Práctica Nro. 9
Excepciones

Objetivo: Conocer e interpretar los distintos modelos de excepciones que implementan los lenguajes de programación.

Ejercicio 1: ¿Explique claramente a qué se denomina excepción?
Ejercicio 2: ¿Qué debería proveer un lenguaje para el manejo de las excepciones? ¿Todos los lenguajes lo proveen?
Ejercicio 3: ¿Qué ocurre cuando un lenguaje no provee manejo de excepciones? ¿Se podría simular? Explique cómo lo haría
Ejercicio 4: Cuando se termina de manejar la excepción, la acción que se toma luego es importante. Indique

01. ¿Qué modelos diferentes existen en este aspecto?
02. Dé ejemplos de lenguajes que utilizan cada uno de los modelos presentados anteriormente. Por cada uno responda respecto de la forma en que trabaja las excepciones.
    a. ¿Cómo se define?
    b. ¿Cómo se lanza?
    c. ¿Cómo se maneja?
    d. ¿Cuál es su criterio de continuación?
03. ¿Cuál de esos modelos es más inseguro y por qué?

Ejercicio 5: La propagación de los errores, cuando no se encuentra ningún manejador asociado, no se implementa igual en todos los lenguajes. Realice la comparación entre el modelo de Java, Python y PL/1, respecto a este tema. Defina la forma en que se implementa en un lenguaje conocido por Ud.
Ejercicio 6: Sea el siguiente programa escrito en Pascal

```
...
Procedure Manejador;
  Begin ...   end;
Procedure P(X:Proc);
  begin
    ....
    if Error then X;
    ....
  end;
Procedure A;
  begin
          ....
          P(Manejador);
          ….
  end;
....
```

¿Qué modelo de manejo de excepciones está simulando? ¿Qué necesitaría el programa para que encuadre con los lenguajes que no utilizan este modelo? Justifique la respuesta.

Ejercicio 7: Sea el siguiente programa escrito en Pascal:

```
Program Principal;
var x:int;  b1,b2:boolean;
 Procedure P (b1:boolean);
      var x:int;
      Procedure Manejador1
                        begin
                  x:=x + 1;
                end;
    begin
    x:=1;
    if b1=true then Manejador1;
    x:=x+4;
    end;
    Procedure Manejador2;
      begin
            x:=x * 100;
    end;
```

```
Begin
      x:=4;
      b2:=true;
      b1:=false;
      if b1=false then Manejador2;
      P(b);
      write (x);
End.
```

a) Implemente este ejercicio en PL/1 utilizando manejo de excepciones
b) ¿Podría implementarlo en JAVA utilizando manejo de excepciones? En caso afirmativo, realícelo.

Ejercicio 8: Sean los siguientes, procedimientos de un programa escrito en JAVA:

```
public static void main (String[] argos){
      Double array_doubles[]= new double[500];
      for (int i=0; i<500; i++){
            array_doubles[i]=7*i;
       }
      for (int i=0 ; i<600 ; i=i+25){
            try{
                  system.out.println("El elemento en "+ i + " es " + acceso_por_indice (array_doubles,i));
            }
            catch(ArrayIndexOutOfBoundsException e){
                  system.out.println(e.tostring());
            }
            catch(Exception a){
                  system.out.println(a.tostring());
            }
            finally{
                  system.out.println("sentencia finally");
            }
      }
}

Public static double acceso_por_indice (double [] v, int indice) throws Exception; ArrayIndexOutOfBoundsException{
          if ((indice>=0) && (indice<v.length)){
                Return v[indice];
          }
          else{
                if (indice<0){
                  // caso excepcional
                  Throw new ArrayIndexOutOfBoundsException(" el índice" + indice + " es un número negativo");
```

```
            }
            else{
              // caso excepcional
              Throw new Exception(" el indice" + indice + " no es una posición válida");
              }
        }
}
```

a) Analizar el ejemplo y decir qué manejadores ejecuta y en qué valores quedan las variables. JUSTIFIQUE LA RESPUESTA.
b) La excepción se propaga o se maneja en el mismo método? ¿Qué instrucción se agrega para poder propagarla y que lleve información?.
c) como modificaría el método "acceso_por_indice" para que maneje él mismo la excepción.

Ejercicio 9: Indique diferencias y similitudes entre Phyton y Java con respecto al manejo de excepciones.
Ejercicio 10: ¿Qué modelo de excepciones implementa Ruby?. ¿Qué instrucciones específicas provee el lenguaje para manejo de excepciones y cómo se comportan cada una de ellas?
Ejercicio 11: Indique el mecanismo de excepciones de javascript.
Ejercicio 12: Sea el siguiente pro**grama escrit**o en PYTHON::Indique el camino de ejecución**.**

```
#!/usr/bin/env python
#calc.py

def dividir():
   x = a / b
   print (("Resultado"), (x))

while True:
   try:
      a = int(input("Ingresa el primer numero: \n"))
      b = int(input("Ingresa el segundo numero: \n"))
      dividir()
      break
  except ZeroDivisionError:
      print ("No se permite dividir por cero")

  finally:
      print ("Vuelve a probar")
```

a) Describa qué caminos ejecuta para diferentes valores de ingreso
b) Agregar el uso de una excepción anónima

