Instituto Politécnico Nacional
Unidad Profesional
Interdisciplinaria De Ingeniería
Campus Zacatecas

Ingeniería en Sistemas Computacionales

Práctica 8 – Clases Abstractas

Alumna: Vanessa Melenciano Llamas

Boleta: 2020670081

Profesora: Monreal Mendoza Sandra Mireya

Materia: Programación Orientada a Objetos

Grupo: 2CM2

Fecha de entrega: 11 de enero de 2021

Índice

Introducción 3
Objetivos 3
Desarrollo 4
Diseño de la solución 4
Funciones 5
Errores detectados 6
Posibles mejoras 6
Conclusión 6
Referencia 7

Introducción

Una clase abstracta es una clase de la cual no se pueden definir (o construir) instancias (u objetos).
Por tanto, las clases abstractas tendrán dos utilidades principales:

1. En primer lugar, evitan que los usuarios de la clase puedan crear objetos de la misma, como dice la definición de clase abstracta. De este modo, en nuestro ejemplo anterior, no se podrán crear instancias de la clase “Articulo". Este es un comportamiento deseado, ¿ya que si bien “Articulo” nos permite crear una Jerarquía sobre las clases “Tipo4” Tipo?" y "Tipo 16" un objeto de la clase "Articulo" como tal no va a aparecer en nuestro desarrollo (todos los artículos tendrán siempre un IVA asignado). Sin embargo, es importante notar que si se pueden declarar objetos de la clase Articulo que luego deberán se construidos como de las clases Tipo4”, ¿Tipo?" 6 Tipo 16").
2. En segundo lugar, permiten crear interfaces que luego deben ser implementados por las clases que hereden de la clase abstracta. Es evidente que una clase abstracta, al no poder ser instanciada, no tiene sentido hasta que una serie de clases que heredan de ella la implementan completamente y les dan un significado a todos sus métodos. A estas clases, que son las que hemos utilizado a lo largo de todo el curso, las podemos nombrar clases concretas para diferenciarlas de las clases abstractas.

De la propia definición de clase abstracta no se sigue que una clase abstracta deba contener algún método abstracto, aunque generalmente será así. En realidad, el hecho de definir una clase cualquiera como abstracta se puede entender como una forma de evitar que los usuarios finales de la misma puedan crear objetos de ella; es como una medida de protección que el programador de una clase pone sobre la misma.

Sin embargo, lo contrario si es cierto siempre: si una clase contiene un método abstracto, dicha clase debe ser declarada como abstracta. Si hemos declarado un método abstracto en una clase, no podremos construir objetos de dicha clase

Una clase que especifica un protocolo, pero es incapaz de implementarlo completamente porque sus subclases pueden tener representaciones diferentes se denomina clase abstracta. Las clases abstractas no pueden ser instanciadas, su finalidad es definir miembros comunes que heredan sus subclases, también llamadas clases concretas. Para declarar una clase abstracta se utiliza la palabra reservada abstracto.

Los métodos definidos en la clase abstracta pueden ser redefinidos (sobrescritos) en sus subclases. Esto es posible gracias al polimorfismo dinámico.

