	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

Introducción

Práctica 5 y 6

Con base en los ejercicios realizados en la sesión práctica, mediante lenguaje Java realiza:

- Rehacer las prácticas 1 a 4 con los conceptos de POO hasta encapsulamiento y paquetes.
- Definir una convención de paquete similar a lo visto en la sesión práctica.
- Cada práctica ajustada va en su propio paquete.
- Se debe crear un jar por cada práctica.
- Se debe crear Javadoc de cada clase.
- En el reporte se debe definir el diagrama de cada clase creada.

La realización de esta práctica nos permite modificar las prácticas que hicimos anteriormente, a una forma más correcta hablando de conceptos de Programación Orientada a Objetos, aplicando los conceptos que se mencionan en las indicaciones. La realización de lo anterior nos da la habilidad de aplicar el conocimiento que se ha visto, para poder realizar proyectos de mejor calidad cada vez mejor.

Marco teórico

Durante la solución de un problema, surge la necesidad de ocultar algunos detalles de la implementación al usuario, es por eso que existe el encapsulamiento. Este se encarga de crear mecanismos para ocultar los datos al “exterior”, algunos mecanismos creados son los métodos de acceso que limitan la comunicación de las variables, estos métodos son: método modificador y método consultor, o en otras palabras, **set** y **get**.
Por necesidad de la práctica se establecerán en private los modificador de acceso de los atributos de una clase.**[1]**

Los paquetes son grupos de clases relacionadas que son usados para organizar los proyectos de una forma más estructurada, aparte de facilitar la reutilización de software y la importación de otras clases. Estos paquetes se añaden en los archivos .java mediante la palabra reservada **package** y la estructura que tendrá. Siguiendo el orden que un archivo java debe tener, la implementación de los paquetes debe de ser lo primero que deba haber: una declaración package, seguido de las declaraciones import y al final las declaraciones de las clases. **[2]**

Para un mejor orden al momento de “ejecutar” el proyecto que se está creando, se implementa el archivo .jar que funciona como un archivo comprimido donde se encuentran los datos esenciales de lo que se esté desarrollando. Los archivos .jar se crean en la terminal con el comando ‘jar’ que funciona como una herramienta para comprimir y descomprimir.

La intención del archivo .jar es similar a la de un archivo .exe (no es lo mismo, solo es una analogía para la conceptualización de ideas), gracias al archivo .jar se puede ejecutar todo el proyecto sin necesidad de compilar todas las clases manualmente. Es esencial en estos instantes que se está trabajando con paquetes, ya que estos generan carpetas que contienen los archivos .class del proyecto, y gracias a los archivos .jar la ejecución del proyecto se facilita.**[3]**

Por último, se debe crear una documentación del código realizado con todos los métodos y clases usadas, es por eso que se usa **javadoc**, que es una herramienta para generar documentación API en formato HTML, esto se puede generar mediante comentarios especiales en el archivo.java. Esta herramienta genera documentación API de todos las clases y métodos que se estén usando en el proyecto (siempre y cuando se ponga su respectivo comentario). **[4]**

Desarrollo

Práctica 1.

En esta práctica se creó la siguiente clase:

Coordenadas
-radio: double
-x1: double
-y1: double
-x2: double
-y2: double
Coordenadas(): void
convRad(double): double
calculoDistancia(): double

Donde se tiene la constante radio y las variables x1, x2, y1 y y2. También se tienen a los métodos convRad que convierte grados a radianes, calculoDistancia que calcula la distancia y el método constructor.

Práctica 2

En esta práctica se creó la siguiente clase:

Triangulo
-filas: int
-numero: int
Triangulo(int): void
calcularNumeros():void

Donde se encuentran los atributos filas y números. También se encuentra el método constructor y el método calcularNumeros que calcula el número que se debe de imprimir en el triángulo de Pascal.

Práctica 3_1.

En esta práctica se creó la siguiente clase:

Cadena
-cad: String
-c: String
-d: String
Cadena(String): void
reemplazarCaracter(): String

Donde se encuentran los atributos cad, c y d. También se encuentra el método constructor y el método reemplazarCaracter que va a reemplazar la oración por el caracter dado.

Práctica 3_2.

En esta práctica se creó la siguiente clase:

Contenedor
-lista: ArrayList
Contenedor(): void
agregarElemento(String): void
mostrarContenedor(): void
sustituirElemento(int, String): void

Donde se tiene al atributo de list que es un ArrayList que contiene una lista de nombres. También se encuentra el método constructor, el método agregarElemento que va a agregar el elemento dado a la lista, mostrarContenedor que imprime los datos de la lista, y sustituirElemento que va a sustituir el índice dado de la lista por el elemento ingresado.

Práctica 3_3.

