



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): _____

321172974

*No. de lista o
brigada:* _____

Semestre: _____

Fecha de entrega: _____

Observaciones: _____

CALIFICACIÓN: _____

ÍNDICE

| | |
|----------------------------------|-----------|
| ÍNDICE..... | 1 |
| INTRODUCCIÓN..... | 2 |
| Planteamiento del problema..... | 2 |
| Motivación..... | 2 |
| Objetivos..... | 2 |
| MARCO TEÓRICO..... | 3 |
| Encapsulamiento..... | 3 |
| Paquetes (packages) en Java..... | 4 |
| Archivos JAR..... | 5 |
| Javadocs..... | 5 |
| DESARROLLO..... | 7 |
| Ejercicios en clase..... | 7 |
| Ejercicios de tarea..... | 7 |
| Estructura de carpetas..... | 7 |
| Diagramas UML..... | 8 |
| Práctica 1..... | 8 |
| Práctica 2..... | 8 |
| Práctica 3..... | 9 |
| Práctica 4..... | 10 |
| Tablas de métodos..... | 10 |
| Modificaciones..... | 10 |
| Práctica 1..... | 10 |
| Práctica 2..... | 11 |
| Práctica 3..... | 11 |
| Práctica 3..... | 12 |
| Pruebas..... | 13 |
| Práctica 1..... | 13 |
| Práctica 2..... | 13 |
| Práctica 3..... | 14 |
| Práctica 4..... | 14 |
| RESULTADOS..... | 14 |
| Práctica 1..... | 14 |
| Práctica 2..... | 15 |
| Práctica 3..... | 16 |
| Práctica 4..... | 18 |
| CONCLUSIONES..... | 21 |
| Referencias..... | 22 |

INTRODUCCIÓN

Planteamiento del problema

Rediseñar las prácticas hechas hasta el momento usando los conceptos nuevos de programación orientada a objetos, como lo es el **encapsulamiento**.

Implementar dichas prácticas en paquetes, crear compresiones jar y documentar las clases usando Javadoc.

Motivación

Los desarrolladores de tecnologías implementadas en Java, gracias a los recursos que proporciona este lenguaje, cuentan con 3 herramientas: **paquetes** para la organización de clases; los archivos **JAR** (Java ARchive) que facilitan la distribución; y **Javadoc**, que facilita la documentación del código de sus proyectos. Un *ingeniero en computación* debe ser capaz de saber dichos conceptos y manejarlos.

Adicionalmente, el replantear las prácticas pasadas hechas con un conocimiento temprano ahora con el conocimiento presente representa una excusa perfecta para poder repasar conceptos de **encapsulamiento** y *programación orientada a objetos*.

Objetivos

Para el *proyecto 1* y para cada una de las **4 prácticas pasadas**

- ☒ Ajustarlo agregando capacidades de **encapsulamiento**
- ☒ Definir un **paquete** con base en las convenciones vistas en clase
- ☒ Crear paquete al momento de compilar, con el cual crear un archivo **JAR**
- ☒ Crear **Javadoc** de las clases creadas

Donde para los ejercicios de tarea: (prácticas pasadas)

- ☒ Definir diagrama de cada clase creada en el **reporte**
- ☒ Cada práctica debe ir ajustada en su propio **paquete**
- ☒ **JAR** por cada práctica

MARCO TEÓRICO

Al momento de dar solución a algún problema enfocándose en la identidad, atributos y comportamiento de cualquier entidad (el paradigma orientado a objetos), es preciso agruparlos y poder controlar la visibilidad de sus componentes siempre buscando la simplicidad de implementar y establecer una buena lógica que le permita a la clase comunicarse con ella misma y sus componentes, así como con otras clases y el sistema en donde coexiste.

Encapsulamiento

En la implementación de algún proyecto real, las clases normalmente ocultan los detalles de la implementación, dado que el usuario se ocupa sólo de conocer la funcionalidad sobre alguna parte del software, no en cómo se implemente dicha funcionalidad¹.

Esto permite que en dado caso de que alguna parte del programa sea reemplazada con otra versión, esto no afectaría al resto del programa. [1, 354]

Se puede pensar a alguna sección de un sistema de software, como un ambiente donde coexisten distintas entidades, cada una con atributos y comportamientos que modifican el estado de dicho ambiente, sea comunicarse o relacionarse con otras clases o consigo misma; se conoce que el sistema funciona, pero no necesariamente sabemos el cómo funciona, en ámbitos mayores solo es importante los resultados e impactos que tiene cada clase con lo demás, sin precisar detalles.

