Trabajo Práctico Excepciones

Ejercicio 1

a. La clase base para todas las excepciones en Java es java.lang.Throwable.

b. La excepción que se produce al invocar un método en un objeto nulo es java.lang.NullPointerException.

c. El método printStackTrace() muestra la traza de la pila de llamadas que llevó a la excepción, incluyendo: El tipo de excepción El mensaje de error La secuencia de llamadas a métodos que produjo la excepción Los números de línea donde ocurrió

Ejercicio 2

Para obtener el mensaje de una excepción se usa el método getMessage().

Ejemplo de código:

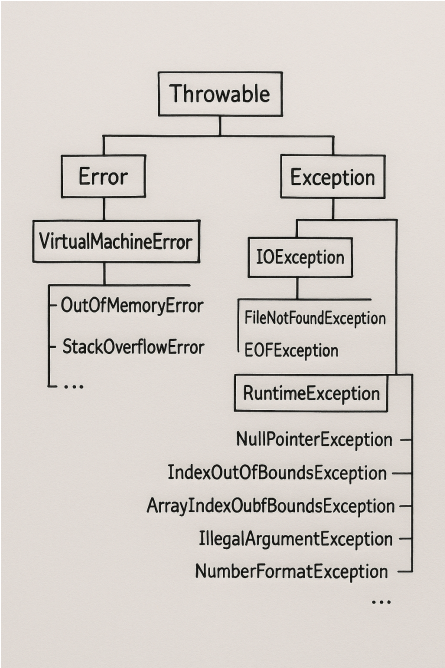
```
try {
   int[] arr = new int[5];
   System.out.println(arr[10]); // Esto lanzará ArrayIndexOutOfBoundsException
} catch (ArrayIndexOutOfBoundsException e) {
   System.out.println("Error: " + e.getMessage());
   // Salida: Error: Index 10 out of bounds for length 5
}
```

Ejercicio 3

El fragmento de código intenta convertir la cadena "hola" a un número entero, lo que lanzará una NumberFormatException. Versión con manejo de excepciones:

```
try {
   String aux = "hola";
   int aux2 = Integer.parseInt(aux);
} catch (NumberFormatException e) {
   System.out.println("Error al convertir el número: " + e.getMessage());
}
```

Ejercicio 6



Documentación de 4 excepciones:

NullPointerException: Ocurre cuando se intenta acceder a un método o propiedad de un objeto que es null.

ArrayIndexOutOfBoundsException: Se produce al intentar acceder a un índice de array que no existe (negativo o mayor que el tamaño).

FileNotFoundException: Lanzada cuando un intento de abrir un archivo falla porque el archivo no existe.

NumberFormatException: Ocurre cuando se intenta convertir una cadena a un número, pero la cadena no tiene el formato adecuado.

Ejercicio 8

El programa muestra el siguiente comportamiento:

Cuando se llama a devuelveNumero(1): El número 1 es impar (1 % 2 == 1), por lo que NO se lanza la excepción

El bloque try intenta retornar el valor 1 Sin embargo, el bloque finally siempre se ejecuta, incluso si hay un return en el try o catch

El finally sobrescribe cualquier return anterior con su propio return 3

Ejercicio 9

Cree un programa que lance una ArithmeticException.

```
public class DivisionPorCero {
  public static void main(String[] args) {
    try {
      int resultado = dividir(10, 0);
```

```
            System.out.println("Resultado: " + resultado);
        } catch (ArithmeticException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }

    public static int dividir(int a, int b) {
        return a / b; // Esto lanzará ArithmeticException si b es 0
    }
}
```