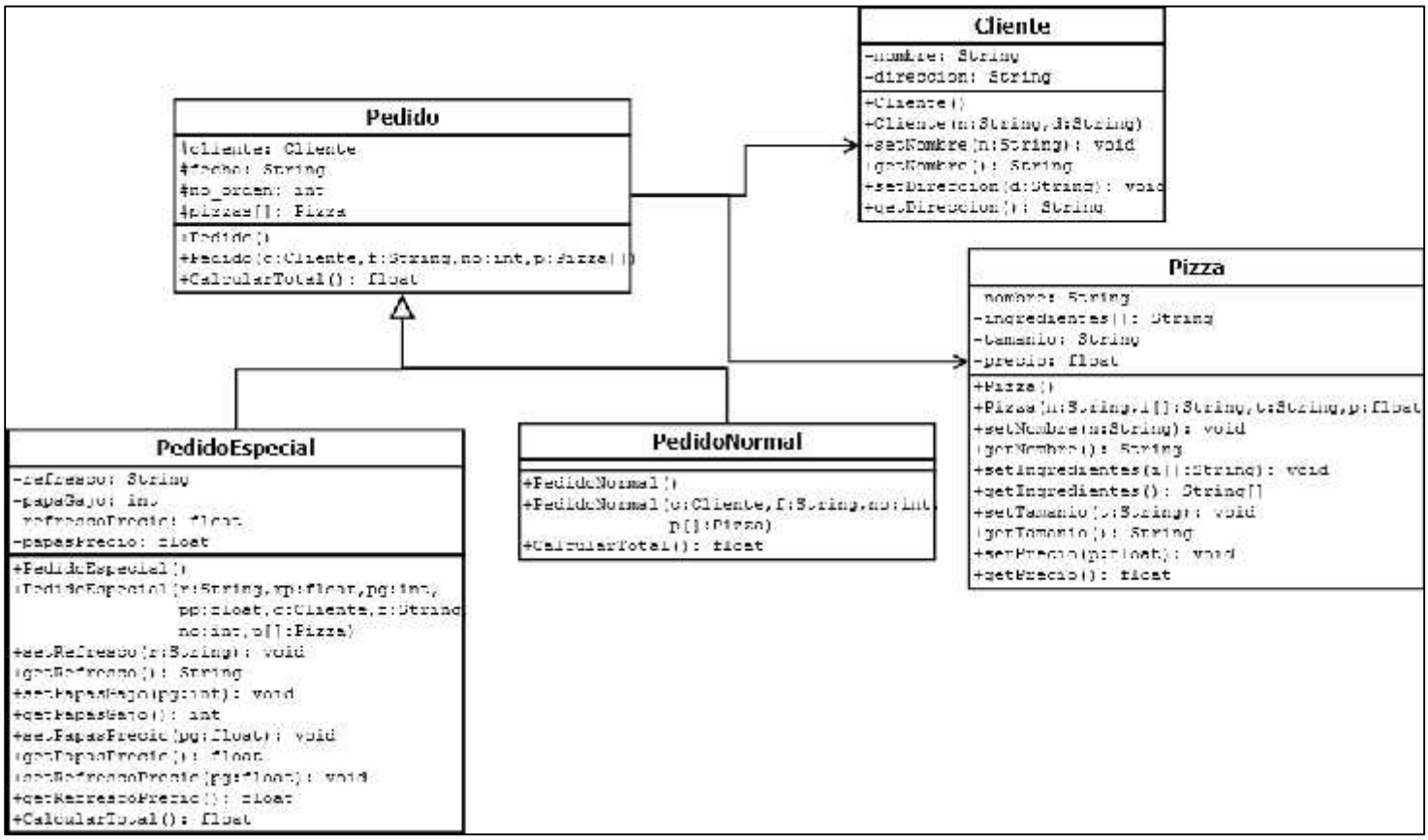
Objetivos

Comprender el funcionamiento principal de las clases abstractas, aplicando la herencia simple.

