

Instrucciones Java del ejercicio

Tabla 6.4. Instrucciones Java del ejercicio 6.4.

Instrucción	Descripción	Formato
<i>try</i>	Bloque de código en el que se intentará capturar una excepción si se produce y captura, se establece qué hacer con ella.	<i>try</i> { bloque donde puede ocurrir una excepción }
<i>catch</i>	Define un conjunto de instrucciones para tratar la excepción generada en el bloque <i>try</i> anterior.	<i>catch</i> (excepción e) { bloque donde se trata el problema }
<i>finally</i>	Bloque donde se pueden definir instrucciones necesarias tanto si hay o no excepciones (se ejecuta siempre).	<i>finally</i> { bloque que se ejecuta siempre }
<i>Exception</i>	Indica condiciones que una aplicación podría querer capturar y gestionar.	<i>Exception</i> e;
<i>ArithmeticException</i>	Esta excepción se lanza cuando ha ocurrido una condición aritmética excepcional. Por ejemplo, una división por cero.	<i>ArithmeticException</i> e;

Enunciado: clase PruebaExcepciones

¿Cuál es el resultado de la ejecución del método *main* del siguiente programa? Determinar qué se imprime en pantalla.

Solución

Clase: PruebaExcepciones

```
package Excepciones;

/**
 * Esta clase denominada PruebaExcepciones lanza diferentes
 * excepciones en situaciones específicas del programa. Los mensajes
 * que se muestran en pantalla ayudan a identificar la porción de código
 * que se ejecutó o no.
 * @version 1.2/2020
 */
public class PruebaExcepciones {

    /**
```

```
* Método main con dos bloques try que generan excepciones que
* deben ser gestionadas
* @throws ArithmeticException Excepción aritmética de división
* por cero
* @throws Exception Excepción general
*/
public static void main(String args[]) {
    // Primer bloque try
    try {
        System.out.println("Ingresando al primer try");
        double cociente = 10000/0; // Se lanza una excepción
        System.out.println("Después de la división"); /* Esta
            instrucción nunca será ejecutada */
    } catch (ArithmeticException e) { // Se captura la excepción
        System.out.println("División por cero"); /* Se imprime en
            pantalla este mensaje */
    } finally {
        /* La sentencia finally siempre se ejecuta, ocurra o no una
            excepción */
        System.out.println("Ingresando al primer finally");
    }

    // Segundo bloque try
    try {
        System.out.println("Ingresando al segundo try");
        Object objeto = null;
        objeto.toString(); // Se lanza una excepción
        /* Esta instrucción nunca será ejecutada porque se lanzó una
            excepción */
        System.out.println("Imprimiendo objeto");
    } catch (ArithmeticException e) { /* La excepción lanzada no es
        de este tipo */
        System.out.println("División por cero");
    } catch (Exception e) { // Se captura la excepción
        System.out.println("Ocurrió una excepción"); /* Se imprime
            en pantalla este mensaje */
    } finally {
        /* La sentencia finally siempre se ejecuta, ocurra o no una
            excepción */
        System.out.println("Ingresando al segundo finally");
    }
}
}
```

Diagrama de clases

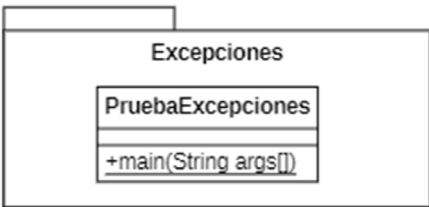


Figura 6.9. Diagrama de clases del ejercicio 6.4.

Explicación del diagrama de clases

Se ha definido un paquete denominado “Excepciones” con una única clase: *PruebaExcepciones*. Esta no tiene atributos, solamente contiene un método *main* con sus argumentos. El código del método *main* establecerá dos bloques *try* para la captura y tratamiento de posibles excepciones, que se pueden generar durante la ejecución del programa.

Diagrama de máquinas de estado

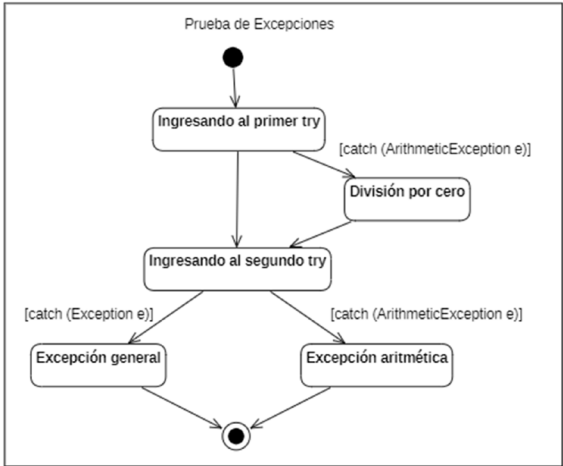


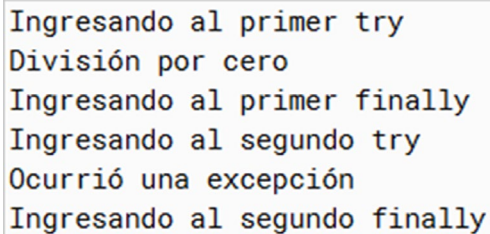
Figura 6.10. Diagrama de estados del ejercicio 6.4.

Explicación del diagrama de máquina de estado

Se ha agregado este tipo de diagrama UML, pero el diagrama de clases no expresa la lógica interna del método *main* donde se lanzan y gestionan posibles excepciones que ocurren durante la ejecución del programa. Un diagrama de máquina de estados permite entender los diferentes estados por los que pasa un programa a partir de los bloques *try*, que se están ejecutando y las posibles excepciones que pueden ocurrir.

El programa inicia pasando al primer estado denominado “Ingresando al primer *try*”. Si se lanza y captura una excepción aritmética, el programa pasa al estado “División por cero”. Así, si en el primer estado no ocurre una excepción o ya pasó por el estado de división por cero, el programa pasa inmediatamente al estado “Ingresando al segundo *try*”. De igual manera que en el primer estado, si se lanza y captura una excepción (en este caso, excepción aritmética), el programa pasa al estado “Excepción aritmética”. Si la excepción es general, el programa pasa al estado “Excepción general”.

Ejecución del programa



```
Ingresando al primer try
División por cero
Ingresando al primer finally
Ingresando al segundo try
Ocurrió una excepción
Ingresando al segundo finally
```

Figura 6.11. Ejecución del programa del ejercicio 6.4.

Ejercicios propuestos

- ¿Cuál es el resultado de la ejecución del siguiente código?

```
class ExcepciónFueraLímite {
    public static void main(String args[]) {
        try {
            String texto = "Programación";
            char caracter = texto.charAt(14);
            System.out.println(caracter);
        }
    }
}
```