**write code to handle exceptions with try/catch/finally**

|  |
| --- |
| If an exception is thrown during a sequence of statements inside a try-catch block, |
| the sequence of statements is interrupted and the flow of control will skip directly |
| to the catch-block. |
| This code can be interrupted by exceptions in several places: |
|  |
| public void openFile(){ |
| try { |
| // constructor may throw FileNotFoundException |
| FileReader reader = new FileReader("FileExample"); |
| int i=0; |
| while(i != -1){ |
| //reader.read() may throw IOException |
| i = reader.read(); |
| System.out.println((char) i ); |
| } |
| reader.close(); |
| System.out.println(" End of file "); |
| } catch (FileNotFoundException e) { |
|  |
| } catch (IOException e) { |
|  |
| } |
| } |
|  |
| ==================================================================================== |
|  |
| Finally |
|  |
| You can attach a finally-clause to a try-catch block. The code inside the finally clause will always be executed, even if an exception is thrown from within the try or catch block. If your code has a return statement inside the try or catch block, the code inside the finally-block will get executed before returning from the method. |
|  |
| public void openFile(){ |
| FileReader reader = null; |
| try { |
| reader = new FileReader("someFile"); |
| int i=0; |
| while(i != -1){ |
| i = reader.read(); |
| System.out.println((char) i ); |
| } |
| } catch (IOException e) { |
| } finally { |
| if(reader != null){ |
| try { |
| reader.close(); |
| } catch (IOException e) { |
| } |
| } |
| System.out.println(" End of File "); |
| } |
| } |
| =============================================================================== |
|  |
| Final Keyword In Java |
| The final keyword in java is used to restrict the user. The java final keyword can be used in many context. Final can be: |
| -variable |
| -method |
| -class |
| **1) Java final variable** |
|  |
| If you make any variable as final, you cannot change the value of final variable(It will be constant). |
|  |
| class Maruthi800{ |
| final int speedlimit=70;//final variable |
| void run(){ |
| speedlimit=200; |
| } |
| public static void main(String args[]){ |
| Maruthi800 obj=new Maruthi800(); |
| obj.run(); |
| } |
| }//end of class |
|  |
| Output:Compile Time Error |
|  |
| **2) Java final method** |
|  |
| If you make any method as final, you cannot override it. |
|  |
| class Car{ |
| final void run(){System.out.println("running");} |
| } |
|  |
| class Maruthi extends Car{ |
| void run(){System.out.println("running safely ");} |
|  |
| public static void main(String args[]){ |
| Maruthi maruthi= new Maruthi(); |
| maruthi.run(); |
| } |
| } |
|  |
| **3) Java final class** |
|  |
| If you make any class as final, you cannot extend it. |
|  |
| final class Car{} |
|  |
| class Maruthi1 extends Car{ |
| void run(){System.out.println("running safely");} |
|  |
| public static void main(String args[]){ |
| Maruthi1 maruthi= new Maruthi(); |
| maruthi.run(); |
| } |
| } |