	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 9-10

Escribir un programa en Java para crear un método que tome una cadena como entrada y lance una excepción si la cadena no contiene letras vocales.

- Adicionalmente se deben cumplir los siguientes requisitos:
- El ejercicio de tarea debe estar encapsulado.
- Se debe definir en un paquete.
- Se debe generar jar y javadoc.
- El reporte debe incluir diagramas estáticos y dinámicos UML.

La solución a esta práctica nos permite analizar y entender de mejor forma el funcionamiento de las excepciones que existen en Java, así como poder crear excepciones personalizadas de acuerdo a nuestras necesidades. También analizaremos el programa creado para poder modelar mediante los diagramas estáticos y dinámicos UML y poder observar cómo funcionan y qué representan cada uno.

Marco teórico

Las excepciones fungen un papel importante en el desarrollo de un programa, estas proporcionan una capacidad para gestionar situaciones que el programador sabe que pueden suceder. Pueden ayudar a prevenir que el programa se rompa, capturando el flujo y redirigirlo a otra parte o puede simplemente crear excepciones para situaciones específicas.
Una excepción es un evento que sucede cuando el flujo del programa se ve interrumpido.
Las excepciones se dividen en 2 tipos: marcados y no marcados. Los marcados son excepciones cuya captura es obligatoria, mientras que los no marcados son excepciones en tiempo de ejecución. **[1]**
Las excepciones se pueden clasificar en diferentes tipos: IOException, Runtime, Class not Found, Arithmetic, etc. Son excepciones que vienen incluidas en la clase Exception que está incluída en Object. **[2]**

Para gestionar las excepciones se usa la declaración try-catch, donde en la parte try se pone la parte que se quiere observar mientras que en los bloques catch se agregan los parámetros de excepciones para procesar la excepción que se necesita que cada bloque catch atrape. Cuando ocurre una excepción en el bloque try, se ejecuta el bloque catch donde el parámetro de excepción coincide con la excepción ocurrida. Se puede añadir un bloque finally que se ejecuta cuando el bloque try y el/los bloques catch hayan terminado su ejecución (similar a un default en un switch). **[3]**

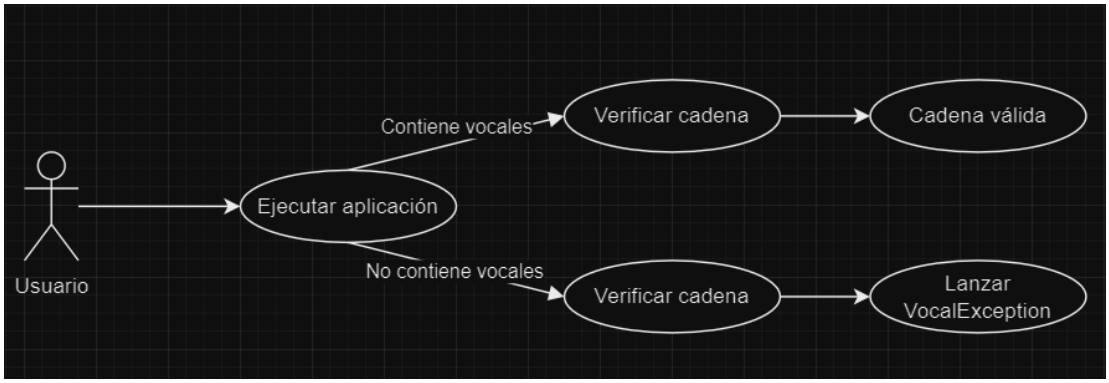
Desarrollo

Práctica 9_10

Para está practica se crearon los siguientes diagramas UML:

