	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación
Salas A y B

Profesor: Ing. José Antonio Ayala Barbosa

Asignatura: Programación Orientada a Objetos

Grupo: 1

No de Práctica(s): Práctica 10 Excepciones y errores

Integrante(s): José Luis Arroyo Chavarría
Francisco Moisés Barrera Guardia
Juan Manuel Peralta Rodríguez
Rodrigo Daniel Reséndiz Cruz

No. de Lista o Brigada: Equipo I

Semestre: 3

Fecha de entrega: 7/1/2021

Observaciones:

CALIFICACIÓN: _____

I. Objetivo

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

II. Introducción

En esta práctica vimos el manejo de errores y excepciones en los programas, los cuales, son susceptibles a fallar debido a parámetros no aceptados por el mismo, para ello vamos a echar un vistazo a los conce4ptos detrás de la realización de las actividades.

De entrada, en un programa se pueden identificar tres tipos de errores:

- Errores sintácticos: también llamados errores de sintaxis, son errores que provienen de una mal escritura de las palabras reservadas que se permiten en el código, expresiones incompletas o erróneas y por variables no declaradas.
- Errores lógicos: Los errores lógicos son aquellos que no impiden el funcionamiento del programa como los sintácticos, pero hacen que el programa obtenga resultados a operaciones no deseados, por ejemplo, una respuesta incorrecta al realizar una operación matemática.
- Errores en tiempo de ejecución: son aquellos que se llevan en el transcurso de ejecución de un programa y se producen cuando el usuario introduce valores que el programa no pidió y no puede procesar como divisiones entre cero o envíos de cadenas cuando se pide un número.

También existen las excepciones, los cuales son errores prevenibles dentro del programa, como las divisiones entre cero. Dentro de la programación, al encontrar una de estas excepciones, podemos cacharlas y encontrarle una salida.

En java se pueden utilizar declaraciones para manejar estos errores y excepciones, estos son el try, catch y finally.

- try: nos permite probar los errores mientras es ejecutado.

- catch: nos permite definir un bloque de código en casdo de que se presente un error
- finally: permite ejecutar un bloque de código después de un try-catch sin importar el resultado.

Además de estas palabras reservadas tenemos a throw, el cual nos permite lanzar excepciones.

Cabe decir que nosotros también podemos implementar nuestras propias excepciones e introducirlas al código.

III. Desarrollo

Actividades:

- Capturar excepciones de un bloque de código seleccionado.
- Capturar errores de un bloque de código seleccionado.

<u>Actividad 1: implementación de un try-catch.</u>

```
System.out.println("1++++++++++++++++++++++++");
try{
String mensajes[] = {"uno", "dos", "tres"};
for (int i = 0; i < 4; i++) {
    System.out.println(mensajes[i]);
}
}catch(ArrayIndexOutOfBoundsException aie){
    System.out.println("Error: fuera de rango");
}
```

Ilustración 1.-implementación de un try catch

En la primera actividad procedimos a utilizar las palabras reservadas try-catch para manejar un error producido en un arreglo de cadenas, en el programa se pide imprimir los primeros cuatro elementos de la lista, pero, solo disponemos de tres elementos por lo que el programa nos daría un error.

Por eso mismo se utiliza try para intentar la operación, si sucede el error (siempre sucede), con catch definimos la excepción a precisar y con ello imprimimos un mensaje que diga que existe un error de rango en el arreglo.

```
1++++++++++++++++++++++++
uno
dos
tres
Error: fuera de rango
```

Ilustración 2.- Ejecución de la primera actividad.

En la ejecución se imprimen los elementos del arreglo y como intenta imprimir un cuarto elemento inexistente, el programa nos manda un mensaje que dice que hay un error.

<u>Actividad 2: implementación de un try-catch-finally.</u>

```java
System.out.println("2++++++++++++++++++++++++++++");
try{
float equis = 5/0;
System.out.println("X = "+equis);
}catch(java.lang.ArithmeticException ae){
    System.out.println("Error: división entre cero");
}finally{
    System.out.println("A pesar de todo, se ejecuta finaly ewe");
}
System.out.println("Fuera del try-catch");
```

Ilustración 3.-código de la segunda actividad.

En la segunda actividad se intenta hacer una operación matemática, pero se va a producir un error porque se hace una división entre cero.

