|  |  |  |
| --- | --- | --- |
|  | **Carátula para entrega de prácticas** | |
| Facultad de Ingeniería | | Laboratorio de docencia |

[[1]](#footnote-1)

Laboratorios de computación

salas A y B

|  |  |
| --- | --- |
| *Profesor:* | Ing. José Antonio Ayala Barbosa |
| *Asignatura:* | Programación Orientada a Objetos |
| *Grupo:* | 1 |
| *No de Práctica(s):* | 10 |
| *Integrante(s):* | Mejia Valdiviezo Eduardo Javier 317078585 lalo.ejmv@gmail.com  Sánchez Ibarra Giovanny 315268188 gioseninn@gmail.com |
| *No. de Lista :* | 19  37 |
| *Semestre:* | 2021-1 |
| *Fecha de entrega:* | 7 / Enero / 2021 |
| *Observaciones:* |  |
|  |  |

CALIFICACIÓN: \_\_\_\_\_\_\_\_\_\_

Este documento contiene las actividades realizadas en la clase de laboratorio de Programación Orientada a Objetos, relacionado a capturar excepciones de un bloque de código seleccionado y errores de un bloque de código seleccionado.

# Previo

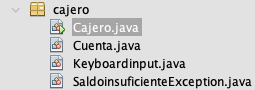
Para el previo se debía realizar las 3 clases que se muestran a continuación:

1.- Cajero

2.- Cuenta

3.- Saldoinsuficiente

Mismas clases que se fueron modificando más adelante.



# Introducción

# 

Objetivo: Identificar bloques de código propensos a generar errores y aplicar técnicas adecuadas para el manejo de situaciones excepcionales en tiempo de ejecución.

En esta practica 10 veremos la importancia de conocer nuestros códigos y saber cómo es que funcionan ya que las aplicaciones pueden presentar distintos errores.

Estos se pueden clasificar en:

• Errores sintácticos: Son todos aquellos errores que se generan por infringir las normas de escritura de un lenguaje: coma, punto y coma, dos puntos, palabras reservadas mal escritas, etc. Normalmente son detectados por el compilador o el intérprete.

• Errores semánticos (o lógicos): Son errores más sutiles, se producen cuando la sintaxis del código es correcta, pero la semántica o significado no es el que se pretendía.

• Errores de ejecución: Son errores que se presentan cuando la aplicación se está ejecutando. Su origen puede ser diverso, se pueden producir por un uso incorrecto del programa por parte del usuario, o se pueden presentar debido a errores de programación, o debido a algún recurso externo al programa. Un error en tiempo de ejecución provoca que la aplicación termine abruptamente. Los lenguajes orientados a objetos proveen mecanismos para manejar errores de ejecución.

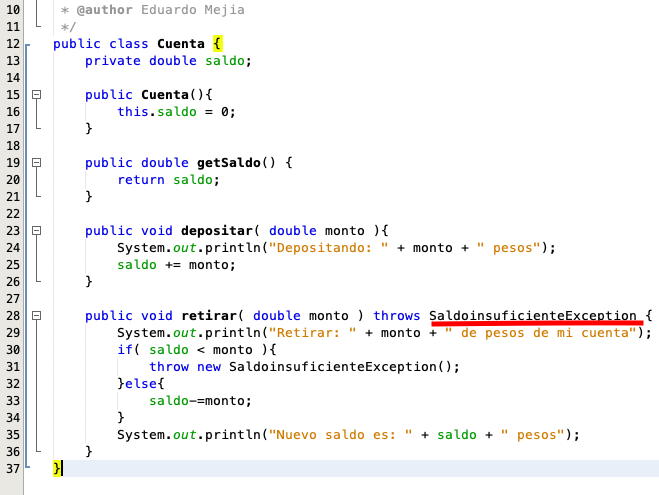
# Desarrollo

A continuación, se muestra las actividades, los paquetes de nuestro proyecto, mismos que contienen todos los procesos, así cómo los programas hechos, conteniendo cada uno, una breve explicación, composición y ejecución.