El encapsulamiento es un mecanismo aplicado sobre las clases que permite organizar y establecer un control sobre la visibilidad de sus componentes, esto para que en un dado momento que se requiera dar mantenimiento, no se debe preocupar con cambiar toda la funcionalidad completa del sistema, porque cada clase se encuentra encapsulada en su propio 'contenedor' que le permite coexistir en el sistema. El reemplazo de dicha parte del sistema por otra nueva (mantenimiento) se facilita.

En Java, el control de visibilidad se logra mediante modificadores de acceso como `private`, `protected` y `public`, lo cual permite definir qué parte del código puede acceder o modificar ciertos atributos o métodos. Esto mejora la seguridad y la integridad de los objetos al prevenir que otras partes del programa modifiquen su estado de manera no controlada.

[1, p.86]

¹ Ocultamiento de información

Paquetes (packages) en Java

Los paquetes (package) son una forma de organizar las clases de manera que se agrupan en espacios de nombres, lo que ayuda a evitar conflictos entre los nombres de clases y a estructurar de manera más clara y jerárquica un proyecto.

Se puede entender como un folder en un directorio de archivos [2]

Existen 2 categorías: [2]

- Los paquetes de la API de Java
- Los paquetes definidos por el usuario

Nos ocuparemos sobre el segundo en esta práctica.

Un paquete se define al comienzo de un archivo de código fuente mediante la palabra clave **package** seguida de la estructura del paquete, lo cual indica al compilador que dicha clase pertenece a un grupo específico.

Ejemplo de estructura de paquete:

```
package mx.unam.fi.poo.g1.p562
```

Además, los paquetes facilitan la *modularidad* del código, permitiendo que desarrolladores trabajen en distintas partes de un proyecto sin interferir en el trabajo de otros.

La creación y uso de paquetes es una práctica común en proyectos de gran escala para mantener un código organizado y manejable. [1, p.360]

Para la compilación y generación de directorios de acuerdo a la estructura de paquetes indicada en los archivos

```
javac -d [ruta de origen] archivoPrincipal.java3 [3]
```

Para ello, se debe considerar que *ruta de origen* es donde se generará la estructura de paquete (directorios) y *archivoPrincipal* el archivo que explícitamente incluye mediante `import` (estructura.paquetes.necesarios) las otras clases necesarias.

² La estructura `mx.unam.fi.poo.g1.p56` implica dicha estructura en directorios, pero también es una convención para nombrar los paquetes. Note el parecido con un dominio de internet, pero invertido. La convención especificada es la que se usará en esta práctica.

³ Es importante notar que dicho comando genera los directorios que indica la estructura de paquetes en el archivo Java, donde almacena los `.class`, dado que el comando en sí es `javac` que realiza la compilación

Archivos JAR

Los archivos JAR (Java ARchive) son archivos comprimidos que agrupan múltiples archivos **en uno solo**, como clases y recursos necesarios para la ejecución de un programa Java. [4]

Esto permite distribuir aplicaciones y bibliotecas Java en un solo archivo, facilitando tanto la portabilidad como la instalación del software.

Un archivo JAR es esencialmente un archivo ZIP con una estructura específica y puede incluir un archivo de manifiesto que define aspectos adicionales como la clase principal de la aplicación. Utilizar archivos JAR es crucial en proyectos de Java para simplificar la distribución y gestión de las aplicaciones, asegurando que todas las dependencias necesarias estén incluidas en un único paquete. [4]

```
jar -cvfe [Nombre.jar] estructura.paquetes estructura/directorios [3]
```

El nombre del jar (Nombre.jar) puede ser cualquiera y dicho nombre se ha de referenciar para ejecutarlo con:

```
java -jar Nombre.jar [3]
```

Dicho archivo es el que se puede distribuir, dado que tiene la facilidad de concentrar todo el funcionamiento de (comúnmente) varias clases en un solo fichero.

Javadocs

JavaDoc es una herramienta incluida en el *JDK* (Java Development Kit) que permite generar documentación en formato HTML a partir de comentarios estructurados en el código fuente.

