Manejo de Excepciones

Módulo H03

*JAVA: An Introduction to Problem Solving & Programming, 6*th Ed. By Walter Savitch
ISBN 0132162709

Excepciones en Java

- Las excepciones son un objeto que señaliza la ocurrencia de evento inesperado durante la ejecución de un programa.
- Las excepciones están compuestas por dos acciones principales:
  - Lanzar la excepción *(Throw the exception)*
    - Crea un objeto de excepción
  - Manejar la excepción (*Handle the exception)*
    - El código detecta la excepción y la trata de acuerdo a lo que se necesita.

Ejemplo

- Veamos un ejemplo de manejo de excepciones de la manera que lo hemos hecho hasta ahora, a través de **IFs**.

```
1   public static void main (String [] args)
2       {
3           Scanner keyboard = new Scanner (System.in);
4           System.out.println ("Enter number of donuts:");
5           int donutCount = keyboard.nextInt ();
6           System.out.println ("Enter number of glasses of milk:");
7           int milkCount = keyboard.nextInt ();
8           //Dealing with an unusual event without Javas exception
9           //handling features:
10          if (milkCount < 1)
11          {
12              System.out.println ("No milk!");
13              System.out.println ("Go buy some milk.");
14          }
15          else
16          {
17              double donutsPerGlass = donutCount / (double) milkCount;
18              System.out.println (donutCount + " donuts.");
19              System.out.println (milkCount + " glasses of milk.");
20              System.out.println ("You have " + donutsPerGlass +
21                      " donuts for each glass of milk.");
22          }
23          System.out.println ("End of program.");
24      }
25
```

Ejemplo

- Ahora veamos el mismo ejemplo, resuelto a través de **excepciones**.

```
1   public static void main (String [] args)
2     {
3         Scanner keyboard = new Scanner (System.in);
4         try
5         {
6             System.out.println ("Enter number of donuts:");
7             int donutCount = keyboard.nextInt ();
8             System.out.println ("Enter number of glasses of milk:");
9             int milkCount = keyboard.nextInt ();
10            if (milkCount < 1)
11                throw new Exception ("Exception: No milk!");
12            double donutsPerGlass = donutCount / (double) milkCount;
13            System.out.println (donutCount + " donuts.");
14            System.out.println (milkCount + " glasses of milk.");
15            System.out.println ("You have " + donutsPerGlass +
16                    " donuts for each glass of milk.");
17        }
18        catch (Exception e)
19        {
20            System.out.println (e.getMessage ());
21            System.out.println ("Go buy some milk.");
22        }
23        System.out.println ("End of program.");
24    }
```

Excepciones en Java

```
Enter number of donuts:
2
Enter number of glasses of milk:
0
No milk!
Go buy some milk.
End of program.
```

Sample screen output

Excepciones en Java

- Notemos el bloque **`try`**
  - Este bloque contiene código que puede fallar en algunas circunstancias.
  - Si la variable **`milkCount`** es menor a 1, la division:
    `donutCount / (double) milkCount`
    regresaría un resultado incongruente.

```
try
{
    System.out.println ("Enter number of donuts:");
    int donutCount = keyboard.nextInt ();
    System.out.println ("Enter number of glasses of milk:");
    int milkCount = keyboard.nextInt ();
    if (milkCount < 1)
        throw new Exception ("Exception: No milk!");
    double donutsPerGlass = donutCount / (double) milkCount;
    System.out.println (donutCount + " donuts.");
    System.out.println (milkCount + " glasses of milk.");
    System.out.println ("You have " + donutsPerGlass +
            " donuts for each glass of milk.");
}
```

Excepciones en Java

Notemos el bloque **`catch`**

- Cuando lanzamos una excepción, el bloque **`catch`** comienza su ejecución inmediatamente.
- Podemos ver su funcionamiento similar al de un método
- El parámetro de entrada es la Excepción.

```
catch (Exception e)
{
    System.out.println (e.getMessage ());
    System.out.println ("Go buy some milk.");
}
```

Clases de Excepciones Predefinidas

- Java tiene clases de Excepciones predefinidas
- Por ejemplo:
  - **`NullPointerException`**
  - **`BadStringOperationException`**
  - **`ClassNotFoundException`**
  - **`IOException`**
  - **`NoSuchMethodException`**
  - **`IndexOutOfBoundsException`**