## Cajero

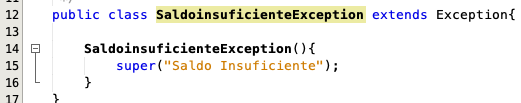
* Clase Cuenta

En esta parte encontramos un poco del cómo va a funcionar una cuenta, primero encontramos el saldo inicializado en 0, también las partes de depositar y retirar, en depositar se realiza únicamente una suma al saldo anterior, por otra parte, en retirar se realiza la resta y en caso de solicitar un retiro mayor al saldo que se encuentra en la cuenta, se hace uso de “SaldoinsuficienteException”, más adelante veremos qué realiza esta función.



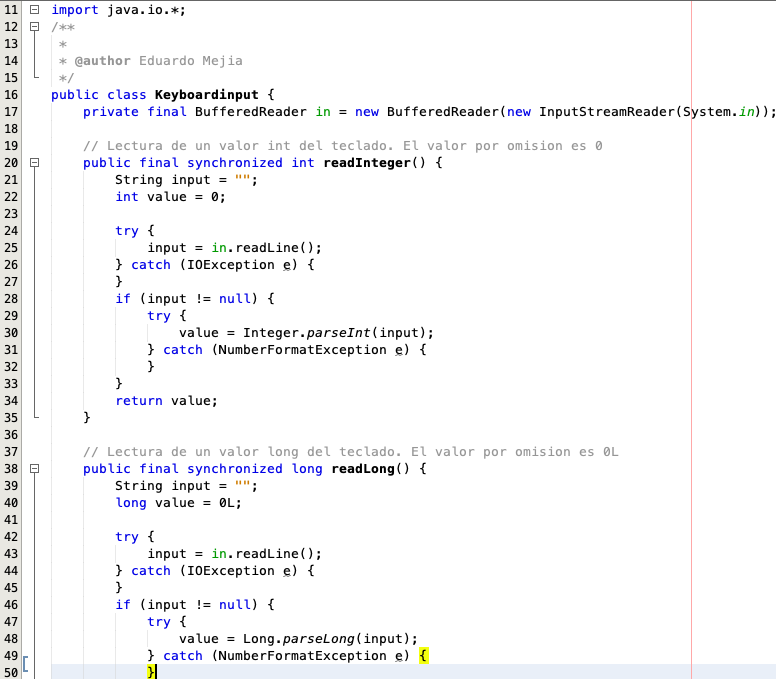
* Clase SaldoinsuficienteException

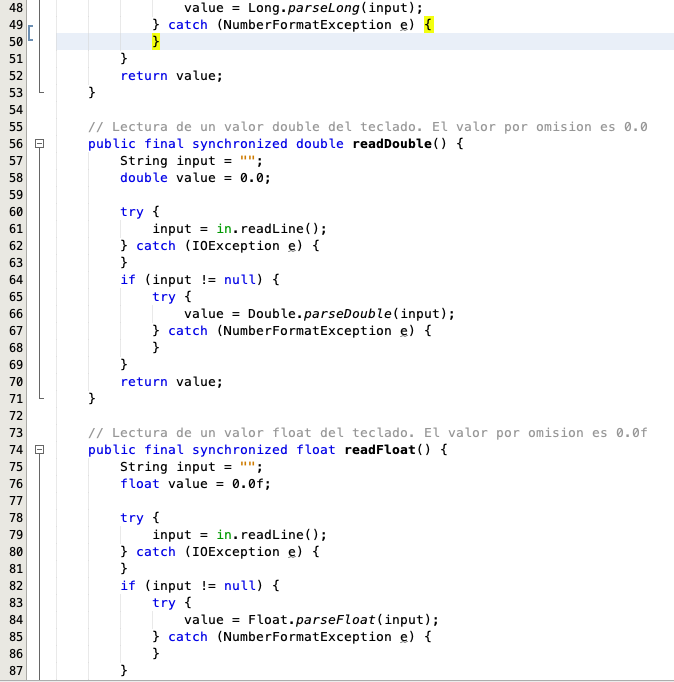
En esta clase se encuentra el método que se usó anteriormente, mismo que nos indica que el saldo es insuficiente en dado caso que en un retiro se solicite un dinero mayor al que hay en la cuenta.