Se incorporan en forma de comentarios arriba de cada clase, método o campo en el código fuente. Javadoc proporciona una descripción de la porción de código ubicada debajo de el; contiene etiquetas marcadas con @ que denotan metadata específica. [5]

Para integrar Javadoc, se escribe /**, en cada enter presionado se pondrá un * y para finalizar el Javadoc de dicha porción de código se hace con */⁴

Esta documentación genera una interfaz navegable y cómoda disponible en formato desplegable en un navegador web. (HTML) ⁵, por lo que etiquetas de dicho lenguaje de marcado está disponible en los comentarios de Javadocs

⁴ Recordando que cada documentación se refiere a la porción de código inmediata abajo

⁵ Note que el formato de los Javadocs generados es muy parecida a la documentación en la API de Java

Facultad de Ingeniería

La documentación generada incluye detalles sobre las clases, métodos, constructores y variables, proporcionado por el desarrollador⁶, mismos que pueden ser identificados fácilmente por:

- Propósito general de la clase / método
- Autor, versión
- Descripción de sus firmas

Para lo cuál, se ocupan etiquetas resaltadas por @, para así darle a conocer a Javadocs que lo que sigue en la escritura es del tipo de la **etiqueta** y por ende debe ser procesada de manera distinta para así acomodar la información donde corresponda en la generación de documentación.

Algunas etiquetas resaltables son: [3]

| Etiquetas ⁷ | Descripción de lo que documentan |
|---|---|
| @author | Menciona el autor del software |
| @version | Para indicar la versión del programa |
| @param [nombre del parámetro específico] ⁸ | Para documentar un parámetro específico de un método, se debe poner el nombre del parámetro |
| @return | Descripción de lo que retorna el método |

Para generar documentación Javadoc (toda la estructura de archivos web) de una clase:

```
javadoc -d [dirección donde se va a insertar toda la estructura] Archivo.java
```

[3]

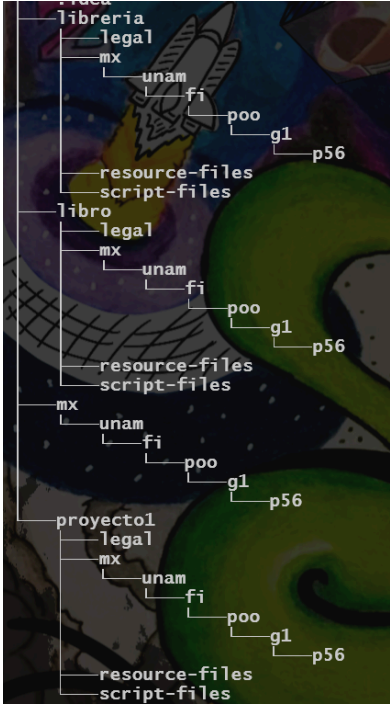
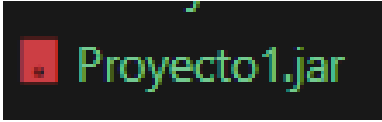
⁶ Dichas descripciones sirven como una referencia útil tanto para otros desarrolladores que mantienen el código como para aquellos que lo utilizan

⁷ Los “[]” no se escriben, solo es para indicarle al lector que lo que va dentro se modifica a

⁸ Si hay más de 1 parámetro en un método, cada parámetro debe ser documentado con dicha etiqueta

DESARROLLO

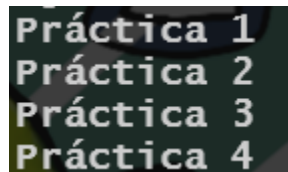
Ejercicios en clase

| Paquetes implementados | JAR creado | JavaDocs generado |
|--|--|---|
|  |  |  |

Ejercicios de tarea

Estructura de carpetas

Cada Práctica rediseñada e implementada se guardará en su respectiva carpeta:



Donde cada una contiene:⁹

- Archivos **.java**
- Directorio del paquete (Cuyo nombre es '**mx**': `mx/unam/fi/poo/g1/p56/[Práctica rediseñada]/Archivos.class`)

⁹ La práctica 3 contiene otro nivel de carpetas, dado que dicha práctica se divide en 3 ejercicios separados

Facultad de Ingeniería

- Archivo **.jar** de la Práctica
- Directorios con la documentación **Javadocs** de la clase respectiva (el nombre del directorio es el nombre de la clase documentada).

