	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorios de docencia

Laboratorio de Computación Salas A y B

Profesor(a): _____

Asignatura: _____

Grupo: _____

No de Práctica(s): _____

Integrante(s): 320095672

_____

_____

_____

No. de lista o brigada: _____

Semestre: _____

Fecha de entrega: _____

Observaciones: _____

_____

CALIFICACIÓN: _____

Índice

González Falcón Luis Adrián

Introducción

Ejercicio 1

Escribir un programa en Java para crear la interfaz Ordenamiento con el método ordenar. Después la interfaz se implementará en clases BubbleSort y SelectionSort.

Ejercicio 2

Escribir un programa en Java para crear la clase abstracta Figura con el método calcular área. Crear las clases Círculo y Cilindro que van a sobreescribir el método calcular área.

Práctica 81

Escribir un programa en Java para crear una interfaz Ordenamiento, y las clases QuickSort y MergeSort que hagan implementación de la interfaz.

Práctica 82

Escribir un programa en Java para crear la clase Empleado con los atributos nombre y rol. E implementar el método calcular salario. Adicionalmente crear las clases derivadas Gerente y Programador con sus propios atributos y que sobreescriban el método calcular salario.

La realización de la práctica será útil para tener la habilidad y el conocimiento de jerarquizar las funciones que algunas entidades deben de tener, así como también crear jerarquías entre las clases que existen. Una vez logrado lo anterior, se espera que esta habilidad otorgue mayor firmeza al momento de crear código.

Marco teórico

metodos abstractos
interfaces
polimorfismo @overide

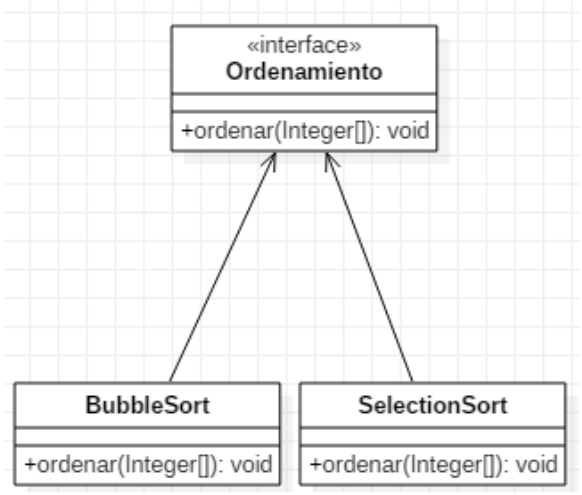
Para la solución de los problemas vistos se tuvo que usar un nuevo concepto visto apenas en clase, llamado **interfaces**. La funcionalidad de estas interfaces es definir y estandarizar las formas en que pueden interactuar los entes entre sí. **[1]**
Por ejemplo, se puede crear una interfaz 'automóvil' que contiene los métodos de "cambiar velocidad". Todos los vehículos deben de cambiar velocidades, pero al mismo tiempo, todos los vehículos cambian sus velocidades de manera diferente, este es la función de **interface**, crea el método que se va a usar para que los demás objetos (los objetos más específicos de automóvil) lo sobrescriban a su gusto.

A veces, se tiene la intención de únicamente heredar comportamiento y sus clases derivadas, es por esto que existe las clases abstractas. Este tipo de clases tienen como propósito crear varias clases derivadas, donde se hereden estos métodos creados en la clase abstracta, con el propósito de que se sobrescriban y así, tener un diseño común donde comparten el mismo método pero que tiene una funcionalidad diferente. Algo a resaltar es que en las funciones abstractas únicamente van a existir métodos abstractos, que funciona como indicador de que ese método existirá en la clase heredada, y por ende, tendrá que aplicar polimorfismo para sobrescribirla y así crear su comportamiento. **[2]**

Desarrollo

Ejercicio 0.

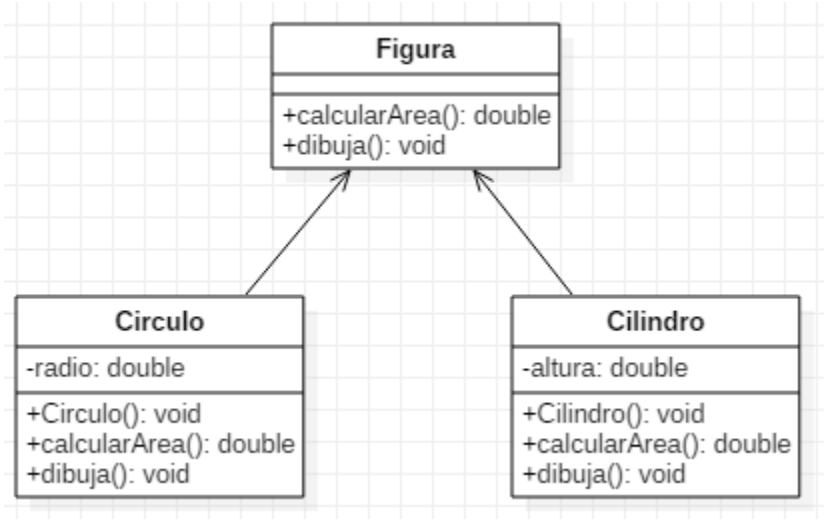
Para este ejercicio se crearon las siguientes clases:



Donde se encuentra la interfaz Ordenamiento con un método llamado ordenar. Se implementa a 2 clases llamadas BubbleSort y SelectionSort donde se sobrescribe el método de la interfaz para adaptarlo a su algoritmo de ordenamiento

Ejercicio 1.

