	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: José Antonio Ayala Barbosa

Asignatura: programación orientada a objetos

Grupo: Gpo 1 22-2

No de Práctica(s): 05

Integrante(s): Rosillo Montijo Emmanuel Alonso

No. de Equipo de cómputo empleado:

No. de Lista o Brigada:

Semestre: 03

Fecha de entrega: 17/02/2022

Observaciones:

CALIFICACIÓN:

_____

Objetivo

Implementar los conceptos de herencia en un lenguaje de programación orientado a objetos.

Introducción

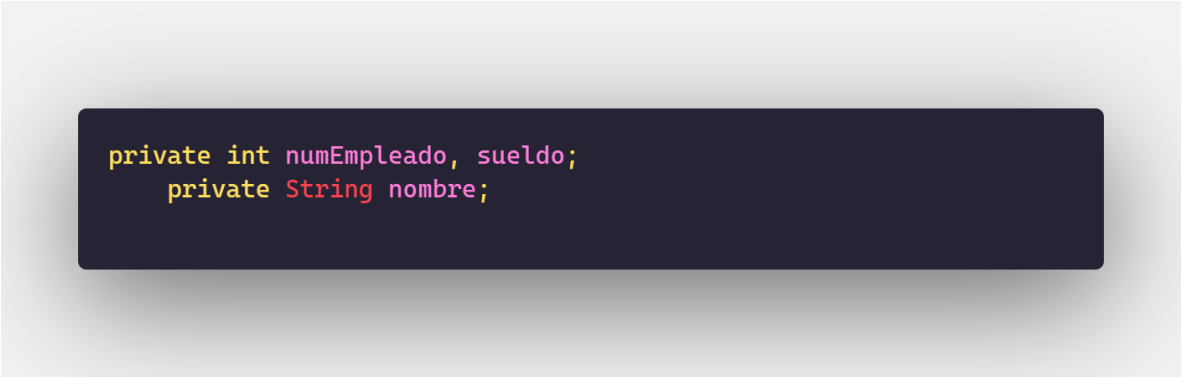
Una definición de herencia: “La herencia es un mecanismo que permite la definición de una clase a partir de la definición de otra ya existente. La herencia permite compartir automáticamente métodos y datos entre clases, subclases y objetos”, para contextualizar, la herencia son los atributos y métodos de otras clases que pueden ser usados en las clases hijas.

Desarrollo

Al empezar a abordar la práctica, podemos partir las secciones de su elaboración en las siguientes secciones.

Primero se creó una clase empleado

Con los atributos:



Contine los clásicos getters y setters de las clases.

Y un método que se implementó:

```
public void aumnetarSueldo(int porcentaje){
        sueldo += (int) (sueldo * porcentaje / 100);
    }
```

Y también contiene el clásico toString.

Luego la clase gerente que hereda de empleados. Esto se puede observar en el método constructor lleno:

```
ublic Gerente(int presupuesto, int numEmpleado, int
sueldo, String nombre) {
        super(numEmpleado, sueldo, nombre);
        this.presupuesto = presupuesto;
    }
```

contiene un atributo heredado indicado por “super”, se encarga de recibir los parámetros de un objeto empleado y utilizarlo como herencia dentro de la clase gerente, gerente contiene de la misma forma los clásicos setters, getters y toString

la clase gerente puede utilizar los atributos de empleado, en el menú principal se puede observar el ejemplo

```
gerente.aumnetarSueldo(100);
     gerente.aumentarSueldo(50_000.5);
```

Con el método que utiliza los atributos de herencia anterior, logra modificar los atributos de la clase empleado.

Y devuelve como resultado compuesto de la clase gerente y empleado

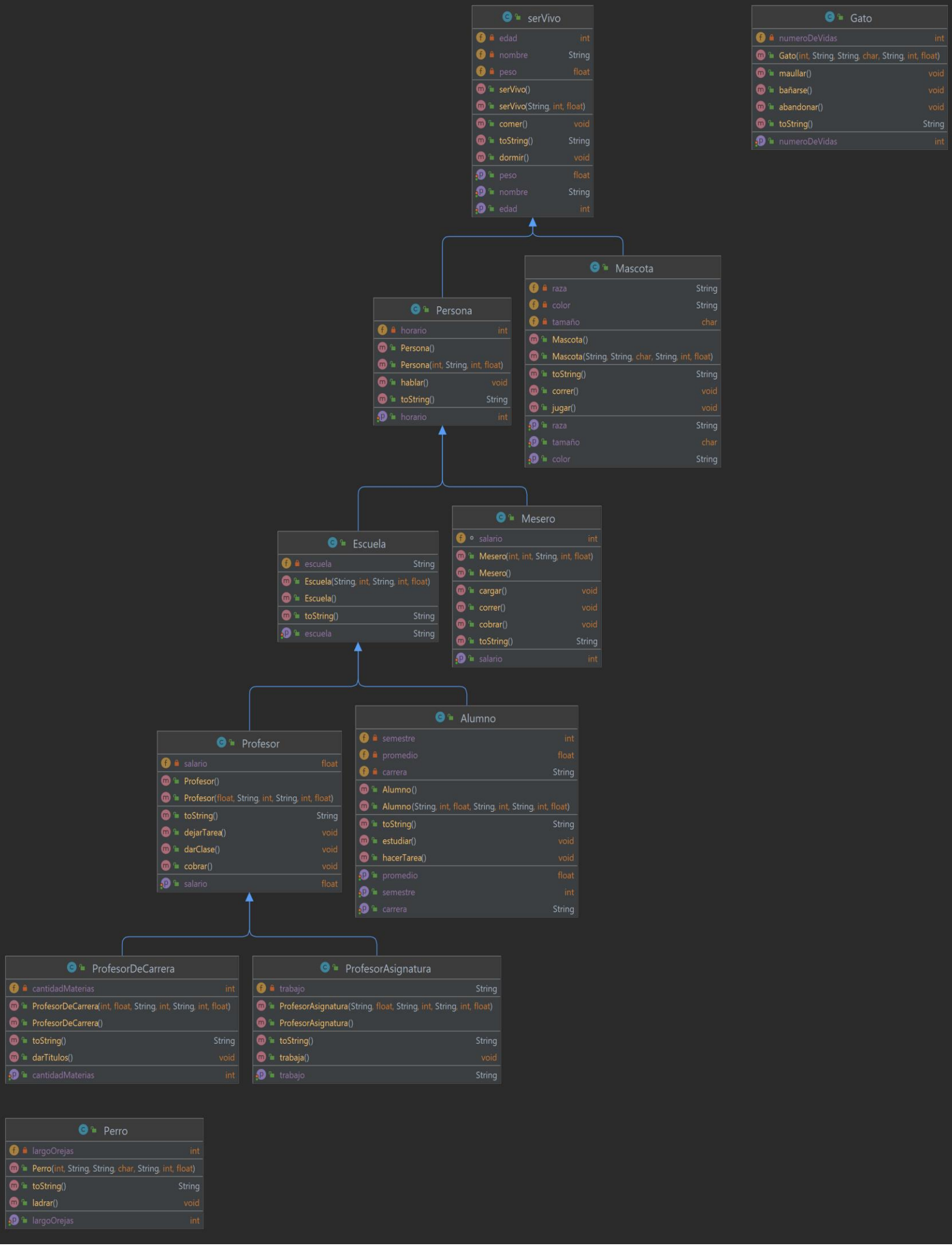
```
Empleado{numEmpleado=11, sueldo=50000, nombre=Frida}Gerente{presupuesto=1000000}
Instancia de gerente
Intancia de empleado
Instancia de object
```