Diagramas UML

Práctica 1

| CalculadoraDistancia |
|--|
| - x1,y1,x2,y2, distancia: double - radioTierra: final doble |
| + degToRad(degress: double): double + calcularDistancia(): double |

Práctica 2

| TrianguloPascal |
|--|
| - filas: int |
| + TrianguloPascal(filas: int) + imprimirTriangulo(): void |

Práctica 3

| CadReemplazo |
|--|
| - anterior, nuevo: char - cad: String |
| Cadreemplazo(cad: String, anterior: char, nuevo: char) + reemplazar(): String + reemplazar(a, n: char): String |

| Entradas |
|--|
| entrada: Scanner(System.in) |
| + leerCadena(msg: String): String + leerChar(msg: String): char |

| ArrayListReemplazo |
|---|
| - arr ArrayList<Integer> entrada: Scanner(System.in) |
| + ArrayListReemplazo() + ArrayListReemplazo(arr: ArrayList<Integer>) + llenarArreglo(msj: String): void + reemplaza2Elemento(valor: int): void |

| SemestreCompu |
|---|
| - SemCompu: HashMap<String, Integer> |
| + SemestreCompu(semCompu: HashMap<String, Integer>) + contieneMateria(materia: String): boolean + buscarClave(materia: String): boolean |

Práctica 4

| CuentaBanco |
|--|
| - nombre, cuenta: String - saldo: double - longnumCuenta: final int = 12 |
| + CuentaBanco(nombre, String, cuenta: String, depositoinicial: double) + depositar(cantidad: double): void + registrarMovimiento(movimiento: Double): void + retirar (cantidad: double): void + implInfoCuenta(caso: String): void + impMovimientos(): void + implInfoCuenta(): void |

Tablas de métodos

Generadas automáticamente con Javadocs¹⁰

Modificaciones

A continuación se describe brevemente la razón de cada re-implementación en cada Práctica.

No todos los métodos son descritos en las siguientes tablas. solo los más importantes

Práctica 1

Clases resultantes:

| | |
|----------------------|-----------|
| CalculadoraDistancia | Practica1 |
|----------------------|-----------|

| CalculadoraDistancia |
|--|
| Clase que implementa un objeto para calcular la distancia de un punto de la tierra a otro. <ul style="list-style-type: none">- Los atributos son los puntos (2 pares de coordenadas)- El método principal de calcularDistancia() |

¹⁰ Recordando que cada clase documentada con Javadocs tiene su respectiva carpeta dentro de su respectivo directorio en la entrega; **index.html** es el punto de partida en cada uno

Facultad de Ingeniería

Se pensó para poder instanciar un objeto, después tener que ingresar los valores desde Practica1.java, e ir estableciendo los valores en el objeto mediante los **setters**.
Al finalizar, cada vez que se llame a calcularDistancia(), lo hace a partir de los valores internos del objeto.

Práctica 2

Clases resultantes:

| | |
|-----------------|-----------|
| TrianguloPascal | Practica2 |
|-----------------|-----------|

| TrianguloPascal | |
|---|---|
| Clase que implementa un objeto para imprimir el triángulo de Pascal. Se instancia un objeto de este tipo con un valor de entrada en el constructor , el cuál es el número de filas. <i>Cada vez que se requiera modificar el valor de esta variable interna, se tendrá que hacer un el setter correspondiente.</i> | |
| imprimirTriangulo() | Método principal que imprime el triángulo de Pascal de n filas, siendo n el valor de la variable interna del objeto |

Práctica 3

Clases resultantes:

| | | | |
|----|--------------------|----------|------------|
| 31 | CadReemplazo | Entradas | Practica31 |
| 32 | ArrayListReemplazo | | Practica32 |
| 33 | SemestreCompu | | Practica33 |

| CadReemplazo | |
|---|--|
| Clase que implementa un objeto para reemplazar un carácter por otro en una cadena. Los 3 son atributos privados del objeto, por lo que su modificación debe ser mediante los getters. | |

Facultad de Ingeniería

| | |
|---------------------|---|
| reemplazar() | Método sobrecargado, una variante hace el reemplazo con los valores de las variables internas, otra variante lo hace con los argumentos pasados en la llamada, pero dichos argumentos no modifican el valor de las variables internas |
|---------------------|---|

| |
|---|
| Entradas |
| Clase que implementa un objeto para la entrada de datos como: String y char |

| | |
|---|--|
| ArrayListReemplazo | |
| Clase que implementa un objeto para poder modificar el segundo valor de un ArrayList El atributo principal es un ArrayList privado, el cuál modifica su valor con los setters correspondientes | |
| llenarArreglo() | Método que solicita al usuario que llene el arreglo |
| reemplaza2Elemento() | Método que realiza el reemplazo en caso de existir el segundo elemento del arreglo |

| | |
|--|--|
| SemestreCompu | |
| Clase que implementa un objeto para almacenar asignaturas de algún semestre de ingeniería en computación. Implementa: <ul style="list-style-type: none">- Verificación de una materia en el semestre- Devolución de la clave numérica de la asignatura en los Planes de estudios de la FI UNAM | |
| Constructor | Solicita un HashMap para poder asignarlo a la variable interna de HashMap para la cuál se hará la búsqueda de la materia |