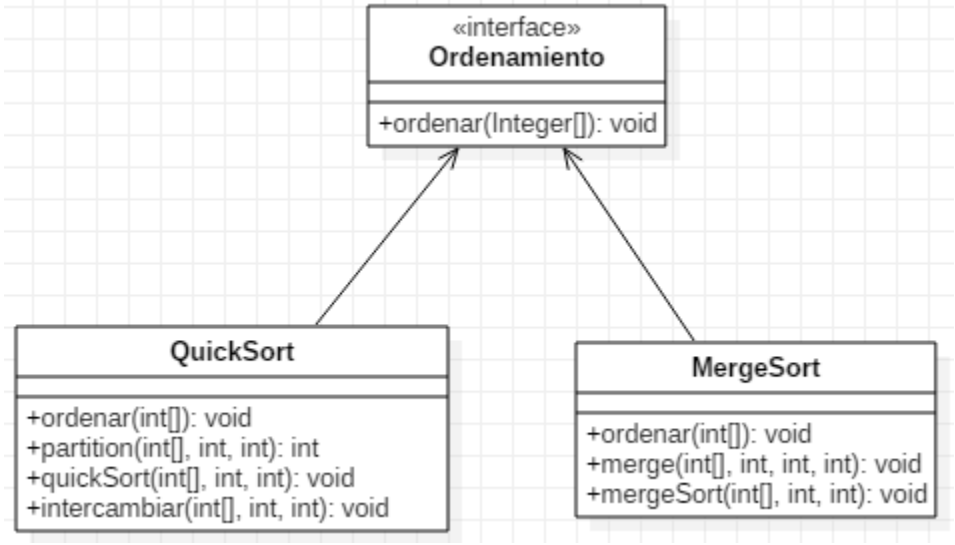
Para este ejercicio se crearon las siguientes clases:



Se tiene a la clase abstracta figura con 2 métodos. Se heredan 2 subclases, Círculo y Cilindro, donde cada una va a sobrescribir ambos métodos para su personalización.

Práctica 81.

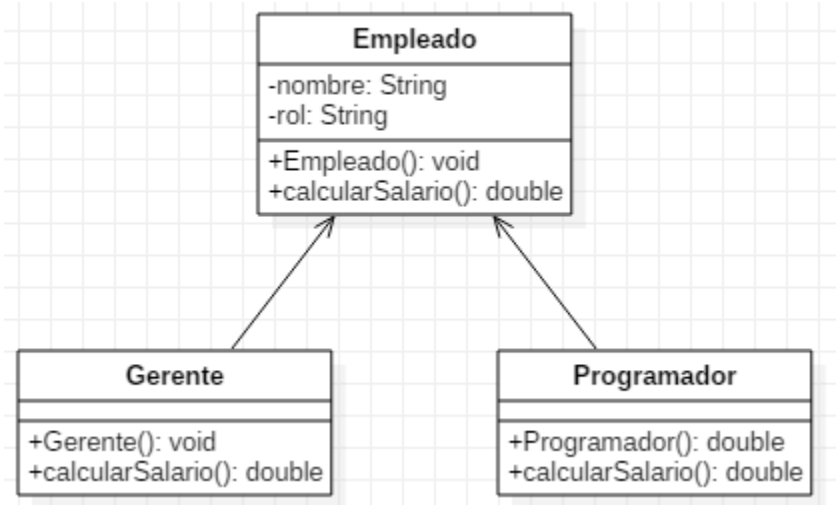
En esta práctica se crearon las siguientes clases:



Se tiene a la interfaz Ordenamiento con su método ordenar. Se implementa a las clases QuickSort y MergeSort para que se sobrescriba el método dependiendo el algoritmo de cada clase.