Por esto mismo hacemos uso de Arithmetic Exception el cual contempla esta operación y con ello se cacha el error, permitiéndonos imprimir un mensaje de error y evitar que el programa pare, es así que utilizamos un finally para imprimir otro mensaje sin importar el resultado obtenido.

```
2++++++++++++++++++++++++++++
Error: división entre cero
A pesar de todo, se ejecuta finaly ewe
Fuera del try-catch
```

Ilustración 4.- Ejecución de la actividad dos.

En la ejecución se imprime el error ya que intentamos dividir entre cero y se imprime un mensaje que se alojaba dentro del finally.

<u>Actividad 3: implementación de un catch anidado.</u>

```java
System.out.println("3++++++++++++++++++++++++++++");
try{
float equis = 5/0;
System.out.println("X = "+equis);
}catch(ArrayIndexOutOfBoundsException aie){
   System.out.println("Error: fuera de rango");
}
catch(java.lang.ArithmeticException ae){
    System.out.println("Error: división entre cero");
}catch(Exception e){
    System.out.println("Excepcion general");
}
finally{
    System.out.println("A pesar de todo, se ejecuta finaly ewe");
}
System.out.println("Fuera del try-catch");
```

Ilustración 5.- código de la actividad 3

En esta actividad se buscaba mostrar que un catch podía ser anidado, es decir, que la estructura de un try-catch podía abarcar varias excepciones. En el código se hace lo mismo que la anterior actividad solamente que intenta verificar i hay un erro de rango en arreglo y luego intenta detectar un problema aritmético correspondiente a la división entre cero, como no aplica el primer catch, prueba el segundo y por eso va a imprimir el error de impresión entre cero.

```
3++++++++++++++++++++++++++++
Error: división entre cero
A pesar de todo, se ejecuta finaly ewe
Fuera del try-catch
```

Ilustración 6.- Ejecución de la actividad 3.

En la ejecución no se imprime el error de rango de arreglos, ya que no cumple con dicha excepción, pero si imprime el error de la división entre cero.

Actividad 4: try-catch con métodos (throw).

```
System.out.println("4++++++++++++++++++");
try{
int division = division(8, 0);
System.out.println("Division ="+division);
}catch(ArithmeticException ae){
    System.out.println("Error: división entre cero");
}


public static int division(int a, int b) throws ArithmeticException{
    int c;
    c= a/b;
    return c;
}
```

Ilustración 7.- Código de la actividad 4.

En esta actividad se hizo la operación de división, pero se efectuaba al hacerse en un método llamado división el cual utiliza la palabra reservada trows la cual especifica que puede tirar una excepción, la cual es de tipo aritmético, después toma los dos numero a dividir y efectúa la operación y si aplica la excepción, en el catch se toma esa excepción y avienta el mensaje de error.

```
4++++++++++++++++++
Error: división entre cero
```

Ilustración 8.- Ejecución de la actividad 4

En la ejecución se puede ver que nos imprime el mensaje de error, debido a que la operación es entre cero.

Actividad 5: try-catch con métodos (throw).

```
System.out.println("5++++++++++++++++++");
 try{
int division2 = division2(8, 0);
System.out.println("Division ="+division2);
}catch(ArithmeticException e){
    System.out.println("Error: división entre cero");
}


public static int division2(int a, int b) throws ArithmeticException{
    if(b==0){
        throw new ArithmeticException();
    }int c=a/b;
    return c;
}
```

Ilustración 9.- Código de la actividad 5

En esta actividad se repitió el proceso de la cuarta actividad, pero en el método se evalúa el segundo valor a operar, si es igual a cero, el método tira la excepción y se devuelve el valor del resultado, por lo que imprimirá el mensaje de error.

```
5++++++++++++++++++
Error: división entre cero
```

Ilustración 10.- Ejecución de la actividad 5.

En la ejecución notamos que se imprime el mensaje de error ya que en el método se detectó que el segundo digito a dividir es 0 y por lo tanto tiró la excepción aritmética.

Actividad 6: Implementación de excepciones.