Práctica 4

Clases resultantes:

| | |
|-------------|-----------|
| CuentaBanco | Practica4 |
|-------------|-----------|

| CuentaBanco | |
|--|--|
| <i>Siendo la clase que menos cambio de todas las prácticas</i> | |
| Clase que implementa un objeto de CuentaBanco, que guarda un nombre, número de cuenta y un saldo, así como los movimientos guardados en un ArrayList, todos con acceso private, por lo que se implementan los getters y setters correspondientes | |
| depositar() | Método para depositar dinero en la cuenta, los movimientos se van añadiendo al ArrayList de movimientos como valores positivos |
| retirar() | Método para retirar dinero de la cuenta, los movimientos se van añadiendo al ArrayList de movimientos como valores negativos |

Pruebas

Prueba de correcto funcionamiento

Para todas las prácticas, se repiten las pruebas más significativas que demuestren que el programa sigue funcionando con la nueva implementación de conceptos

Práctica 1

| | |
|-------------------------|---|
| Caso normal | Del Anexo al Conjunto Principal, ambos de la Facultad de Ingeniería |
| 19.3277756, -99.1824400 | Distancia = 478.07 |
| 19.3317526, -99.1841603 | |

Práctica 2

Triángulo de Pascal de **12**

Práctica 3

- Palabra reemplazada completamente
- Arreglo de 4 elementos cambia el segundo por 16
 - Arreglo más pequeño intenta eliminación
- Búsqueda de las materias: POO y EDA II

Práctica 4

Ejecución completa del programa

RESULTADOS

Práctica 1

```
== Bienvenido a la Practica 1 ==  
Esta aplicacion calcula la distancia entre dos puntos de la tierra  
Ingrese las coordenadas del punto 1:  
Latitud (x1): 19.3277756  
  
Longitud (y1): -99.182440  
  
Ingrese las coordenadas del punto 2:  
Latitud (x2): 19.3317526  
  
Longitud (y2): -99.1841603  
La distancia entre  
el punto 1: (0.337333, -1.731060)  
y el punto 2: (0.337403, -1.731090)  
  
es : 0.4776 km  
  
==Gracias por usar la aplicacion. Saliendo del programa...==
```

Note que el resultado es el esperado e igual al de la Práctica original

```
== Bienvenido a la Practica 2 ==
Esta aplicacion calcula la distancia entre dos puntos de la tierra
Ingrese las coordenadas del punto 1:
Latitud (x1): 13

Longitud (y1): 15

Ingrese las coordenadas del punto 2:
Latitud (x2): 13

Longitud (y2): 15

La distancia entre
el punto 1: (0.226893, 0.261799)
y el punto 2: (0.226893, 0.261799)

es : 0.0000 km

==Gracias por usar la aplicacion. Saliendo del programa...==
```

Con un funcionamiento igualmente correcto para 2 coordenadas iguales

Práctica 2

```
== Bienvenido a la Practica 2 ==
Esta aplicacion despliega el Triangulo de Pascal
Ingresa el numero de filas: 12

      1
     1 1
    1 2 1
   1 3 3 1
  1 4 6 4 1
 1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1
1 8 28 56 70 56 28 8 1
1 9 36 84 126 126 84 36 9 1
1 10 45 120 210 252 210 120 45 10 1
1 11 55 165 330 462 462 330 165 55 11 1

==Gracias por usar la aplicacion. Saliendo del programa...==
```


Práctica 3

```
== Bienvenido al Ejercicio 1 de la Practica 3 ==  
Ingresa una cadena:  
Programacion Orientada a Objetos  
Ingresa el caracter que quieras reemplazar:  
O  
Ingresa el caracter por el que lo quieres reemplazar:  
o  
La cadena original es: Programacion Orientada a Objetos  
La cadena con el reemplazo es: Programacion orientada a objetos  
  
== Gracias por usar la aplicacion. Saliendo del programa... ==
```