En esta práctica se creó la siguiente clase:

ContenedorMap
-contenedorMap: HashMap<Integer, String>
ContenedorMap(): void
obtenerElemento(int): String

Donde se tiene al atributo contenedorMap que es un HashMap. También se encuentra el método constructor y el método obtenerElemento que devuelve el valor que está asociado a la llave dada.

Práctica 4.

En esta práctica se creó la siguiente clase:

CuentaBanco
-nombre: String
-cuenta: int
-depositoInicial: double
CuentaBanco(String, int, double): void
CuentaBanco(String, int): void
retirarDinero(double): void
depositarDinero(double): void
detallesCuenta(): String

Donde se tienen a los atributos nombre, cuenta y deposioInicial. También se encuentran los métodos constructores (Overloading), retirarDinero que resta dinero de la cuenta de banco, depositarDinero que aumenta dinero de la cuenta de banco y detallesCuenta que muestra la información general de la cuenta.

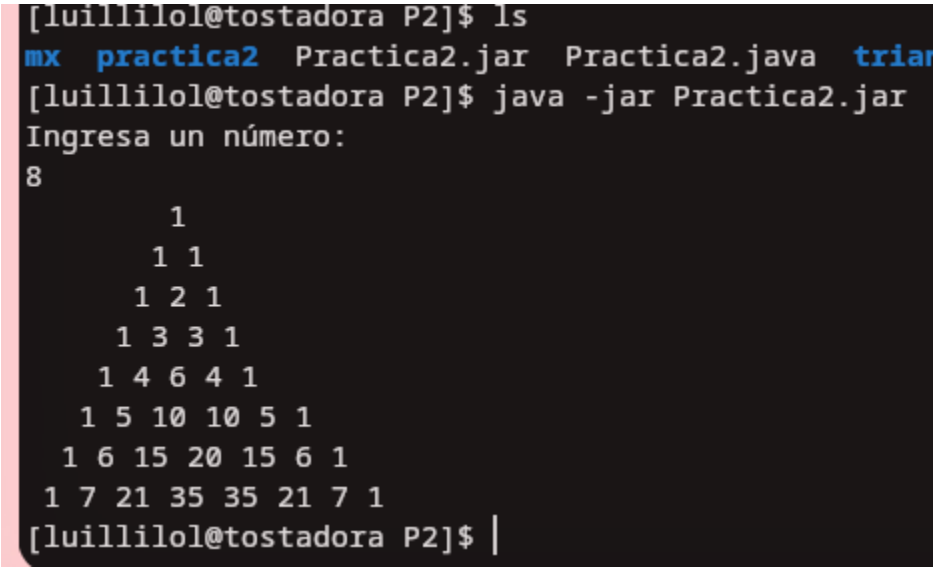
Resultados

Ejecución de la Práctica 1:

```
[luillilol@tostadora P1]$ java -jar Practica1.jar
Latitud x1: 19.33110
Longitud y1: -99.18465
Latitud x2: 19.37117
Longitud y2: -98.99644
La distancia entre los dos puntos es: 20.24216893316194
[luillilol@tostadora P1]$
```

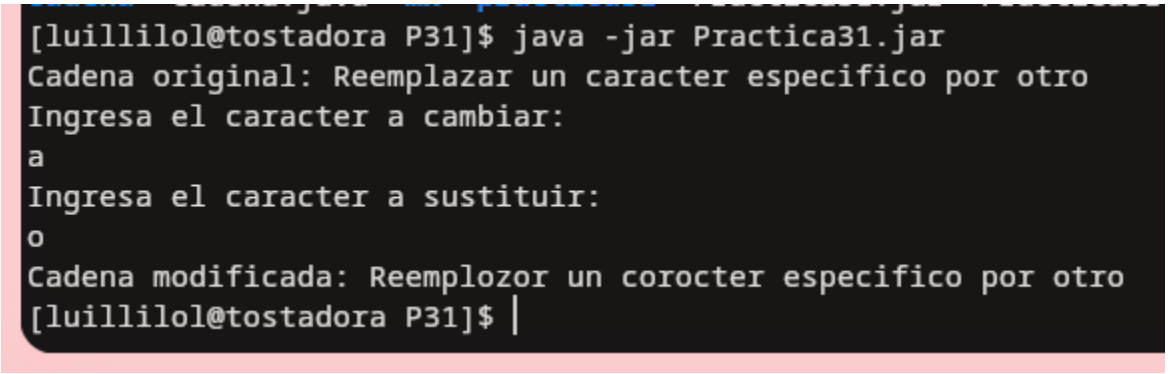
Se registran las dos coordenadas y se calcula la distancia entre esos dos puntos

Ejecución de la Práctica 2:



El usuario ingresa el número de “altura” del triángulo de Pascal y se imprime.