Para la última actividad se implementó una excepción para un sistema de cajero automático, dicha excepción impide que el usuario retire cantidades de dinero en una cuenta sin dinero o con menos dinero del pedido, a continuación, se presentarán las clases.

```java
package Cajero;
public class Cuenta {
    private double saldo;
    public Cuenta(){
        saldo = 0;
    }

    public void depositar(double monto){
        System.out.println("Depositando..."+monto);
        saldo += monto;
        System.out.println("nuevo saldo: "+saldo);
    }

    public void retirar(double monto) throws SaldoInsuficienteException{
        System.out.println("Retirando..."+monto);
        if(saldo < monto){
            throw new SaldoInsuficienteException();
        }else{
            saldo-= monto;
            System.out.println("nuevo saldo: "+saldo);
        }
    }
}
```

Ilustración 11.- Código Actividad 6 (Cuenta)

Esta clase se encarga de la funcionalidad del cajero automático, el cual tiene dos acciones, depositar dinero o retirar dinero a la cuenta. En el método de depositar se evalúa el saldo con el monto, si el saldo es menor al monto solicitado tira la excepción que creamos, en caso contrario se resta el dinero a la cuenta.

```java
package Cajero;

public class SaldoInsuficienteException extends Exception{
    SaldoInsuficienteException(){
        super("saldo insuficiente");
    }
}
```

Ilustración 12.- Código de actividad 6 (Excepción creada)

En otra clase se formó la excepción la cual se llamó SaldoInsuficienteException que extiende de exception de ella se va a introducir un mensaje para dicha excepción el cual especifica que no hay saldo suficiente.

```java
package Cajero;
public class Cajero {
    public static void main(String[] args) {
        Cuenta cuenta = new Cuenta();
        try{
        cuenta.depositar(500);
        cuenta.retirar(500);
        cuenta.retirar(100);
        }catch(SaldoInsuficienteException ex){
            System.out.println("Error: saldo insuficiente");
        }
    }
}
```

Ilustración 13.- Código Actividad 6 (clase principal)

En la clase principal hacemos uso de los métodos por medio de un try de nuestra clase cuenta, por lo que primero depositamos 500 pesos, lego retiramos 500 pesos y finalmente intentamos retirar 100 pesos, como ya no hay dinero en la cuenta se utiliza un catch que utiliza la excepción SaldoInsuficienteException para darle al usuario un mensaje que le diga que no tiene el saldo para efectuar el retiro correspondiente.

```
run:
Depositando...500.0
nuevo saldo: 500.0
Retirando...500.0
nuevo saldo: 0.0
Retirando...100.0
Error: saldo insuficiente
BUILD SUCCESSFUL (total time: 0 seconds)
```

Ilustración 14.- Ejecución Actividad 6

En la ejecución podremos ver cada acción tomada por el cajero automático, pero cuando se intenta retirar 100 pesos después de mostrarnos 0 pesos en la cuenta. Se nos muestra un mensaje que dice: “Error: saldo insuficiente”

IV. Código fuente

- **POOP9:**

```
package poop9;
public class POOP9 {
   public static void main(String[] args)
{
System.out.println("1**********************
**************************");
      try{
         String mensajes[] = {"Antonio",
"Javier", "Gabriela"};
         for (int i = 0; i < 4; i++){

System.out.println(mensajes[i]);
         }
}catch(ArrayIndexOutOfBoundsExcep
tion aie){
         System.out.println("Error:
apuntador fuera de rango");
      }
System.out.println("2**********************
**************************");
      try{
         float equis = 5/0; //primer
ejemplo: 5/2
         System.out.println("Equis =
"+equis);
      }catch(ArithmeticException ae){
         System.out.println("Error:
division entre cero");
      }finally{
         System.out.println("A pesar de
todo, se ejecuta finally");
      }
      System.out.println("Fuera de try-
catch");
      //ArithmeticException ae = lo que
lance la excepcion
System.out.println("3**********************
**************************");
      try{
         float equis = 5/0;
         System.out.println("Equis =
"+equis);
      }catch(Exception e){
         System.out.println("Excepcion
general");
      }finally{
         System.out.println("A pesar de
todo, se ejecuta finally");
      }
      System.out.println("Fuera de try-
catch");
System.out.println("4**********************
**************************");
      //Propagancia de exceptions
      try{
         int division = division(8,0);
         System.out.println("Division =
"+division);
      }catch(ArithmeticException e){
         System.out.println("Exception
aritmetica");
         //e.printStackTrace();
      }
      /*
      int division = division(8,0);
//division(8,2)
      System.out.println("Division =
"+division);*/
```

```
System.out.println("5**********
**************************");
        try{
            int division = division2(8,0);
            System.out.println("Division  =
"+division);
        }catch(ArithmeticException e){
            System.out.println("Exception
aritmetica");
            //e.printStackTrace();
        }
    }
    public static int division(int a, int b)
throws ArithmeticException{
        int c;
        /*try{
            c = a/b;
        }catch(ArithmeticException e){
            System.out.println("Exception
aritmetica");
            c=0;
        }*/
        c=a/b;
        return c;
    }
    public static int division2(int a, int b)
throws ArithmeticException{
        if(b==0){
            throw                    new
ArithmeticException();
        }int c= a/b;
        return c;
    }
}
```