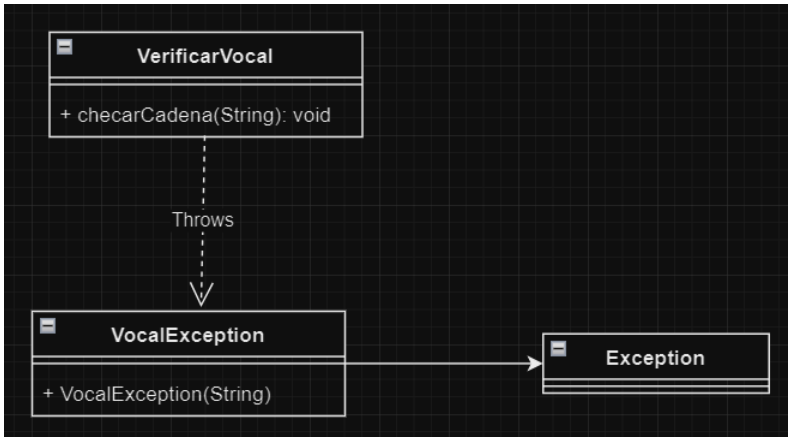
Diagramas estáticos:

Diagrama casos de uso



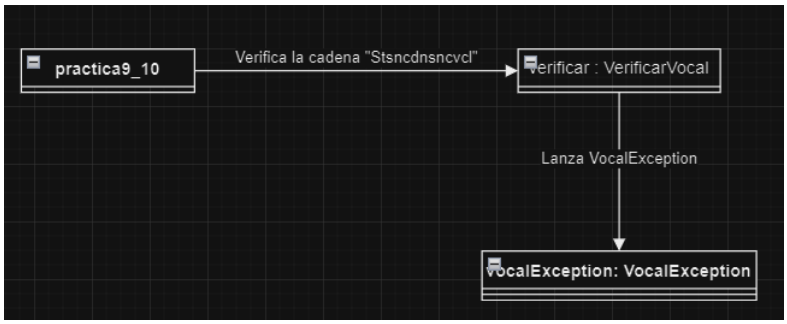
Se pueden observar elementos como el actor, los casos de uso y cómo se relacionan.

Diagrama de clases



Se observan las clases existentes y sus relaciones entre sí

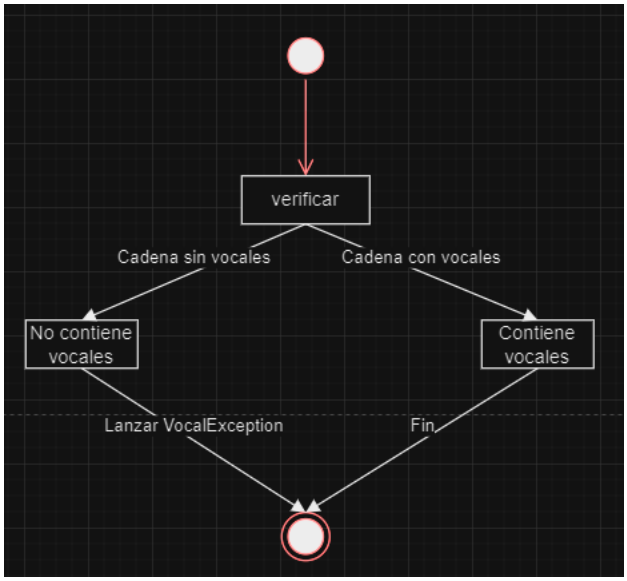
Diagrama de Objetos:



En este diagrama se encuentran elementos como los objetos que se encuentran en la ejecución en un momento dado y su medio de relación con los otros objetos existentes.