Ejecución de la Práctica 3_1:



Se tiene una oración original, el usuario ingresa el caracter que quiere cambiar y luego ingresa el carácter que quiere que lo sustituya, en este caso se cambiaron las **a** por las **o**.

Ejecución de la Práctica 3_2:

```
[luillilol@tostadora P32]$ java -jar Practica32.jar
Elementos del contenedor:
Luis
Adrián
Ximena
Fernando
Mauricio
Santiago
Miguel
Nombre para reemplazar:
Falcon

Elementos sustituidos:
Luis
Falcon
Ximena
Fernando
Mauricio
Santiago
Miguel
[luillilol@tostadora P32]$
```

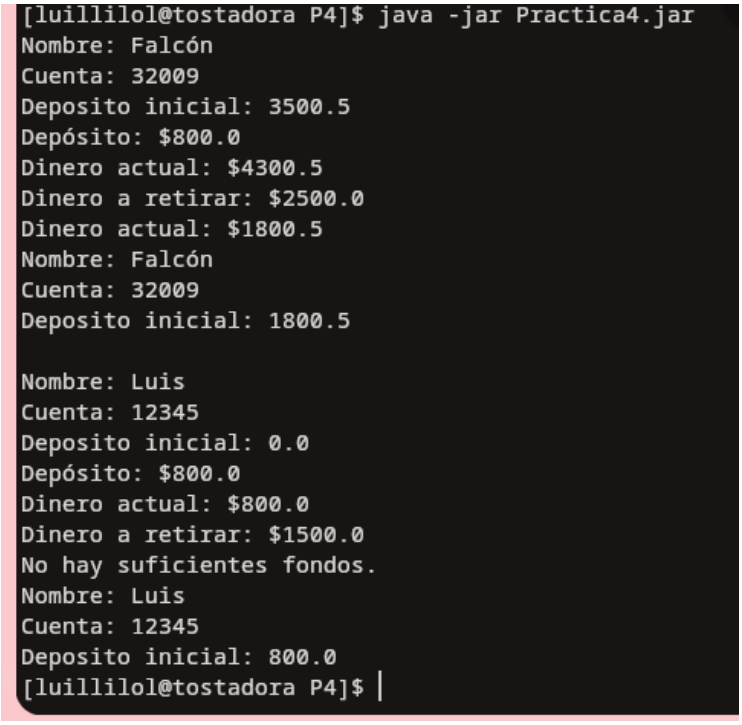
Se tiene una lista de nombres donde se va a sustituir el segundo (Adrián), el usuario ingresa el nombre Falcón para que lo sustituya luego imprime el resultado.

Ejecución de la Práctica 3_3:

```
[luillilol@tostadora P33]$ java -jar Practica33.jar
Ingresa la llave para saber el nombre del perrito (1-10)
8
El perrito es: Chihuahua
[luillilol@tostadora P33]$ java -jar Practica33.jar
Ingresa la llave para saber el nombre del perrito (1-10)
10
El perrito es: Labrador
[luillilol@tostadora P33]$ java -jar Practica33.jar
Ingresa la llave para saber el nombre del perrito (1-10)
3
El perrito es: Dalmata
[luillilol@tostadora P33]$
```

Se tiene un HashMap con 10 nombres de razas de perros, el usuario ingresa el número (key) donde quiere saber la raza, se ejecutó 3 veces para mostrar 3 razas diferentes.

Ejecución de la Práctica 4:



Se tienen dos cuentas de banco y se depositaron y retiraron dinero de esas cuentas individualmente, mostrando su estado en cada acción.

Conclusiones

Gracias a los conceptos vistos en la sección del Marco Teórico se pudieron modificar todas las prácticas que se habían hecho anteriormente para poder darle una forma más enfocada a la Programación Orientada a Objetos. Es importante destacar que estos conocimientos son de vital importancia al momento de resolver problemas, ya que nos dan una mejor organización y facilidad para su resolución. Su aplicación es muy grande, desde los videojuegos, hasta en grandes proyectos de empresas. Aparte, es un gran avance hacia el entendimiento de los conceptos más avanzados que se relacionen con este tema.

Referencias.

[1] P. J. Deitel and H. M. Deitel, *Cómo programar JAVA.*7th.México:PEARSON EDUCATION, 2008, p. 354.

[2] P. J. Deitel and H. M. Deitel, *Cómo programar JAVA.*7th.México:PEARSON EDUCATION, 2008, p. 355.

[3] Oracle(n.d.).jar[Online].Available
https://docs.oracle.com/javase/8/docs/technotes/tools/windows/jar.html

[4] Oracle(n.d.).Javadoc Tool[Online].Available
https://www.oracle.com/java/technologies/javase/javadoc.html