Las O (mayúsculas) son reemplazadas por las o (minúscula)

```
== Bienvenido al Ejercicio 2 de la Practica 3 ==  
Ingresa la cantidad de elementos (enteros) que deseas agregar al ArrayList:  
3  
Ingresa el elemento 1:  
1  
Ingresa el elemento 2:  
2  
Ingresa el elemento 3:  
El arreglo original es : [1, 2, 3]  
  
Ingresa el valor (entero) que quieres reemplazar en la segunda posicion: 16  
  
El arreglo modificado es: [1, 16, 3]  
  
== Gracias por usar la aplicacion. Saliendo del programa... ==
```

El segundo elemento si existe y puede ser eliminado

Facultad de Ingeniería

```
== Bienvenido al Ejercicio 2 de la Practica 3 ==
Ingresa la cantidad de elementos (enteros) que deseas agregar al ArrayList:
1
Ingresa el elemento 1:
23

Ingresa el valor (entero) que quieres reemplazar en la segunda posicion: 45
El arreglo no tiene suficientes elementos para hacer el reemplazo

El arreglo modificado es: [23]

== Gracias por usar la aplicacion. Saliendo del programa... ==
```

El segundo elemento no existe y no puede ser eliminado

```
== Bienvenido al Ejercicio 3 de la Practica 3 ==
Este programa busca las claves de las asignaturas del tercer semestre de Ingenieria en Computacion
porque cuando un estudiante se inscribe, nunca recuerda sus claves :)

EDA
La asignatura EDA no se encuentra en el tercer semestre de Ingenieria en Computacion
Ingresa el nombre de la asignatura de tercer semestre de compu que deseas buscar:
EDA II
La clave de la asignatura EDA II es: 1317

== Gracias por usar la aplicacion. Saliendo del programa... ==
```

Búsqueda en el Mapa de EDA II

```
== Bienvenido al Ejercicio 3 de la Practica 3 ==
Este programa busca las claves de las asignaturas del tercer semestre de Ingenieria en Computacion
porque cuando un estudiante se inscribe, nunca recuerda sus claves :)

Ingresa el nombre de la asignatura de tercer semestre de compu que deseas buscar:
Poo
La clave de la asignatura P00 es: 1323

== Gracias por usar la aplicacion. Saliendo del programa... ==
```

Búsqueda de POO encontrada

```
Ingresa el nombre de la asignatura de tercer semestre de compu que deseas buscar:
hola
La asignatura HOLA no se encuentra en el tercer semestre de Ingenieria en Computacion
```

Hola no se encuentra como materia

Práctica 4

```
== Bienvenido al banco (Practica 4) ==
+-----+-----+-----+
| Nombre      | Num. Cuenta | Saldo      |
+-----+-----+-----+
| Maria Fernanda | 123456      | 20000.00   |
+-----+-----+-----+
| Movimientos  |
1: 20000.0
+-----+
Saldo: 25000.25

Saldo: 22500.13

No se puede depositar una cantidad negativa
No se puede retirar la cantidad solicitada
+-----+-----+-----+
| Nombre      | Num. Cuenta | Saldo      |
+-----+-----+-----+
| Maria Fernanda | 123456      | 22817.00   |
+-----+-----+-----+
| Movimientos  |
1: 20000.0
2: 5000.25
3: -2500.12
4: 316.87
+-----+
El numero de cuenta debe tener 12 digitos, por lo que se tomaran los primeros a 12 digitos
Numero de cuenta: 988765432123

No se puede depositar una cantidad negativa, la cuenta se creara con saldo 0
Saldo: 0.0
```

```
+-----+-----+-----+
| Nombre      | Num. Cuenta | Saldo      |
+-----+-----+-----+
| Heriberto C. | 988765432123 | 20000.00   |
| Movimientos  |
1: 20000.0
+-----+

+-----+-----+-----+
| Nombre      | Num. Cuenta | Saldo      |
+-----+-----+-----+
| Maria Fernanda | 982         | 0.00       |
+-----+-----+-----+
| Movimientos  |
No hay movimientos registrados
+-----+
```