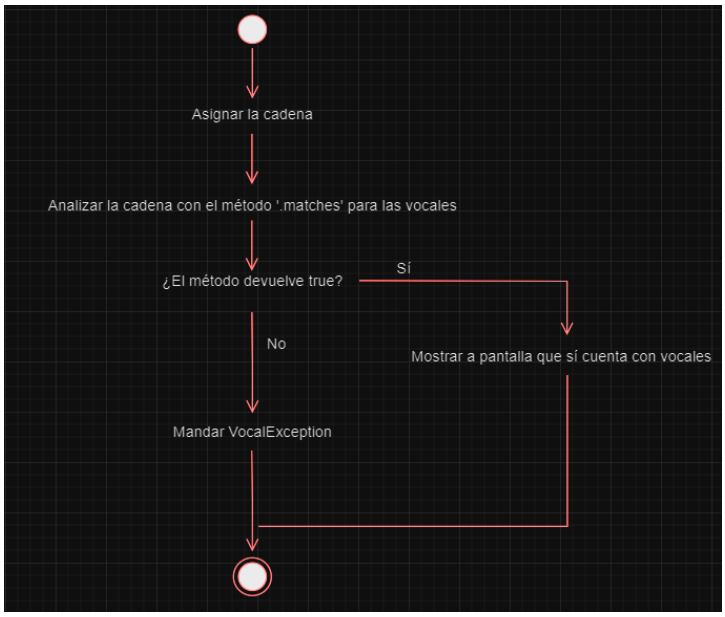
Diagramas dinámicos:

Diagrama de estados:



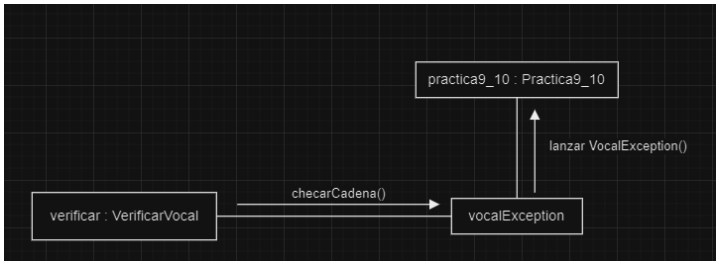
Aquí se visualizan las entradas y los estados del objeto, relacionados mediante nodos y transiciones.

Diagrama de actividades



Este diagrama muestra las actividades que se realizan en la ejecución y las condiciones para el manejo del flujo de la ejecución.

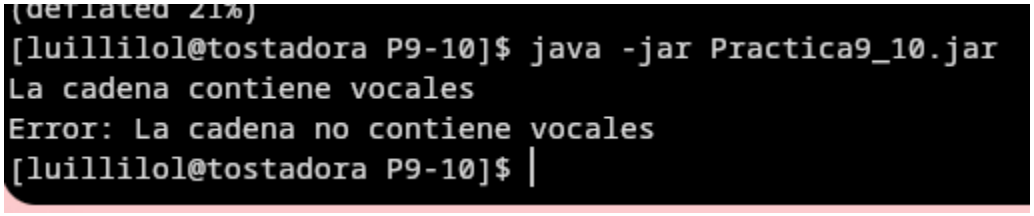
Diagrama de comunicación:



En este último diagrama podemos observar los elementos y las clases existentes y la comunicación hacia los demás mediante ciertos métodos.

Resultados:

Ejecución de la práctica 9_10:



Se manda primero la cadena “Esta es una cadena con vocales”, se compara y se manda el mensaje predeterminado. Cuando se manda el mensaje “Stsncdnsnvcls”, se compara y manda el error que hemos creado, donde indica que la cadena no contiene vocales.

Conclusiones

Se pudo observar que nosotros podemos crear nuestras propias excepciones, no solo usar las que ya están creadas por *default* de la clase Exception, esto es una habilidad realmente importante ya que nos abre una puerta de posibilidades para poder obtener y crear nuestras condiciones para el flujo de la ejecución del programa. Se usaron conocimientos previos como herencia y polimorfismo para poder crear estas excepciones particulares. Otra parte importante son los diagramas UML, que nos ayudaron a modelar el programa en forma de tablas o diagramas. La capacidad de poder generar diagramas como los anteriores es importante ya que en el medio profesional sirven para mostrar los sistemas, entender el diseño y también la implementación realizada.

Referencias.

[1] IBM. (2024, Abr, 14). Excepciones Java [Online]. Available https://www.ibm.com/docs/es/i/7.3?topic=driver-java-exceptions

[2] “Programación Orientada a Objetos”, class notes for 1323, Facultad de ingeniería, 2024

[3] P. J. Deitel and H. M. Deitel, *Cómo programar JAVA.*7th.México:PEARSON EDUCATION, 2008, p. 584.