élogos

EXCEPCIONES JAVA

élogos

ÍNDICE

EXCEPCIONES JAVA

1. Clasificación de las excepciones 3

2. Lanzamiento de excepciones. Cláusula throw 5

1. Clasificación de las excepciones

Tipos de errores en Java

Una aplicación Java puede capturar las excepciones a fin de realizar algún tratamiento de la situación en vez de que se detenga el programa.

Uso de excepciones

Una vez diseñado el programa se definen el manejo de casos en que se produzca un error en tiempo de ejecución para ello se tendrá que capturar la excepción.

```
public class MiPrimeraClaseExcepcional {
public void unMetodoExcepional() throws MiPrimeraExcepcion {
    //Código donde se puede producir MiPrimeraExcepción
}
…
}
```

Aquí indicamos que este método puede lanzar (producir) una excepción, que además se llama MiPrimeraExcepcion.

Excepciones y errores

Throwable es la superclase de todas clases de excepción, las cuales heredan a su de Exception.

Excepciones del Sistema

1. ArithmeticException:

Se produce cuando sucede un error de tipo aritmético, por ejemplo, cuando tratamos de dividir un número por cero, el programa no entiende lo que es y se para.

2. NullPointerException:

Se produce cuando tratamos de acceder a una propiedad de un objeto que no existe todavía para el programa.

Ejemplo:

```
String cadena=null:
int longitud=cadena.length();  //Error porque el objeto no existe aún.
```

3. ClasCastException:
Se produce cuando no hacemos correctamente un casting entre clases.

4. ArrayIndexOutOfBoundsException:
Se produce cuando accedemos a una posición de un array superior al número de elementos del array.

5. NegativeSizeArrayIndexException:
Se produce cuando accedemos a una posición de un array inferior al elemento cero del array, es decir, a una posición negativa del array.

Uso práctico de excepciones 1

Recordemos el método que definimos anteriormente denominado unMetodoExcepcional():

Recordar método

```
public class MiPrimeraClaseExcepcional {
public void unMetodoExcepional() throws MiPrimeraExcepcion {
                ...
               }
               ...
               }
```

Llamar método

```
public void otroMetodoExcepcional() throws MiPrimeraExcection {
MiPrimeraClaseExcepcional mPCE = new MiPrimeraClaseExcepcional();
mPCE.unMetodoExcepcional();
               }
```

Si no tratamos aquí la excepción MiPrimeraExcepcion, al menos hemos de lanzarla (indicar) a los llamadores de nuestro método otroMetodoExcepcional() (lo hacemos mediante la sentencia throws) y hemos de hacerlo así porque estamos llamando al método unMetodoExcepcional() que puede producir dicha excepción.

Uso práctico de excepciones 2

A continuación introduciremos como se captura una excepción en un programa. Si quisiéramos nosotros mismos manejar la excepción lo haríamos de la siguiente forma :

```
public void metodoResponsable() {
MiPrimeraClaseExcepcional mPCE = new MiPrimeraClaseExcepcional();
 try {
mPCE.unMetodoExcepcional();
}
catch (MiprimeraExcepcion m) {
// aquí haríamos las acciones pertinentes
// si se produjera la excepción MiPrimeraExcepcion
}
}
```

2. Lanzamiento de excepciones. Cláusula throw

Manejar la excepción

Utilizando la instrucción throw, podemos lanzar la excepción a aquella parte del código donde se llama a nuestro método, después de haber realizado un tratamiento de la misma.

```
public void metodoResponsable() throws MiprimeraExcepcion {
MiPrimeraClaseExcepcional mPCE = new MiPrimeraClaseExcepcional();
 try {
        mPCE.unMetodoExcepcional();
          }
catch (MiPrimeraExcepcion m) {
         // aquí haríamos las acciones pertinentes
        // si se produjera la excepción
        MiPrimeraExcepcion throw m; //
        relanzamos la excepción
        }
        }
```

Lanzar excepciones si algo no funciona bien

También podemos lanzar excepciones, si en algún punto del programa detectamos que algo no funciona bien:

```
public class MiPrimeraClaseExcepcional {
    public void unMetodoExcepional() throws MiprimeraExcepcion {
        ...
        if (algoInusualOcurrio()) {
        throw new MiPrimeraExcepcion();
        // la ejecucion ya no llegaría aquí
        }
        }
        ...
}
```

Esta vez throw actúa de forma similar a break, salta sin ejecutar las líneas de código inmediatamente.

Pasos para lanzar la excepción

1. **Se crea el objeto de tipo excepción.**
2. **Con la palabra reservada throw se lanza**

```
3. class Demothrow {
4. static void ProcDemo() {
5. try { throw new NullPointerException ("Mi Error"); }
6. catch (NullPointerException ex)
     { System.out.println ("Captura
     en ProcDemo"); throw ex;
7. }
8. } // fin ProcDemo
9. public static void
     main(String [] args) { try
     {ProcDemo();}
     catch(NullPointerExcept
     ion ex) {
10. System.out.println("Capturada en main"+ex);
11. }
12. } // fin main
13. } // fin de clase
```

3. **Si una función captura una excepción, la excepción viaja por la pila de llamada a funciones hasta que una función la captura o finaliza la ejecución del programa**

main() calcula() divide()
// Error try { calcula() }
catch() { . . . }
En un bloque try se pueden definir varios tipos de excepciones:

```
class MultiCatch {
public static void main(String [] args) {
try { int a = args.length;
int b = 42 / a;
//AritmethicException int[] c =
{2, 8};
c[a] = 99; //IndexOutOfBoundsException
}
catch (AritmethicException ex) { System.out.println (“No hay argumentos”+ex);
}
catch (IndexOutOfBoundsException ex) {
 System.out.println (“Demasiado argumentos”);
}
catch (Exception ex) { // Para  demás excepciones
System.out.println (“Error desconocido”+ex);
}
} // fin de main
} // fin de clase
```