Predefined Exception Classes

- Example code

```
SampleClass object = new SampleClass();
try
{
     <Possibly some code>
     object.doStuff(); //may throw IOException
     <Possibly some more code>
}
catch(IOException e)
{
     <Code to deal with the exception, probably including the following:>
     System.out.println(e.getMessage());
}
```

Definición de Nuevas Clases de Excepciones

- Además de las excepciones ya predefinidas en Java, es posible crear clases de excepciones nuevas.
- Para esto, debemos definir una nueva clase que herede de la clase **`Exception`**, o alguna otra clase derivada de la clase **`Exception`**.

```
public class DivideByZeroException extends Exception
{
    public DivideByZeroException () {
        super ("Dividing by Zero!");
    }

    public DivideByZeroException (String message) {
        super (message);
    }
}
```

Definición de Nuevas Clases de Excepciones

Guidelines

- Usa la Clase **`Exception`** como la base de la clase
- Define por lo menos dos constructores:
  - Constructor default, sin parámetros
  - Constructor con un parámetro **`String`**
- Comienza la definición de cada constructor con una llamada al constructor base, usando **`super`**
- No redefinas el método heredado **`getMessage`**

Lanzar Excepciones

Hay 3 formas en las que una excepción puede ser lanzada:

1. Instrucciones de Java
2. La palabra reservada **`throw`**
3. Definiendo un método que lance una excepción.

1. Excepciones por instrucciones

Algunas operaciones inválidas invocan automáticamente una excepción. Por ejemplo:

- Operaciones matemáticas inválidas:
  - ***Arithmetic Operation***
- Acceder a índices inválidos en arreglos
  - ***Index out of bounds***
- Variables de referencia no inicializadas
  - ***Null pointer***

2. Excepciones por `throw`

Un programador puede invocar explícitamente una excepción bajo ciertas condiciones. Para esto, se utiliza la palabra reservada **`throw`** y se instancia un objeto de tipo Excepción.

```
try {
    if (s == null)
        throw new NullPointerException();
    else if (s.length() == 0)
        throw new IllegalArgumentException();
    else
        throw new Exception();
}
```

3. Métodos y Excepciones

- Podemos diseñar métodos que lancen una excepción bajo ciertas condiciones.
- Esto nos da la ventaja de:
  - Delegar el manejo de errores al programa que hace uso del método.
  - Mantener diferentes manejos de errores dependiendo de cada escenario.
- Este comportamiento lo lograremos con la palabra reservada **`throws`**.

3. Métodos y Excepciones

- Debemos incluir las excepciones que un método puede lanzar en la declaración del método:

```
public Type Method_Name(Parameter_List) throws List_Of_Exceptions
Body_Of_Method
```

- OJO!
  - La palabra **`throw`** se usa para lanzar la excepción.
  - La palabra **`throws`** se usa para declarar que el método puede lanzar la excepción.

