• Descripción de distintos tipos de errores que pueden ocurrir y cómo se manejan en Java

\*Errores de sintaxis:

Al compilar un archivo puede ocurrir un error estos errores son de codificación o sintaxis, como:

−Punto y coma faltante

−Errores de ortografía

−Asignación de un valor a una variable que no sea el tipo

correcto

Es común realizar varias etapas de arreglo de errores de sintaxis antes de compilar el archivo de manera exitosa

El olvido de un punto y coma al final de una sentencia Java se considera un error de sintaxis

Exception in thread "main" java.lang.Error: Unresolved compilation

problem:

Syntax error, insert ";" to complete Statement

at ErrorIndicators.main(ErrorIndicators.java:8)

El uso de = en lugar de == para comparar valores en una condición "si" es un error de sintaxis 6

if(x=y){

…

}

if(x==y){

…

}

Un error ortográfico en el nombre de una variable o un método se considera un error de sintaxis asegúrese de verificar que ha escrito la variable o el método de la misma manera que lo ha declarado para arreglar este error, verifique la ortografía de:

−La variable o el método donde ha sido declarado

−Donde usted invoca la variable o el método

Exception in thread "main" java.lang.Error:

Unresolved compilation problem:

variableName cannot be resolved to a variable

at MisspellingVariables.main(MisspellingVariables.java:7)

\*Errores de lógica

Los errores de lógica ocurren como resultado de la lógica incorrecta de un programador estos errores no producen un error de compilación o de tiempo de ejecución por ejemplo, un ciclo se ejecuta demasiada cantidad de veces o el programa produce un resultado incorrecto

Colocación de un punto y coma después de una condición "si" o inicialización de un ciclo:

−Los intérpretes leen el punto y coma como el final del ciclo, lo que significa que todo lo que va después del punto y coma será tratado como si estuviera fuera del ciclo

for(int i = 0; i < 5; i++);

System.out.println(i);

El uso de == para comparar Strings u objetos es un error de lógica, a menos que usted realice verificaciones para ver si se refieren al mismo objeto

String s1 = "Hello";

String s2 = "Goodbye";

if(s1==s2)

System.out.println("They are equal");

por este punto y coma, siempre se visualizará "Hello", aunque x sea menor que 10

if(x > 10);

System.out.println("Hello");

• Descripción de excepciones utilizadas en Java

Una vez que un archivo se compila de manera exitosa, puede ocurrir un error al probar el archivo durante el tiempo de ejecución estos errores de tiempo de ejecución se denominan excepciones y deberán ser manejados por el programador que usa el código en el programa, se los conoce también como “throws” (lanzar) la excepción

• Determinación de las excepciones que se arrojan para una clase base

Creación de un bloque “Try/Catch”

Un bloque “try/catch” puede manejar excepciones controladas y no controladas

A continuación, podrá ver un ejemplo de un bloque “try/catch”

try{

...code that might cause an exception

}

catch (exception e){

...code to handle the exception

}

• Escribir un código para manejar una excepción arrojada mediante el método de una clase base

public class MyClass extends BaseClass {

public void myMethod() {

try {

someMethod(); // llamamos al método que puede arrojar una excepción

} catch (Exception e) { // capturamos la excepción

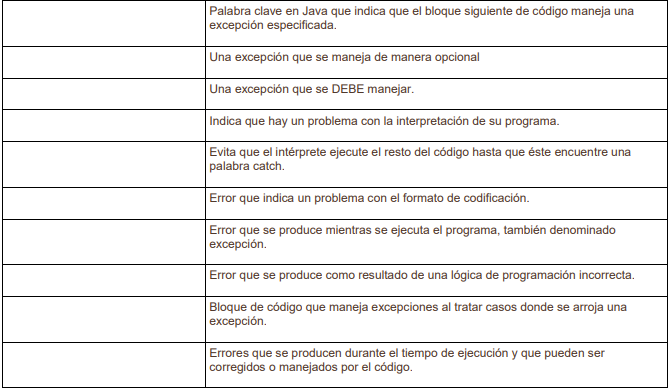
// manejamos la excepción de acuerdo a nuestras necesidades