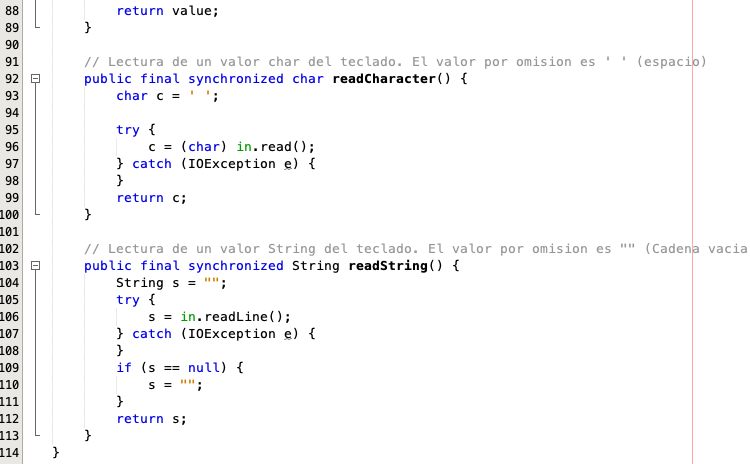


* Clase Keyboardinput

En esta parte aunque parece algo muy complicado, se encuentran las lecturas del teclado en valores int, long, doublé, float, char y string.

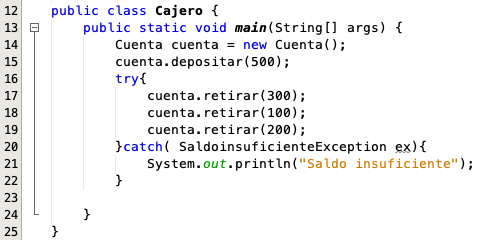






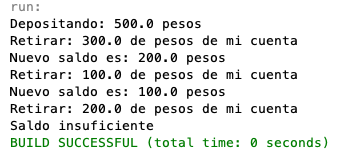
* Clase Cajero

En esta clase encontramos nuestro main, en donde primero se genera una cuenta a la que con el método “depositar”, se le depositan 500 pesos, posteriormente se hacen algunos retiros de distintos valores, hasta que arroja el “Saldo insuficiente”.



* Ejecución Cajero

A continuación, podemos observar cómo es que funciona nuestro programa, primero se realiza el depósito de los 500 pesos y después se hacen algunos retiros en donde podemos observar que si se pasa el límite nos muestra el mensaje de “Saldo insuficiente”.