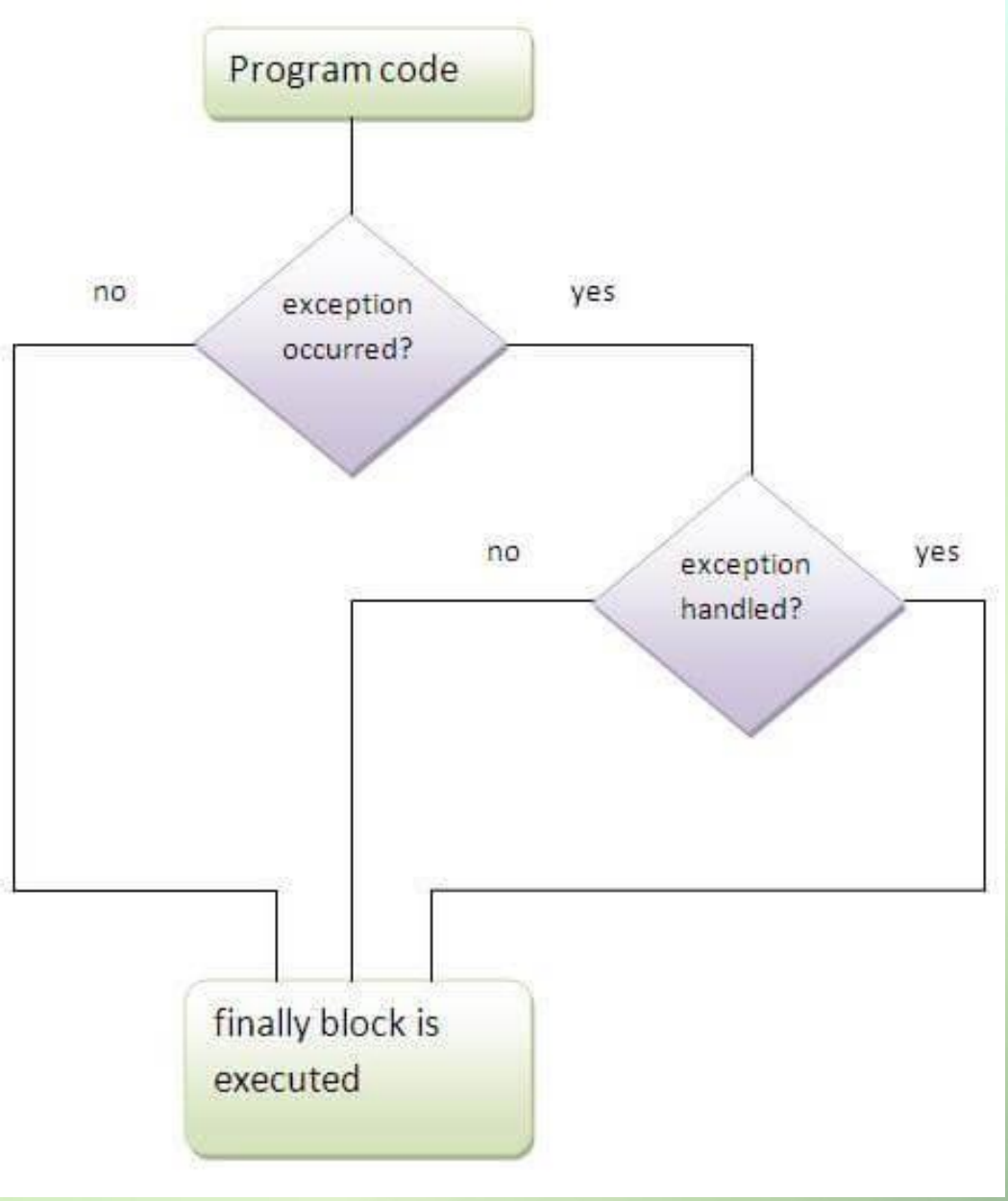
3. Métodos y Excepciones

- Si un método lanza una excepción, el programa que llama al método puede recibirla.
- Ejemplo:

```
9
10      public static void main (String [] args)
11      {
12          try {
13              divide(1,0);
14          } catch(ArithmeticException e) {
15              System.out.println("Excepcion lanzada: " + e.getMessage());
16          }
17      }
18      public static int divide(int a, int b) throws ArithmeticException{
19          return a/b;
20      }
21
```

El bloque `finally`

- Podemos agregar un bloque opcional **`finally`** después de la secuencia del **`catch`**
- El código del bloque **`finally`** siempre se ejecutará, sin importar si la excepción es lanzada o cachada.
  - Aunque haya una instrucción **`return`** dentro del catch, la sección **`finally`** también se ejecutará.
  - La única forma de evitarla es con **`System.exit(0);`**



Resumen

- Las excepciones pueden lanzarse:
  - Desde instrucciones de Java.
  - Desde métodos
  - Cuando el programador usa la instrucción **`throw`**
- Un método que puede lanzar una excepción, pero no cacharla debe usar la cláusula: **`throws`**
- Las excepciones se cachan con un bloque **`catch`**

```
1  public class UsefinallyBlock {
2      public static void main (String[] args) {
3          try {
4              String inputString=args[0];
5              System.out.println("Received: " + inputString);
6          }
7          catch (ArrayIndexOutOfBoundsException ex) {
8              System.out.println("Array index out of bounds exception raised!");
9          }
10         finally {
11             System.out.println("End of program");
12         }
13     }
14 }
```