**TP - Excepciones**

**Maximo Aguilera**

**Parte A**

a. En Java, la clase que define las excepciones se llama Exception. Esta clase es la clase base para todas las excepciones en Java.

b. La clase en Java que representa las excepciones que se producen al invocar un método en un objeto nulo se llama NullPointerException. Esta excepción se lanza cuando se intenta acceder o invocar un método en un objeto que es null.

c. El método printStackTrace() en Java se utiliza para imprimir la pila de llamadas al método que generó la excepción, junto con la descripción de la excepción y el lugar en el código donde ocurrió. Esta información es útil para depurar el código y encontrar la causa raíz de la excepción. El método printStackTrace() se llama generalmente en el bloque catch de un manejo de excepciones para mostrar la traza de la excepción.

**Ejercicio 1:**

try {

int resultado = 10 / 0; // Esto generará una ArithmeticException

} catch (ArithmeticException e) {

System.out.println("Se produjo una excepción: " + e.getMessage());

}

**Ejercicio 2:**

String aux = "hola";

try {

int aux2 = Integer.parseInt(aux);

System.out.println("Número convertido: " + aux2);

} catch (NumberFormatException e) {

System.out.println("Error al convertir la cadena en número: " + e.getMessage());

}

**Ejercicio 3:**

try {

FileInputStream fis = new FileInputStream("archivo\_que\_no\_existe.txt");

} catch (IOException e) {

System.out.println("Se produjo una IOException: " + e.getMessage());

}

**Ejercicio 4:**

public class Main {

public static void main(String[] args) {

try {

metodoExterno();

} catch (Exception e) {

System.out.println("Se produjo una excepción: " + e.getMessage());

}

}

public static void metodoExterno() throws Exception {

try {

metodoInterno1();

} catch (NullPointerException e) {

System.out.println("Excepción NullPointerException capturada en metodoExterno()");

throw new Exception("Excepción lanzada desde metodoExterno()");

}

}

public static void metodoInterno1() {

int[] array = null;

try {

array[0] = 1;

} catch (ArrayIndexOutOfBoundsException e) {

System.out.println("Excepción ArrayIndexOutOfBoundsException capturada en metodoInterno1()");

metodoInterno2();

}

}

public static void metodoInterno2() {

try {

int resultado = 10 / 0;

} catch (ArithmeticException e) {

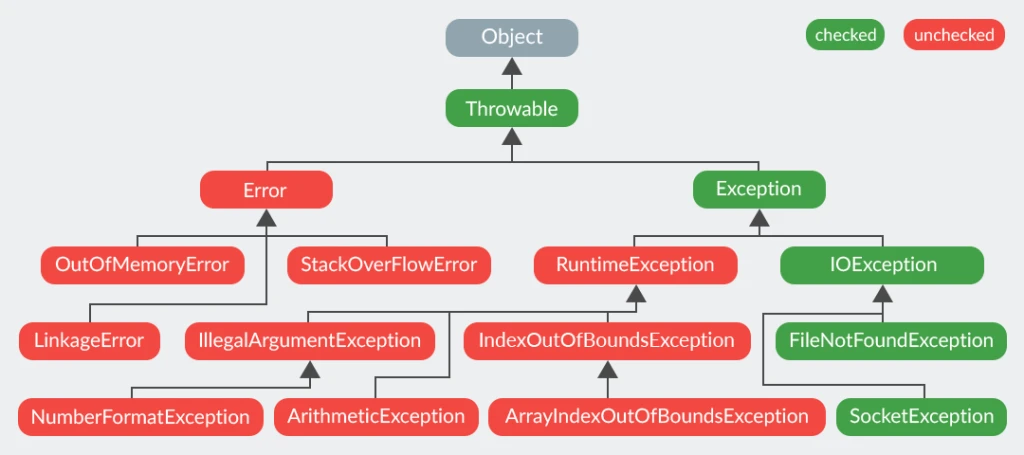
System.out.println("Excepción ArithmeticException capturada en metodoInterno2()");

}

}

}

**Ejercicio 5:**



* NullPointerException: Se produce cuando se intenta acceder a un miembro de un objeto que es null.
* ArrayIndexOutOfBoundsException: Se produce cuando se intenta acceder a un índice inválido en un array.
* ArithmeticException: Se produce cuando ocurre una condición aritmética excepcional, como la división por cero.
* IOException: Se produce cuando ocurre un error de entrada/salida durante la lectura o escritura de datos.

**Ejercicio 6:**

class MiExcepcion extends Exception {

public MiExcepcion(String mensaje) {

super(mensaje);

}

}

class MiOtraExcepcion extends MiExcepcion {

public MiOtraExcepcion(String mensaje) {

super(mensaje);

}

}

public class Main {

public static void main(String[] args) {

try {

metodoQueLanzaMiExcepcion();

} catch (MiExcepcion e) {

System.out.println("Se produjo una MiExcepcion: " + e.getMessage());

}

}

public static void metodoQueLanzaMiExcepcion() throws MiExcepcion {

throw new MiOtraExcepcion("¡Esta es MiOtraExcepcion!");

}

}

**Ejercicio 7:**

El bloque finally siempre se ejecuta, independientemente de si se lanzó una excepción o no. En este caso, el bloque finally siempre devuelve 3.

**Parte B**

**Ejercicio 1:**

try {

int resultado = 10/ 0;

} catch (ArithmeticException e) {

System.out.println("Se produjo una ArithmeticException: " + e.getMessage());

}

**Ejercicio 2:**

El ejercicio 2 está comprimido

**Ejercicio 3:**

import java.util.Random;

public class ParImparException extends Exception {

public ParImparException(String mensaje) {

super(mensaje);

}

}

public class Main {

public static void main(String[] args) {

Random random = new Random();

int numero = random.nextInt(100); // Genera un número aleatorio entre 0 y 99

try {

if (numero % 2 == 0) {

throw new ParImparException("El número generado es par: " + numero);

} else {

throw new ParImparException("El número generado es impar: " + numero);

}

} catch (ParImparException e) {

System.out.println(e.getMessage());

}

}

}

**Ejercicio 4:**

public class Main {

public static void main(String[] args) {

try {

metodo1();

} catch (Exception e) {

System.out.println("Excepción manejada en el método 4: " + e.getMessage());

}

}

public static void metodo1() throws Exception {

metodo2();

}

public static void metodo2() throws Exception {

metodo3();

}

public static void metodo3() throws Exception {

metodo4();

}

public static void metodo4() throws Exception {

throw new Exception("Excepción lanzada en el método 4");

}

}