Ejercicio 13: Sea el siguiente código escrito en JAVA

```
public class ExcepcionUno extends Exception {

        public ExcepcionUno(){
                super(); // constructor por defecto de Exception
         }
```

```
	public ExcepcionUno( String cadena ){
		super( cadena ); // constructor param. de Exception
	}

}

public class ExcepcionDos extends Exception {

	public ExcepcionDos(){
		super(); // constructor por defecto de Exception
	 }

	public ExcepcionDos( String cadena ){
		super( cadena ); // constructor param. de Exception
	}

}
public class ExcepcionTres extends Exception {

	public ExcepcionTres(){
		super(); // constructor por defecto de Exception
	 }

	public ExcepcionTres( String cadena ){
		super( cadena ); // constructor param. de Exception
	}

}


public class Lanzadora {

	public void lanzaSiNegativo(int param) throws ExcepcionUno {
		if (param < 0)
			throw new ExcepcionUno("Numero negativo");
	}

	public void lanzaSimayor100(int param) throws ExcepcionDos {
		if (param >100 and param<125)
			throw new ExcepcionDos("Numero mayor100");
	}

	public void lanzaSimayor125(int param) throws ExcepcionTres {
		if (param >= 125)
			throw new ExcepcionTres("Numero mayor125");
	}

}




import java.io.FileInputStream;
import java.io.IOException;

public class Excepciones {

	public static void main(String[] args) {
		// Para leer un fichero
```

```
                Lanzadora lanza = new Lanzadora();
                FileInputStream entrada = null;
                int leo;
                try {
                        entrada = new FileInputStream("fich.txt");
                        while ((leo = entrada.read()) != -1){
                                if (leo < 0)
                                        lanza.lanzaSiNegativo(leo);
                                else if (leo > 100)
                                        lanza.lanzaSimayor100(leo);
                        }

                        entrada.close();
                        System.out.println("Todo fue bien");
                }
                catch (ExcepcionUno e) { // Personalizada
                        System.out.println("Excepcion: " + e.getMessage());
                }
                catch (ExcepcionDos e) { // Personalizada
                        System.out.println("Excepcion: " + e.getMessage());
                }
                catch (IOException e) { // Estándar
                        System.out.println("Excepcion: " + e.getMessage());
                }
                finally {
                        if (entrada != null)
                                try {
                                        entrada.close(); // Siempre queda cerrado
                                }
                                catch (Exception e) {
                                        System.out.println("Excepcion: " + e.getMessage());
                                }
                        System.out.println("Fichero cerrado.");
                }
        }
}
```

a) Indique cómo se ejecuta el código. Debe quedar en claro los caminos posibles de ejecución, cuales son los manejadores que se ejecutan y cómo se buscan los mismos y si en algún caso se produce algún error.

Ejercicio 14. Dado el siguiente código en Java. Indique todos los posibles caminos de resolución, de acuerdo a los números que vaya leyendo del archivo.

```
class ExcepcionE1 extends Exception {

        public  ExcepcionE1(){
                super(); // constructor por defecto de Exception
        }

        public  ExcepcionE1( String cadena ){
                super( cadena ); // constructor param. de Exception
        }

 }
```

```
class ExcepcionE2 extends Exception {

      Public ExcepcionE2(){
            super(); // constructor por defecto de Exception
      }

      Public ExcepcionE2( String cadena ){
            super( cadena ); // constructor param. de Exception
      }

}

// Esta clase lanzará la excepción
public class  Evaluacion {
            void Evalua( int edad ) throws ExcepcionE1, ExcepcionE2 {
                  if ( edad < 18 )
                        throw new ExcepcionE1( "Es una persona menor de edad" );
                  else  if ( edad> 70 )
                        throw new ExcepcionE2( "Es persona mayor de edad" );

            }

            void Segmenta( int edad ) throws ExcepcionE1, ExcepcionE2 {
                  if ( edad  < 35 )
                        throw new ExcepcionE1( "Es una persona joven" );

            }

}

class AnalisisEdadPoblacion{
      public static void main( String[] args ){
            // Para leer un fichero
            Evaluacion Invoca = new Evaluacion();
            FileInputSt   ream entrada = null;
            int leo;
             try{
            entrada = new FileInputStream( "fich.txt" );
            while ( ( leo = entrada.read() ) != -1 )  {
                try {
                     if (leo<0) {
                         throw new ExcepcionE1( "Edad inválida" );
                     }
                     else{
                           if (leo>120){
                             throw new ExcepcionE1( "Edad inválida" );
                           }
                      }
                     invoca.evalua (leo);
                    invoca.segmenta( leo );
                     System.out.println( " ( "Es persona adulta, Todo fue bien" );
                }
                catch ( ExcepcionE2 e ){
                     System.out.println( "Excepcion: " + e.getMessage() );
```

```
                }
                catch ( ExcepcionE1 e ){
                        System.out.println( "Excepcion: " + e.getMessage() );
                }
            } catch (FileNotFoundException e1) {
                    System.out.println("No se encontró el archivo");
            } catch (IOException e) {
                    System.out.println("Problema para leer los datos");
            }finally {
                    if (entrada != null)
                            try {
                                    entrada.close();
                            } catch (Exception e) {
                                    System.out.println("Excepcion: " + e.getMessage());
                            }
                    System.out.println("Fichero cerrado.");
            }
        }
}
```