Facultad de Ingeniería

Ejecución completa del ejercicio idéntica al del original

Explicación de ejecución

Creación de una cuenta de manera correcta, ninguna excepción

```
== Bienvenido al banco (Practica 4) ==
+-----+-----+-----+
| Nombre      | Num. Cuenta | Saldo      |
+-----+-----+-----+
| Maria Fernanda | 123456      | 20000.00   |
+-----+-----+-----+
| Movimientos  |
1: 20000.0
+-----+
```

Después de depositar 5000.25 y luego retirar 2500.12

```
Saldo: 25000.25

Saldo: 22500.13
```

Después de depositar 316.87 y querer depositar un número negativo (Excepción 1), junto con querer retirar una cantidad mayor a la del saldo actual (Excepción 2)

```
No se puede depositar una cantidad negativa
No se puede retirar la cantidad solicitada
```

```
+-----+-----+-----+
| Nombre      | Num. Cuenta | Saldo      |
+-----+-----+-----+
| Maria Fernanda | 123456      | 22817.00   |
+-----+-----+-----+
| Movimientos  |
1: 20000.0
2: 5000.25
3: -2500.12
4: 316.87
+-----+
```

Facultad de Ingeniería

Creación de cuenta bajo Excepción 3

```
CuentaBanco c2 = new CuentaBanco("Heriberto C.", "988765432123456789", 20000);
```

```
+-----+  
El numero de cuenta debe tener 12 digitos, por lo que se tomaran los primeros a 12 digitos  
Numero de cuenta: 988765432123
```

Creación de cuenta bajo Excepción 4

```
CuentaBanco c3 = new CuentaBanco("Maria Fernanda", "982", -123);
```

```
No se puede depositar una cantidad negativa, la cuenta se creara con saldo 0  
Saldo: 0.0
```

Impresión últimas dos cuentas

```
+-----+-----+-----+  
| Nombre      | Num. Cuenta | Saldo      |  
+-----+-----+-----+  
| Heriberto C. | 988765432123 | 20000.00 |  
+-----+-----+-----+  
| Movimientos |  
1: 20000.0  
+-----+  
  
+-----+-----+-----+  
| Nombre      | Num. Cuenta | Saldo      |  
+-----+-----+-----+  
| Maria Fernanda | 982         | 0.00 |  
+-----+-----+-----+  
| Movimientos |  
No hay movimientos registrados  
+-----+
```

CONCLUSIONES

Cuando un desarrollador en software trata con tecnologías, debe aprender a manejar correctamente aquellas que le faciliten a él y a sus colegas la labor de sostener, mantener y documentar todo el proyecto.

Los paquetes en java permiten poder separar y clasificar las clases, para poder utilizar solo las necesarias, siguiendo siempre el principio de modularidad.

Los archivos JAR permiten una mejor distribución, lo cuál es esencial puesto que no se necesita compartir el código fuente para poder compartir un proyecto.

Por último Javadocs es un claro ejemplo de las herramientas que proporciona Java, la cuál facilita completamente la labor de documentación y la engloba en un estándar legible y elegante.

Todo, en conjunto con un repaso de conceptos como lo es el encapsulamiento, hacen que esta práctica sea un recordatorio no solo de conceptos, sino de la labor que tiene el ingeniero en computación en aprender las diversas tecnologías y herramientas que le permitan desenvolverse como un excelente desarrollador de software y le permitan destacar laboralmente.

El link al repositorio de Github con la Práctica 5-6 es el siguiente:

<https://github.com/TheUniqueFersa/POO/tree/Practica5-6>

Referencias

- [1] P. Deitel y H. Deitel. *CÓMO PROGRAMAR EN JAVA*. Sétima edición. México: Pearson Educación México, 2008. ISBN: 978-970-26-1190-5.

- [2] “Java Packages W3Schools” W3Schools Online Web Tutorials. Accedido el 27 de septiembre de 2024. [En línea]. Disponible:
https://www.w3schools.com/java/java_packages.asp

- [3] “Programación Orientada a Objetos” notas de clase de 1323, División de Ingeniería Eléctrica, Facultad de Ingeniería de la Universidad Nacional Autónoma de México, Verano 2024.

- [4] “IBM i 7.5 Archivos JAR”. IBM - United States. Accedido el 27 de septiembre de 2024. [En línea]. Disponible:
<https://www.ibm.com/docs/es/i/7.5?topic=platform-java-jar-class-files>

- [5] “Javadocs | IntelliJ IDEA”. IntelliJ IDEA Help. Accedido el 27 de septiembre de 2024. [En línea]. Disponible:
<https://www.jetbrains.com/help/idea/javadocs.html>