- **Cajero:**

1. Cajero.java

```
package Cajero;
public class Cajero {
    public static void main(String[]
args){
        Cuenta cuenta = new Cuenta();
        cuenta.depositar(500);
        try {
            cuenta.retirar(300);
            cuenta.retirar(100);
            cuenta.retirar(200);
        }                              catch
(SaldoInsuficienteException ex) {
            System.out.println("Saldo
Insuficiente");
        }
    }
}
```

2. Cuenta.java

```
package Cajero;
public class Cuenta {
    private double saldo;
    public Cuenta(){
        this.saldo = 0;
    }
    public double getSaldo() {
        return saldo;
    }
    public       void       depositar(double
monto){
        System.out.println("Depositando:
"+monto+ "pesos");
        saldo+=monto;
    }
    public void retirar(double monto)
throws SaldoInsuficienteException{
        System.out.println("Retirando
monto");
        if(saldo<monto)
            throw                    new
SaldoInsuficienteException();
        else{
            saldo -= monto;
        }
        System.out.println("Nuevo  saldo
"+saldo+" pesos");
    }
}
```

3. SaldoInsuficienteException.java

```
package Cajero;
```

```
public class SaldoInsuficienteException extends Exception{
   SaldoInsuficienteException(){
      super("Saldo Insuficiente");
   }
}
```

- **Trabajo a entregar(POOP9 y Cajero con el usuario):**
  - A continuación solo se mostrara el código ejecutable ya que estos fueron los que sufrieron cambios para que el usuario puede poner la información con la ayuda del “import java.util.Scanner;” junto con “Scanner S = new Scanner(System.in);” que hará el trabajo de guardar la información.

1. **POOP9**

```
package poop9;
import java.util.Scanner;
public class POOP9 {
   public static void main(String[] args)
{
System.out.println("1**********************************************");
      try{
         String mensajes[] = {"Antonio", "Javier", "Gabriela"};
         for (int i = 0; i < 4; i++){
System.out.println(mensajes[i]);
         }
}catch(ArrayIndexOutOfBoundsException aie){
         System.out.println("Error: apuntador fuera de rango");
      }
System.out.println("2**********************************************");
      try{
         int n;
         int m;
         Scanner Sn = new Scanner(System.in);
         Scanner Sm = new Scanner(System.in);
         System.out.println("Pon el dividiendo:");
         n = Sn.nextInt();
         System.out.println("Pon el divisor:");
         m = Sm.nextInt();
         float equis = n/m;
         System.out.println("Equis = "+equis);
      }catch(ArithmeticException ae){
         System.out.println("Error: division entre cero");
      }finally{
         System.out.println("A pesar de todo, se ejecuta finally");
      }
      System.out.println("Fuera de try-catch");
System.out.println("3**********************************************");
      try{
         int a;
         int b;
         Scanner Sa = new Scanner(System.in);
         Scanner Sb = new Scanner(System.in);
         System.out.println("Pon el dividiendo:");
         a = Sa.nextInt();
         System.out.println("Pon el divisor:");
         b = Sb.nextInt();
         float equis = a/b;
         System.out.println("Equis = "+equis);
      }catch(Exception e){
         System.out.println("Excepcion general");
      }finally{
```

```
        System.out.println("A pesar de todo, se ejecuta finally");
    }
    System.out.println("Fuera de try-catch");


System.out.println("4**********************************************");
    //Propagancia de exceptions
    try{
        int a;
        int b;
        Scanner Sa = new Scanner(System.in);
        Scanner Sb = new Scanner(System.in);
        System.out.println("Pon el dividiendo:");
        a = Sa.nextInt();
        System.out.println("Pon el divisor:");
        b = Sb.nextInt();
        int division = division(a,b);
        System.out.println("Division = "+division);
    }catch(ArithmeticException e){
        System.out.println("Exception aritmetica");
    }
System.out.println("5**********************************************");
    try{
        int a;
        int b;
        Scanner Sa = new Scanner(System.in);
        Scanner Sb = new Scanner(System.in);
        System.out.println("Pon el dividiendo:");
        a = Sa.nextInt();
        System.out.println("Pon el divisor:");
        b = Sb.nextInt();
        int division = division2(a,b);
        System.out.println("Division = "+division);
    }catch(ArithmeticException e){
        System.out.println("Exception aritmetica");
    }
  }
  public static int division(int a, int b) throws ArithmeticException{
    int c;
    c=a/b;
    return c;
  }
  public static int division2(int a, int b) throws ArithmeticException{
    if(b==0){
        throw new ArithmeticException();
    }int c= a/b;
    return c;
  }
}
```