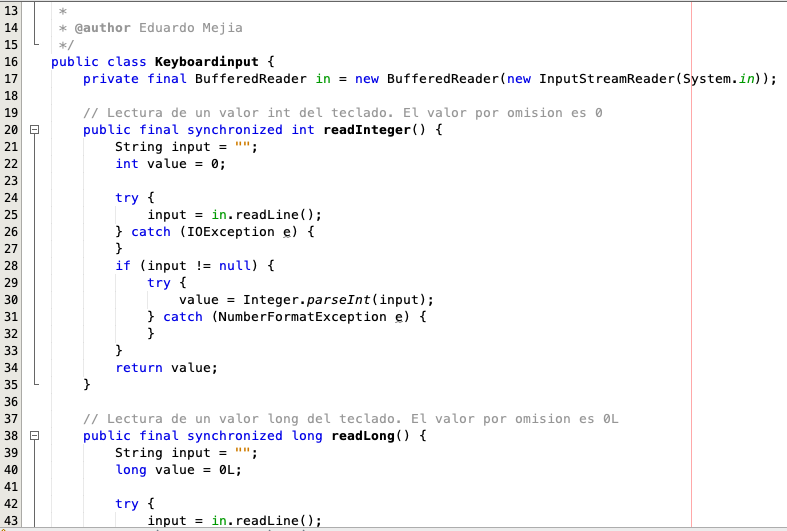


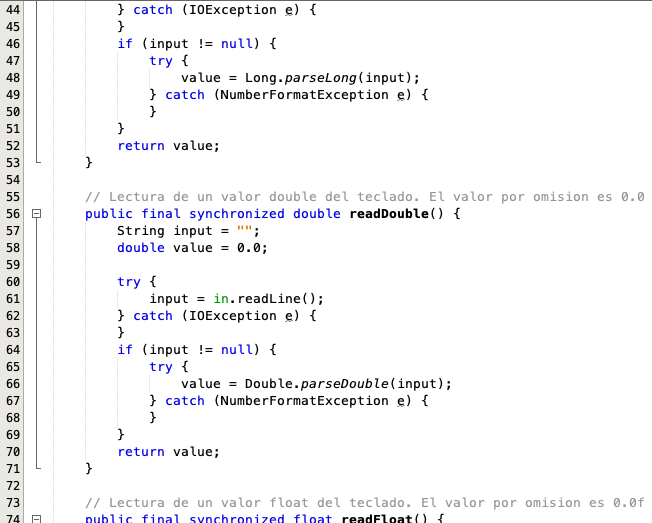
## POO10

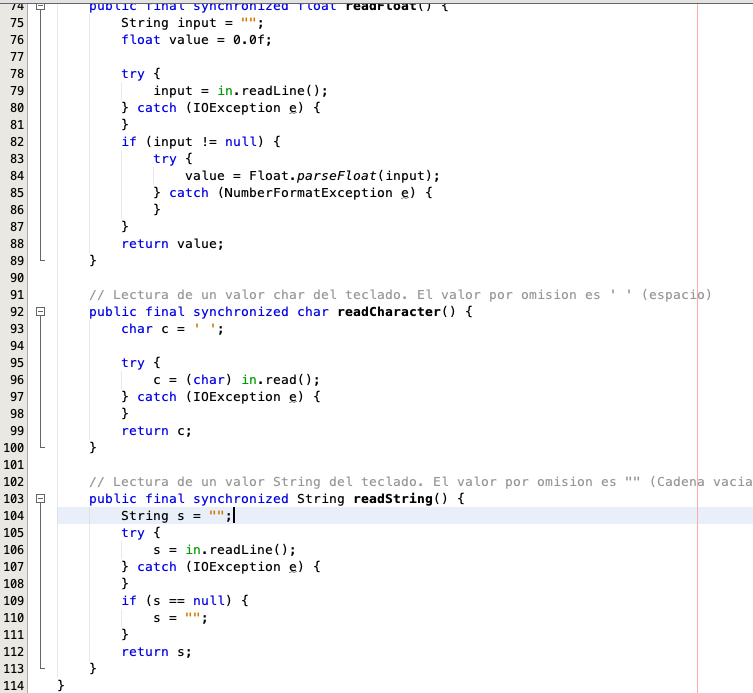
En esta parte de la práctica, podemos visualizar un poco más acerca de algunos errores, mismos que se dividieron 4 partes, cómo pueden ser los apuntadores fuera de rango, las divisiones entre cero y algunas excepciones aritméticas.

* Keyboardinput

Al igual que el Keyboardinput anterior, nos sirve para las lecturas del teclado en valores int, long, doublé, float, char y string.

