s){
        System.out.println("Imprimiento mensaje: " + s);
    }
}
```

```java
//Importamos el método estático a utilizar
import static com.gm.Utileria.imprimir;

public class EjemploPaquetes {

    public static void main(String[] args) {
        imprimir("Hola");
    }
}
```

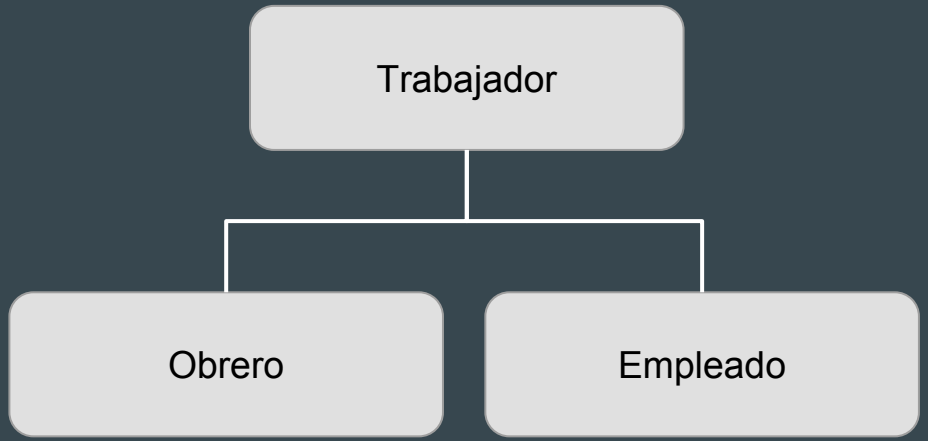
Paquetes más importantes en Java

Paquete	Contenido
java.lang	Contiene clases esenciales, se importa implícitamente sin necesidad de un sentencia import
java.util	Contiene las clases de utilería más comunes (Scanner).
java.io	Clases que definen distintos flujos de datos (input/output).
java.net	Se usa en combinación con las clases del paquete java.io para leer y/o escribir datos en la red.
java.applet	Contiene las clases necesarias para crear applets y se ejecutan en la ventana del navegador.
java.awt	Contiene clases para crear una aplicación GUI (Graphic User Interface) independiente de la plataforma.

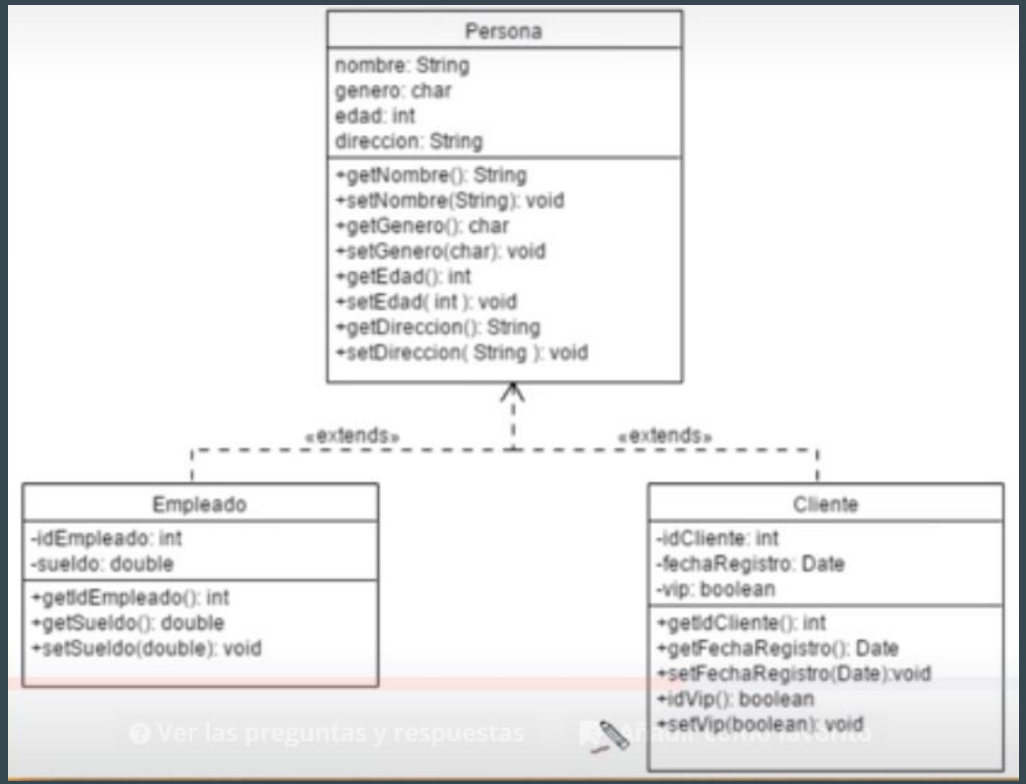
Ejemplo

Herencia

- Utilizar la estructura de una clase padre
- La clase hija se puede acceder a los métodos y atributos de la clase padre
- Se usa la palabra reservada **extends**, seguido del nombre de la clase de la cual se quiere heredar
- Representar comportamiento común
- Evitar duplicación de código
- Se utiliza la palabra **super** para acceder a los componentes del padre

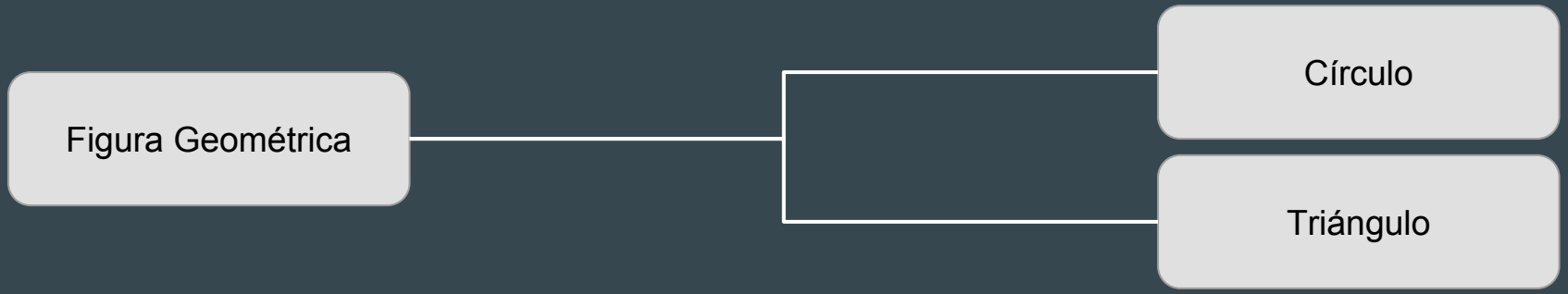


Ejemplo



Clase Persona
+ getNombre()
+ setNombre()
+ getGenero(), etc

Clase Empleado
+getIdEmpleado()
+getSueldo(), etc



Polimorfismo

Es la capacidad que tienen los objetos de comportarse de múltiples formas.

Ejercicio: Aplicando POO

Trabajemos en conjunto:

- Nombre
- Atributos
- Métodos:
  - Arranque
  - Operaciones
- Constructor
- Destructor