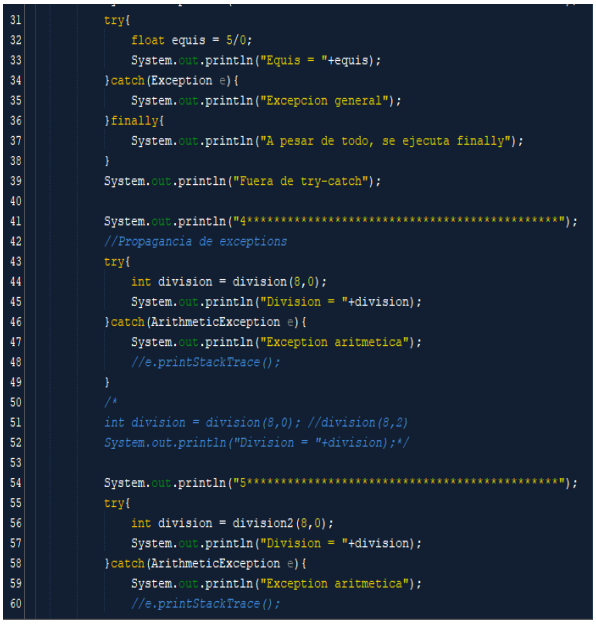
2. **Cajero**

```
package Cajero;
import java.util.Scanner; //Sirve para que el usuario ponga el monto deseado a depositar o retirar
public class Cajero {
  public static void main(String[] args){
    int a; //Monto a depositar
    int b; //Monto a retirar
    Scanner S = new Scanner(System.in);
    Scanner Sb = new Scanner(System.in);
    Cuenta cuenta = new Cuenta();
    System.out.println("Ponga el saldo a depositar: ");
    a = S.nextInt();
    cuenta.depositar(a);
    try {
        System.out.println("Ponga el saldo a retirar: ");
        b = Sb.nextInt();
```

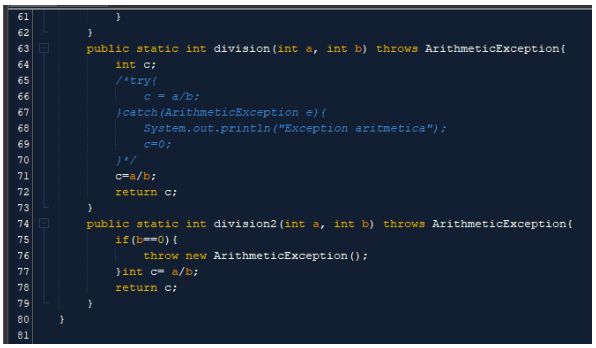
```
            cuenta.retirar(b);
        }                                   catch
(SaldoInsuficienteException ex) {
            System.out.println("Saldo
Insuficiente");
        }
    }
}
```