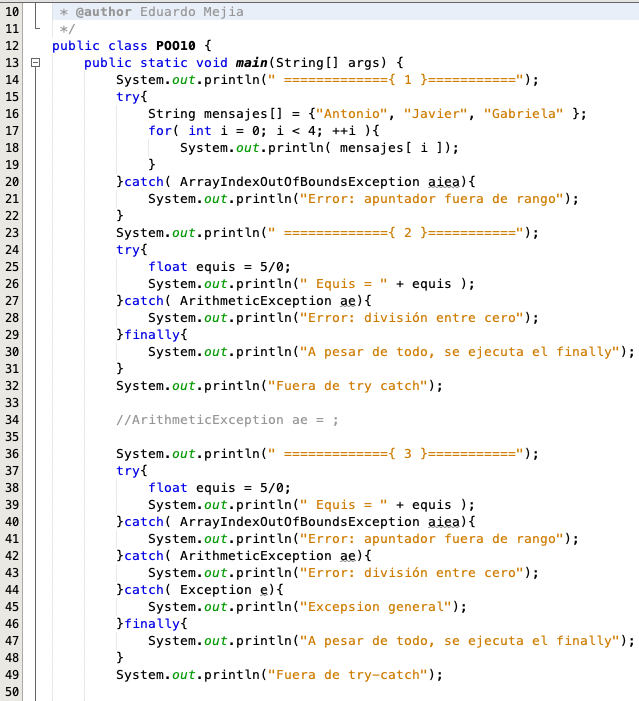






* POO10

Aquí es donde se encuentra la codificación de las 4 partes mencionadas anteriormente:



A partir de la línea 72, se realizaron las funciones de las divisiones utilizadas.



# Conclusión

Visualizando los errores a mayores rasgos, es necesario conocerlos ya que toda aplicación presenta errores, por eso es que también se deben conocer los distintos tipos que existen, para así saber cómo solucionarlos.

Es muy importante saber cómo interactúan las clases y los objetos y como esto nos puede facilitar las cosas ya que podemos crear un objeto o clase que represente varias cosas como en este caso un cajero que contiene una cuenta que se pueden realizar tales operaciones o métodos con los datos que introduzcamos o que tengamos al ejecutar el programa.

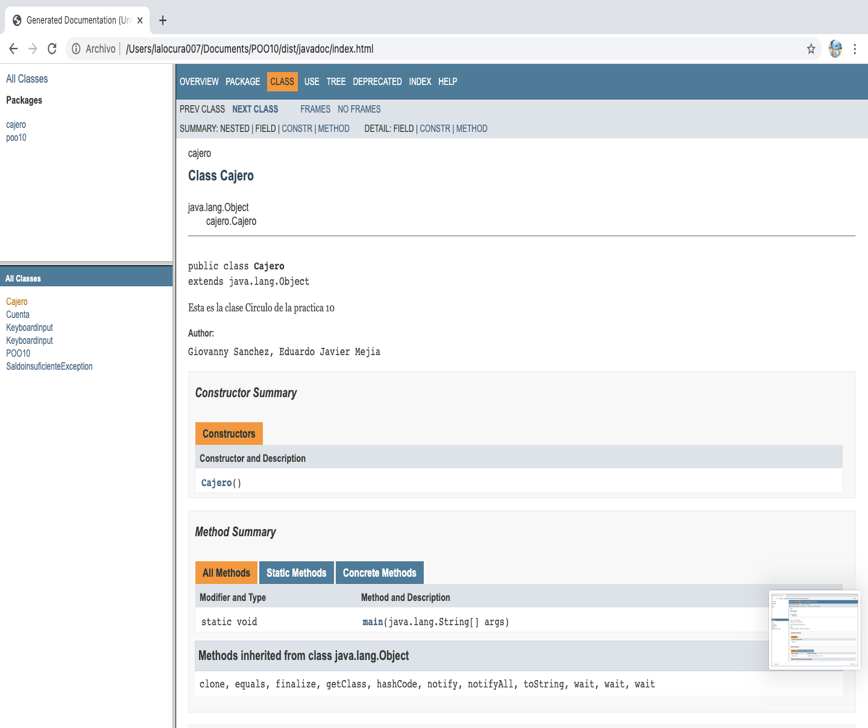
En pocas palabras podemos crear lo que queramos con la creatividad necesaria podemos hacer programas o muy complejo o muy prácticos y sencillos para todo público.

# Referencias

<http://profesores.fi-b.unam.mx/annkym/LAB/poo_p10.pdf>

DOCUMENTACIÓN:

<file:///Users/lalocura007/Documents/POO10/dist/javadoc/index.html>



1. [↑](#footnote-ref-1)