Las instancias se pueden observar con método *instanceOf*

```
if (gerente instanceof Gerente){
            System.out.println("Instancia de gerente");
        }if(gerente instanceof Empleado){
            System.out.println("Intancia de empleado");
        }if (gerente instanceof Object){
            System.out.println("Instancia de object");
        }
```

Para poder observar a que instancia pertenece cada uno.

Luego como actividades extra se generaron 11 clases, cada una generarda con diferentes métodos y atributos, pero en común la herencia que posee cada una, para poder observar estás clases, al ser bastantes, se creó un **UML** para poder observarlas gráficamente y optimizar así el tiempo y el espacio de la práctica, de esta forma.


serVivo
edad int
nombre String
peso float
serVivo()
serVivo(String, int, float)
comer() void
toString() String
dormir() void
peso float
nombre String
edad int
Gato
numeroDeVidas int
Gato(int, String, String, char, String, int, float)
maullar() void
bañarse() void
abandonar() void
toString() String
numeroDeVidas int
Persona
horario int
Persona()
Persona(int, String, int, float)
hablar() void
toString() String
horario int
Mascota
raza String
color String
tamaño char
Mascota()
Mascota(String, String, char, String, int, float)
toString() String
correr() void
jugar() void
raza String
tamaño char
color String
Escuela
escuela String
Escuela(String, int, String, int, float)
Escuela()
toString() String
escuela String
Mesero
salario int
Mesero(int, int, String, int, float)
Mesero()
cargar() void
correr() void
cobrar() void
toString() String
salario int
Profesor
salario float
Profesor()
Profesor(float, String, int, String, int, float)
toString() String
dejarTarea() void
darClase() void
cobrar() void
salario float
Alumno
semestre int
promedio float
carrera String
Alumno()
Alumno(String, int, float, String, int, String, int, float)
toString() String
estudiar() void
hacerTarea() void
promedio float
semestre int
carrera String
ProfesorDeCarrera
cantidadMaterias int
ProfesorDeCarrera(int, float, String, int, String, int, float)
ProfesorDeCarrera()
toString() String
darTitulos() void
cantidadMaterias int
ProfesorAsignatura
trabajo String
ProfesorAsignatura(String, float, String, int, String, int, float)
ProfesorAsignatura()
toString() String
trabaja() void
trabajo String
Perro
largoOrejas int
Perro(int, String, String, char, String, int, float)
toString() String
ladrar() void
largoOrejas int

Podemos observar sus métodos y atributos de cada clase, cuales son sus respectivos padres, el tipo de encapsulamiento y sus respectivas instancias

Cada uno tiene un método en el constructor de herencia, como ejemplo:

```
public ProfesorDeCarrera(int cantidadMaterias, float
salario, String escuela, int horario, String nombre,
int edad, float peso) {
        super(salario, escuela, horario, nombre, edad,
 peso);
        this.cantidadMaterias = cantidadMaterias;
    }
```

```
ublic ProfesorAsignatura(String trabajo, float salario, String escuela, int
 horario, String nombre, int edad, float peso) {
        super(salario, escuela, horario, nombre, edad, peso);
        this.trabajo = trabajo;
    }
```

Y su instancia:

```
public Profesor(float salario, String escuela, int horario, String nombre, int edad,
float peso) {
        super(escuela, horario, nombre, edad, peso);
        this.salario = salario;
    }
```

Conclusiones:

La herencia en la programación orientada objetos, facilita la manera de realizar un código más complejo, debido a que no existe la necesidad de copiar clases para ser utilizadas en otras y de esta forma, tener un programa más entendible y corto.

Referencias:

Alex Walton. (agosto 12, 2020).

Herencia en Java: Tipos y Ejemplos.

Hoy, de Java desde cero Sitio web: https://javadesdecero.es/poo/herencia-java-tipos-ejemplos/