- **Capturas**

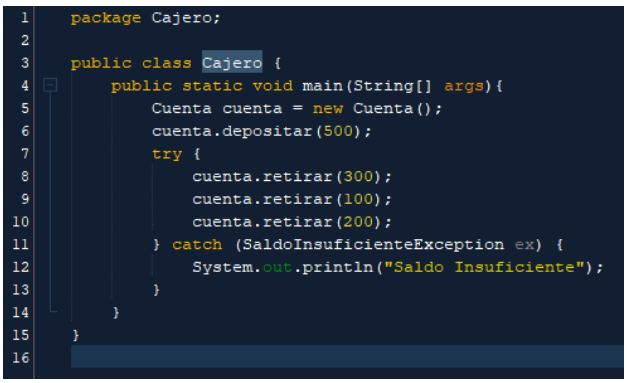
```
package poop9;

public class POOP9 {

    public static void main(String[] args) {

        System.out.println("1**********************************************");
        try{
            String mensajes[] = {"Antonio", "Javier", "Gabriela"};
            for (int i = 0; i < 4; i++){
                System.out.println(mensajes[i]);
            }
        }catch(ArrayIndexOutOfBoundsException aie){
            System.out.println("Error: apuntador fuera de rango");
        }

        System.out.println("2**********************************************");
        try{
            float equis = 5/0; //primer ejemplo: 5/2
            System.out.println("Equis = "+equis);
        }catch(ArithmeticException ae){
            System.out.println("Error: division entre cero");
        }finally{
            System.out.println("A pesar de todo, se ejecuta finally");
        }
        System.out.println("Fuera de try-catch");

        //ArithmeticException ae = lo que lance la excepcion

        System.out.println("3**********************************************");
```



```
        try{
            float equis = 5/0;
            System.out.println("Equis = "+equis);
        }catch(Exception e){
            System.out.println("Excepcion general");
        }finally{
            System.out.println("A pesar de todo, se ejecuta finally");
        }
        System.out.println("Fuera de try-catch");

        System.out.println("4**********************************************");
        //Propagancia de exceptions
        try{
            int division = division(8,0);
            System.out.println("Division = "+division);
        }catch(ArithmeticException e){
            System.out.println("Exception aritmetica");
            //e.printStackTrace();
        }
        /*
        int division = division(8,0); //division(8,2)
        System.out.println("Division = "+division);*/

        System.out.println("5**********************************************");
        try{
            int division = division2(8,0);
            System.out.println("Division = "+division);
        }catch(ArithmeticException e){
            System.out.println("Exception aritmetica");
            //e.printStackTrace();
```

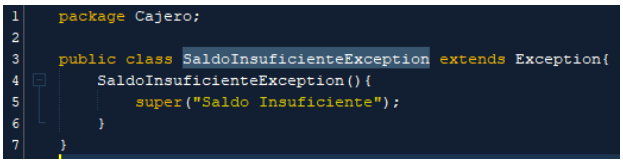


```
        }
    }
    public static int division(int a, int b) throws ArithmeticException{
        int c;
        /*try{
            c = a/b;
        }catch(ArithmeticException e){
            System.out.println("Exception aritmetica");
            c=0;
        }*/
        c=a/b;
        return c;
    }
    public static int division2(int a, int b) throws ArithmeticException{
        if(b==0){
            throw new ArithmeticException();
        }int c= a/b;
        return c;
    }
}
```



```
package Cajero;

public class Cajero {
    public static void main(String[] args){
        Cuenta cuenta = new Cuenta();
        cuenta.depositar(500);
        try {
            cuenta.retirar(300);
            cuenta.retirar(100);
            cuenta.retirar(200);
        } catch (SaldoInsuficienteException ex) {
            System.out.println("Saldo Insuficiente");
        }
    }
}
```

```
package Cajero;

public class Cuenta {
    private double saldo;

    public Cuenta(){
        this.saldo = 0;
    }

    public double getSaldo() {
        return saldo;
    }

    public void depositar(double monto){
        System.out.println("Depositando: "+monto+ "pesos");
        saldo+=monto;
    }

    public void retirar(double monto) throws SaldoInsuficienteException{
        System.out.println("Retirando monto");
        if(saldo<monto)
            throw new SaldoInsuficienteException();
        else{
            saldo -= monto;
        }
        System.out.println("Nuevo saldo "+saldo+" pesos");
    }
}
```



```
package Cajero;

public class SaldoInsuficienteException extends Exception{
    SaldoInsuficienteException(){
        super("Saldo Insuficiente");
    }
}
```

```
run:
1**********
Antonio
Javier
Gabriela
Error: apuntador fuera de rango
2**********
Error: division entre cero
A pesar de todo, se ejecuta finally
Fuera de try-catch
3**********
Excepcion general
A pesar de todo, se ejecuta finally
Fuera de try-catch
4**********
Exception aritmetica
5**********
Exception aritmetica
BUILD SUCCESSFUL (total time: 0 seconds)
```