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

}

}

}



excepciones

try-catch

throws

MalformedInputException

Error de logica

Errores de sintaxis

excepciones controladas

excepciones no controladas

try

catch

1. Describa la diferencia entre error de sintaxis, error lógico y excepción.

\*Error de sintaxis:

Ocurre cuando el código no sigue las reglas gramaticales y de estructura del lenguaje de programación. es un error en la escritura del código, suele ser detectado por el compilador y no permite que el programa se ejecute correctamente, algunos ejemplos de errores de sintaxis incluyen falta de paréntesis, llaves, puntos y comas, errores en la declaración de variables, etc.

\*Error lógico:

Ocurre cuando el código sigue la sintaxis correcta, pero produce resultados inesperados debido a un error en la lógica del programa, es un error en el diseño o la estructura del programa lo que puede ser más difícil de detectar porque no producen errores de compilación ni excepciones, pero puede causar resultados inesperados o incoherentes en la salida del programa, algunos ejemplos de errores lógicos incluyen comparaciones erróneas, lógica incorrecta en las declaraciones condicionales, errores en las fórmulas matemáticas, etc.

\*Excepción:

Ocurre cuando el programa encuentra una situación inesperada o de error en tiempo de ejecución, como una división por cero, una operación de entrada/salida fallida, una referencia nula, un índice fuera de rango en un arreglo, etc. Es una señal que el programa envía para informar que algo salió mal y no puede continuar su ejecución normal. Las excepciones pueden ser manejadas por el programa mediante el uso de bloques try-catch, que permiten tomar medidas para solucionar el problema y continuar con la ejecución del programa en lugar de detenerse abruptamente.

2. ¿Cuál es el problema del siguiente código? Debe imprimir "Hello World" cuatro veces a la pantalla.

String str = “Hello World”;

for(int i = 0; i < 4; i++);

{

System.out.println(str);

str+= ”!”;

}

Para corregir este problema, simplemente elimine el punto y coma (;) después del encabezado del bucle for de la siguiente manera:

String str = "Hello World";

for(int i = 0; i < 4; i++) {

System.out.println(str);

str += "!";

}

3. Describa una excepción que haya experimentado antes en su programa. Explique de qué manera se podría manejar con un bloque de código try/catch.

NullPointerException. Esta excepción ocurre cuando se intenta acceder a una variable o método de un objeto que es nulo (no se ha inicializado) y, por lo tanto, no tiene un valor asignado.

Para manejar esta excepción en un bloque de código try-catch, podríamos envolver cualquier línea de código que pueda arrojar una NullPointerException dentro de un bloque try y capturar la excepción en un bloque catch.

String myString = null;

try {

System.out.println(myString.length());

} catch (NullPointerException e) {

System.out.println("La variable myString es nula y no se puede acceder a su longitud.");

}

5. ¿Cuál es la diferencia entre una excepción marcada y una excepción sin marcar?

La diferencia principal entre ambas es que las excepciones marcadas son comprobadas por el compilador en tiempo de compilación, mientras que las excepciones sin marcar no lo son.

Las excepciones marcadas son excepciones que se derivan de la clase Exception (o de sus subclases directas) y que deben ser capturadas (manejadas) por el programa en tiempo de compilación. Esto significa que, si un método lanza una excepción marcada, el compilador obliga al programador a manejar esa excepción, ya sea mediante el uso de una cláusula try-catch o mediante el uso de la cláusula throws para propagar la excepción al método que lo invocó. Ejemplos de excepciones marcadas incluyen IOException, SQLException, ClassNotFoundException, etc.

Por otro lado, las excepciones sin marcar son excepciones que se derivan de la clase RuntimeException (o de sus subclases directas) y que no necesitan ser capturadas por el programa en tiempo de compilación. Esto significa que el compilador no obliga al programador a manejar estas excepciones y, por lo tanto, pueden ser ignoradas por completo si se desea. Ejemplos de excepciones sin marcar incluyen NullPointerException, ArithmeticException, IndexOutOfBoundsException, etc.