Practica 82

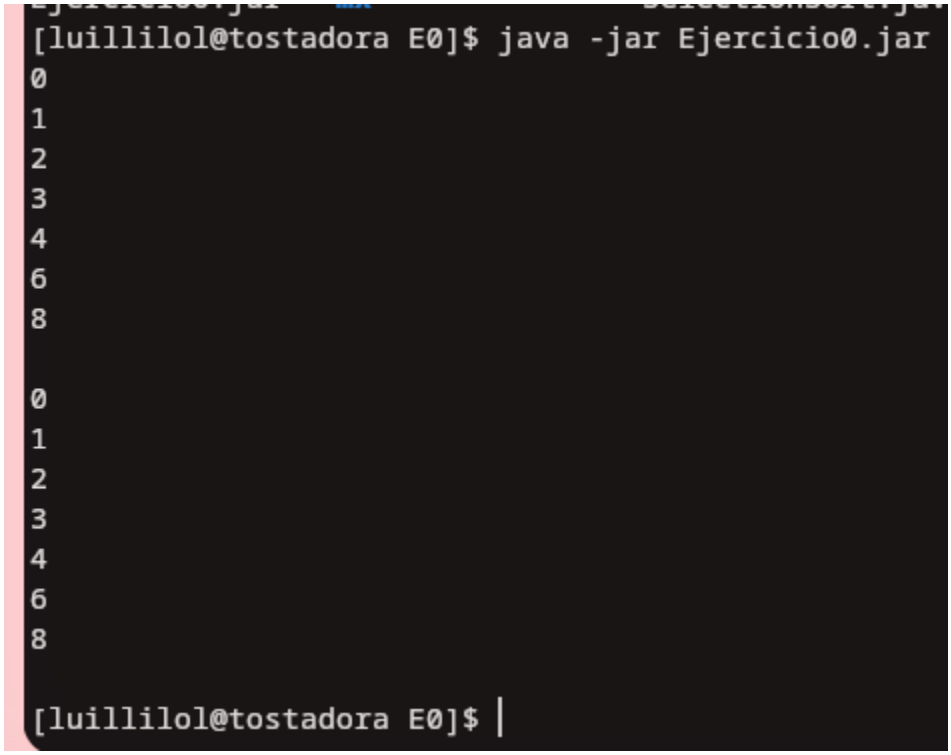
Para esta práctica se crearon las siguientes clases:



Se tiene a la clase empleado con sus atributos nombre y rol, aparte sus métodos constructor y calcularSalario. Hereda 2 clases, Gerente y Programador, cada una va a sobrescribir el método calcularSalario.

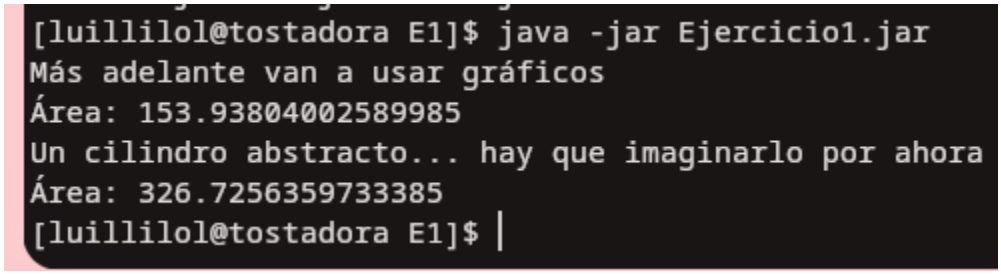
Resultados:

Ejecución del Ejercicio 0:



Se tiene el arreglo desordenado y se ordena mediante bubblesort y selectionsort

Ejecución del Ejercicio 1:



Se crean los objetos Circulo y Cilindro.y se calculan sus áreas.

Ejecución de la Práctica 81:

```
[luillilol@tostadora P81]$ java -jar Practica81.jar

Quick Sort:
88 77 35 12 49 13 40 7 64 8 73 39 83 27 16 9 100 8 35 75
7 8 8 9 12 13 16 27 35 35 39 40 49 64 73 75 77 83 88 100

Merge Sort:
75 13 49 15 79 3 65 89 86 57 42 11 8 28 68 81 93 47 48 9
3 8 9 11 13 15 28 42 47 48 49 57 65 68 75 79 81 86 89 93
[luillilol@tostadora P81]$
```

Se crean los arreglos desordenados y se ordenan mediante QuickSort y MergeSort

Practica 82:

```
[luillilol@tostadora P82]$ java -jar Practica82.jar
Nombre del empleado: Luis Falcon
Rol del empleado: Gerente general
Salario: 85000.0

Nombre del empleado: Santiago Danda
Rol del empleado: Programador Fullstack
Salario: 50000.0
[luillilol@tostadora P82]$
```

Se crean los objetos Gerente y Programador, también se muestran sus datos y su salario.

Conclusiones

Gracias a los conceptos vistos en la sección del Marco Teórico se pudieron crear entidades abstractas que solo almacenaran el comportamiento para que las clases derivadas pudieran ocuparlas. También se pudo simular la herencia múltiple mediante las interfaces (que según lo visto en clase, esa es uno de sus propósitos), creando una interfaz e implementarla en las demás clases. Es importante destacar que estos conocimientos son de gran importancia para modelar de una mejor manera a las entidades y sus relaciones con las demás. Además de poder tener un mejor entendimiento del tema y también de los próximos que se aproximen.

Referencias.

[1] P. J. Deitel and H. M. Deitel, *Cómo programar JAVA.*7th.México:PEARSON EDUCATION, 2008, p. 477.
[2] P. J. Deitel and H. M. Deitel, *Cómo programar JAVA.*7th.México:PEARSON EDUCATION, 2008, p. 423.