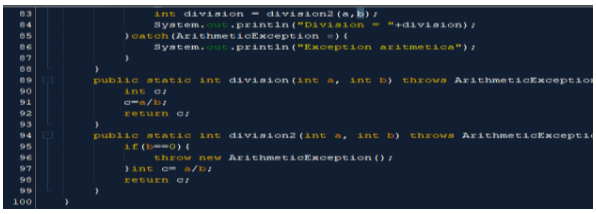
```
run:
Depositando: 500.0pesos
Retirando monto
Nuevo saldo 200.0 pesos
Retirando monto
Nuevo saldo 100.0 pesos
Retirando monto
Saldo Insuficiente
BUILD SUCCESSFUL (total time: 0 seconds)
```

```
run:
Ponga el saldo a depositar:
900
Depositando: 900.0 pesos
Ponga el saldo a retirar:
1000
Retirando monto
Saldo Insuficiente
BUILD SUCCESSFUL (total time: 5 seconds)
```

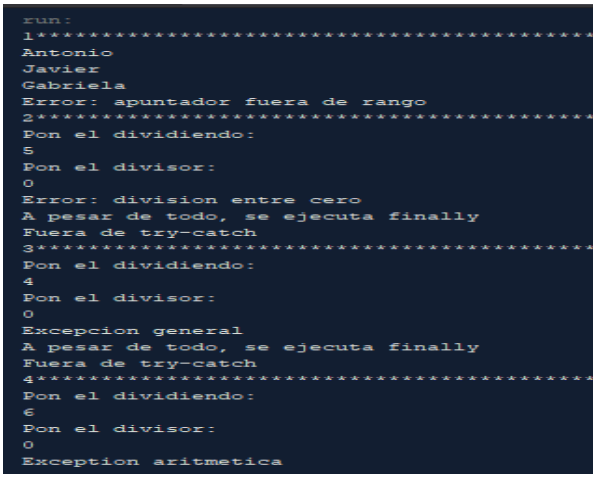
```
run:
Ponga el saldo a depositar:
900
Depositando: 900.0 pesos
Ponga el saldo a retirar:
300
Retirando monto
Nuevo saldo 600.0 pesos
BUILD SUCCESSFUL (total time: 11 seconds)
```

```java
package poop9;
import java.util.Scanner;
public class POOP9 {

    public static void main(String[] args) {

        System.out.println("1**********");

        try{
            String mensajes[] = {"Antonio", "Javier", "Gabriela"};
            for (int i = 0; i < 4; i++){
                System.out.println(mensajes[i]);
            }
        }catch(ArrayIndexOutOfBoundsException aie){
            System.out.println("Error: apuntador fuera de rango");
        }

        System.out.println("2**********");
        try{
            int n;
            int m;
            Scanner Sn = new Scanner(System.in);
            Scanner Sm = new Scanner(System.in);
            System.out.println("Pon el dividiendo:");
            n = Sn.nextInt();
            System.out.println("Pon el divisor:");
            m = Sm.nextInt();
            float equis = n/m;
            System.out.println("Equis = "+equis);
        }catch(ArithmeticException ae){
            System.out.println("Error: division entre cero");
        }finally{
            System.out.println("A pesar de todo, se ejecuta finally");
        }
        System.out.println("Fuera de try-catch");

        System.out.println("3**********");
        try{
            int a;
            int b;
            Scanner Sa = new Scanner(System.in);
            Scanner Sb = new Scanner(System.in);
            System.out.println("Pon el dividiendo:");
            a = Sa.nextInt();
            System.out.println("Pon el divisor:");
            b = Sb.nextInt();
            float equis = a/b;
            System.out.println("Equis = "+equis);
        }catch(Exception e){
            System.out.println("Excepcion general");
        }finally{
            System.out.println("A pesar de todo, se ejecuta finally");
        }
        System.out.println("Fuera de try-catch");

        System.out.println("4**********");
        //Propagancia de exceptions
        try{
            int a;
            int b;
            Scanner Sa = new Scanner(System.in);
            Scanner Sb = new Scanner(System.in);
            System.out.println("Pon el dividiendo:");
            a = Sa.nextInt();
            System.out.println("Pon el divisor:");
            b = Sb.nextInt();
            int division = division(a,b);
            System.out.println("Division = "+division);
        }catch(ArithmeticException e){
            System.out.println("Exception aritmetica");
        }

        System.out.println("5**********");
        try{
            int a;
            int b;
            Scanner Sa = new Scanner(System.in);
            Scanner Sb = new Scanner(System.in);
            System.out.println("Pon el dividiendo:");
            a = Sa.nextInt();
            System.out.println("Pon el divisor:");
            b = Sb.nextInt();
```