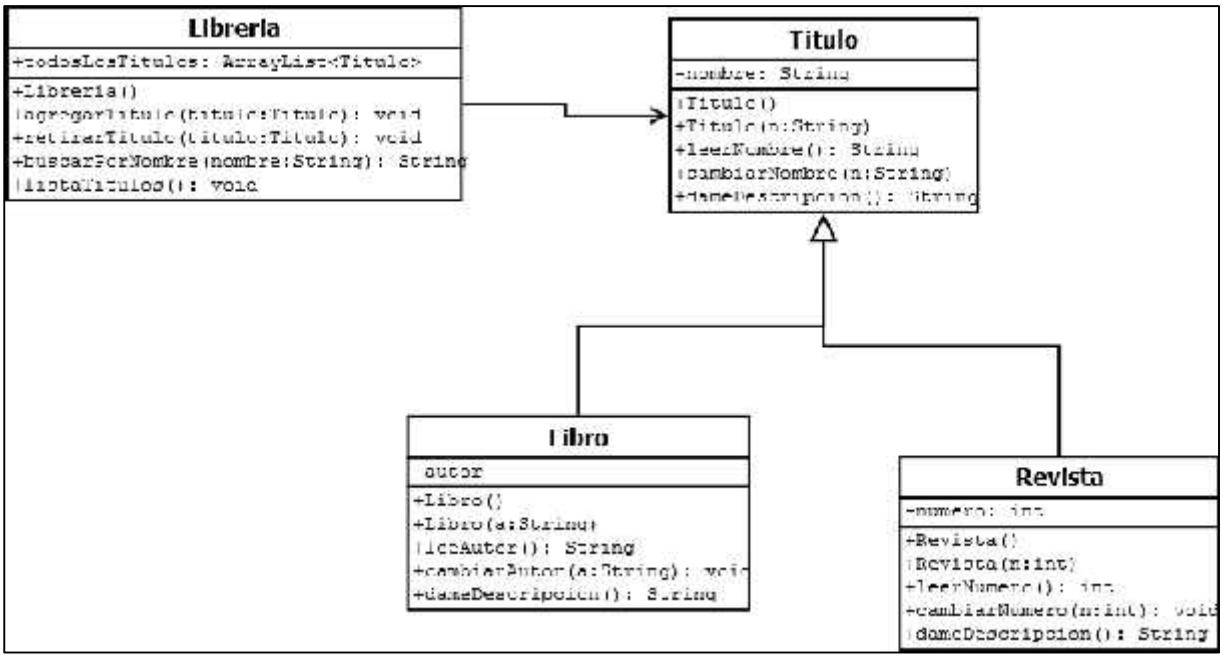
Desarrollo

Diseño de la solución

- Ejercicio 1



- Ejercicio 2



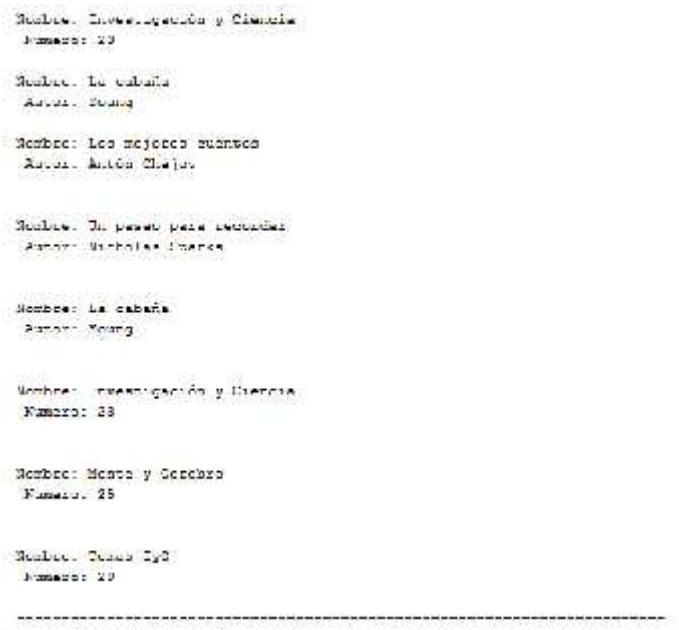
Funciones

- Ejercicio 1

Para comprobar el funcionamiento del código, se hizo una clase “main”:

```
public static void main(String args[]){
    Libreria MiLibreria = new Libreria();

    Libro MiLibro1 = new Libro( a: "Antón Chéjov", n: "Los mejores cuentos");
    Libro MiLibro2 = new Libro( a: "Nicholas Sparks", n: "Un paseo para recordar");
    Libro MiLibro3 = new Libro( a: "Young", n: "La cabaña");

    Revista MiRevista1 = new Revista( n: 23,  nom: "Investigación y Ciencia");
    Revista MiRevista2 = new Revista( n: 25,  nom: "Mente y Cerebro");
    Revista MiRevista3 = new Revista( n: 29,  nom: "Temas IyC");

    MiLibreria.agregarTitulo( titulo: MiLibro1);
    MiLibreria.agregarTitulo( titulo: MiLibro2);
    MiLibreria.agregarTitulo( titulo: MiLibro3);
    MiLibreria.agregarTitulo( titulo: MiRevista1);
    MiLibreria.agregarTitulo( titulo: MiRevista2);
    MiLibreria.agregarTitulo( titulo: MiRevista3);

    MiLibreria.listaTitulos();

    System.out.println( "\n" + MiLibreria.buscarPorNombre( nombre: "Investigación y Ciencia"));
    System.out.println( "\n" + MiLibreria.buscarPorNombre( nombre: "La cabaña"));

    MiLibreria.retirarTitulo( titulo: MiLibro2);
    MiLibreria.listaTitulos();
}
```

Dando como resultado:



- Ejercicio 2

Para este ejercicio, se metieron los datos correctos para hacer un pedido de las pizzas, mostrando el total de la cuenta:

Errores detectados

En esta práctica no hubo grandes dificultades ya que nos temas vistos, se habían usado con anterioridad, siendo solo una comprobación de la aplicación de todo.

Posibles mejoras

En este caso, lo que se puede mejorar es seguir practicando la utilización de la sobre escritura con las clases abstractas para entender mejor, y comprender los atributos protected.

Conclusión

En Java declaramos una clase abstracta con la palabra reservada abstract. También podemos hacer lo mismo con los métodos: si una clase tiene métodos abstractos, entonces nuestra clase deberá ser abstracta. Nuestro método abstracto será compartido por las clases que hereden de nuestra clase abstracta. Y son utilizadas para la sobre escritura de clases dentro de la herencia simple.

Al desarrollar la práctica, pudo ser más claro la implementación de la herencia simple utilizando las clases abstractas con sus respectivos métodos, manejando barias clases en los proyectos donde ya no es un solo tema el que se implementa, sino que se hace mención a temas ya vistos, reafirmando y comprendiendo estos.

Referencia

Ottogalli Fernández, K. A., Martínez Morales, A. A., & León Guzmán, L. (2011). NASPOO: una notación algorítmica estándar para Programación Orientada a Objetos. *Universidad Privada Dr. Rafael Belloso Chacín*, pp. 81-102.