Laboratorio I

Principios y practicas de desarrollo de software orientado a objetos

Docente:

Dario Alejandro Riaño Velandia

Estudiantes:

Isabella Callejas Mandon - 2202030
Geiner Duvan Guevara Vargas - 2201840

Grupo: A1
Subgrupo:G5

Universidad Industrial de Santander
Facultad de Fisicomecánicas
Escuela de Ingeniería de Sistemas

Bucaramanga, Agosto del 2023

Introducción

La programación orientada a objetos se basa en el concepto de crear un modelo del problema de destino en sus programas. La programación orientada a objetos disminuye los errores y promociona la reutilización del código a partir de la construcción de clases que son abstracciones limitadas por la necesidad del objeto [IBM, 2017]. Por tanto, en esta práctica de laboratorio, se plantean dos clases: una de ellas se inspira en el modelo de residencias, en la cual se han incluido atributos y métodos relacionados con los huéspedes; la segunda clase, por otro lado, se refiere a una máquina impresora. las cuales fueron planteados en un diagrama UML para después implementarlo en lenguaje de programación java.

Índice

1. Ejercicios

1.1. Primer objeto

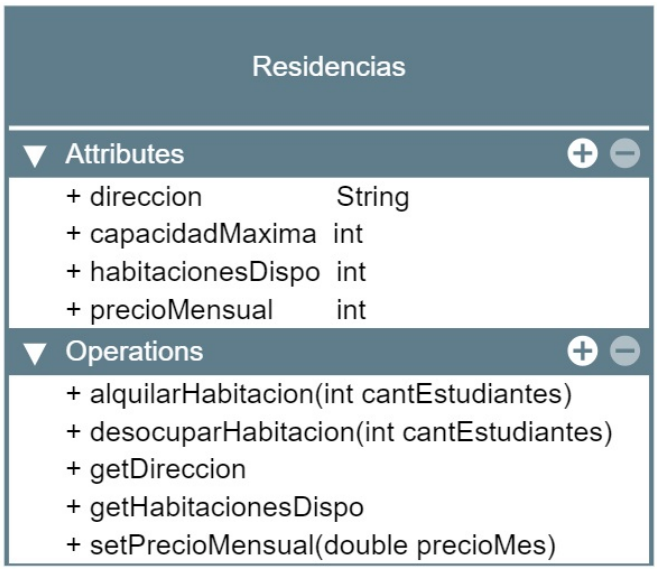

Figura 1: UML de la clase de Residencia

```java
public class Residencia {
    String direccion;
    int capacidadMax;
    int habitacionesDispo;
    double precioMensual;

    public Residencia() {
        direccion = "direccion1";
        capacidadMax = 20;
        habitacionesDispo = capacidadMax;
        precioMensual = 300.50;
    }
```

```
13
14      public void alquilarHabitacion(int cantEstudiantes) {
15          if (cantEstudiantes <= habitacionesDispo) {
16              habitacionesDispo -= cantEstudiantes;
17              System.out.println(cantEstudiantes + "
                    habitaciones alquiladas con exito.");
18          } else {
19              System.out.println("No hay suficientes
                    habitaciones disponibles para alquilar.");
20          }
21      }
22
23      public void desocuparHabitacion(int cantEstudiantes)
            {
24          if (capacidadMax - habitacionesDispo >=
                cantEstudiantes) {
25              habitacionesDispo += cantEstudiantes;
26              System.out.println(cantEstudiantes + "
                    habitaciones desocupadas con exito.");
27          } else {
28              System.out.println("No es posible desocupar
                    esta cantidad de habitaciones.");
29          }
30      }
31
32      public String getDireccion() {
33          return direccion;
34      }
35      public int getHabitacionesDispo() {
36          return habitacionesDispo;
37      }
38
39      public void setPrecioMensual(double precioMes) {
40          precioMensual = precioMes;
41      }
42  }
```

1.2. Segundo objeto

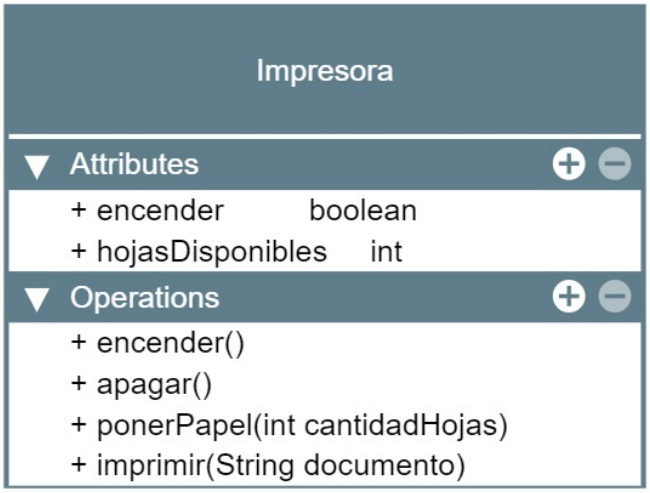

Figura 2: UML de la clase de Impresora

```
1  public class Impresora {
2      String modelo;
3      boolean encendida;
4      int hojasDisponibles;
5      String tipoConexion;
6
7      public Impresora() {
8          modelo = "LG";
9          tipoConexion = "Internet";
10         encendida = false;
11         hojasDisponibles = 0;
12     }
13
14     public void encender() {
15         if (encendida==false) {
16             encendida = true;
17             System.out.println("Impresora encendida.");
18         } else {
19             System.out.println("La impresora ya esta
                   encendida.");
20         }
21     }
22
23     public void apagar() {
24         if (encendida==true) {
```

```
            encendida = false;
            System.out.println("Impresora apagada.");
        } else {
            System.out.println("La impresora ya esta
                apagada.");
        }
    }

    public void ponerPapel(int cantidadHojas) {
        if (encendida==true) {
            hojasDisponibles += cantidadHojas;
            System.out.println("Se han puesto " +
                cantidadHojas + " hojas de papel.");
        } else {
            System.out.println("Enciende la impresora.");
        }
    }

    public void imprimir(String documento) {
        if (encendida==true && hojasDisponibles > 0) {
            System.out.println("Imprimiendo documento: "
                + documento);
            hojasDisponibles--;
        } else if (encendida==false) {
            System.out.println("Enciende la impresora
                para imprimir.");
        } else {
            System.out.println("No hay suficiente papel
                para imprimir.");
        }
    }

}
```

2. Conclusiones

En conclusión, la implementación de estas dos clases desarrolladas bajo el paradigma de programación orientado a objetos demuestra el poder que tiene en reutilización de código y como se aprovecha esto en la estructura del código.

Referencias

[IBM, 2017] IBM (2017). Programación orientada a objetos. https://www.ibm.com/docs/es/spss-modeler/saas?topic=language-object-oriented-programming. Accessed: 2023-08-30.