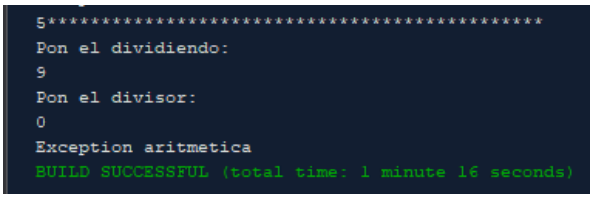
```java
package Cajero;
import java.util.Scanner; //Sirve para que el usuario ponga el monto deseado a depositar o retirar
public class Cajero {
    public static void main(String[] args){
        int a; //Monto a depositar
        int b; //Monto a retirar
        Scanner S = new Scanner(System.in);
        Scanner Sb = new Scanner(System.in);
        Cuenta cuenta = new Cuenta();
        System.out.println("Ponga el saldo a depositar: ");
        a = S.nextInt();
        cuenta.depositar(a);
        try {
            System.out.println("Ponga el saldo a retirar: ");
            b = Sb.nextInt();
            cuenta.retirar(b);
        } catch (SaldoInsuficienteException ex) {
            System.out.println("Saldo Insuficiente");
        }
    }
}
```



```
            int division = division2(a,b);
            System.out.println("Division = "+division);
        }catch(ArithmeticException e){
            System.out.println("Exception aritmetica");
        }
    }
    public static int division(int a, int b) throws ArithmeticExceptio
        int c;
        c=a/b;
        return c;
    }
    public static int division2(int a, int b) throws ArithmeticExcepti
        if(b==0){
            throw new ArithmeticException();
        }int c= a/b;
        return c;
    }
}
```



```
run:
1**********
Antonio
Javier
Gabriela
Error: apuntador fuera de rango
2**********
Pon el dividiendo:
5
Pon el divisor:
0
Error: division entre cero
A pesar de todo, se ejecuta finally
Fuera de try-catch
3**********
Pon el dividiendo:
4
Pon el divisor:
0
Excepcion general
A pesar de todo, se ejecuta finally
Fuera de try-catch
4**********
Pon el dividiendo:
6
Pon el divisor:
0
Exception aritmetica
```



```
5**********
Pon el dividiendo:
9
Pon el divisor:
0
Exception aritmetica
BUILD SUCCESSFUL (total time: 1 minute 16 seconds)
```

V. Conclusión

- José Luis Arroyo Chavarría:

Con lo visto en esta práctica me hizo a comprender y a desarrollar programas que puedan dar un aviso o una solución a un error presente en este código, me parece interesante poder hacer acciones cuando inflige un movimiento imprevisto y que en cierta parte podremos ver en distintos lugares como el ejemplo realizado de los cajeros o hasta nuestra computadora cuando queremos instalar programas pero no tenemos suficiente espacio en nuestra pc. Al final esta práctica y sus ejercicios nos ayudara y nos servirá para nuestros futuros trabajos.

- Francisco Moisés Barrera Guardia:

Esta práctica me hizo entender que a pesar de los fallos que el programador tenga dentro de la sintaxis de un programa, la maquina también comete errores los cuales se pueden arreglar y se pueden ver de otra manera

- Juan Manuel Peralta Rodríguez

Gracias al desarrollo de esta práctica pudimos entender de una mejor manera el tema tratado, cumpliendo de esta forma los objetivos planteados al inicio de la misma, por otro lado, durante el desarrollo se aprendieron cosas nuevas las cuales nos ayudarán para el desarrollo de nuevos proyectos que se lleguen a plantear a futuro.

- Rodrigo Daniel Reséndiz Cruz

En esta práctica aprendí nueva forma de evitar que un programa se detenga a causa de un error y mejorar la experiencia del usuario al señalarle que hay un error en una operación, considero que las funcionalidades

aprendidas nos ayudarán a hacer programas con un mayor nivel de calidad.

VI. Referencias

- Java Exceptions (Try...Catch). (1999). w3schools. https://www.w3schools.com/java/java_try_catch.asp
- ERRORES COMUNES EN EL PROCESO DE PROGRAMACION. (2010). frip. http://www.frlp.utn.edu.ar/materias/algoritmos/errores2010.pdf