PROGRAM #5 EXCEPTION HANDELING IN JAVA

Manuel Holguin

10-05-2022

Table of Contents

Problem Description

For this programming assignment we were tasked with interpreting an ADA into a JAVA program. Once we performed this process, we then had to make sure to only change the contents of the second portion of the while loop where changes are made to the Freq array. Using JAVA exceptions, we had to make sure to throw an error which would cause the program to run the catch segment. Within this catch segment we had to run all the changed to Freq as per the SRS (System Requirement Specifications) we could not make any of the changes for the array outside of the exception block.

Solution

In order to meet the requirements, I implemented an "ArrayIndexOutOfBoundsException" exception which would by default put us within the catch's scope and all the changes would be done through there.

```
try {
    //Segement causes exception and puts us within the catch section.
    int[] exceptionArray = {1,2,3};
    System.out.println(exceptionArray[5]);

} catch (ArrayIndexOutOfBoundsException e) {
```

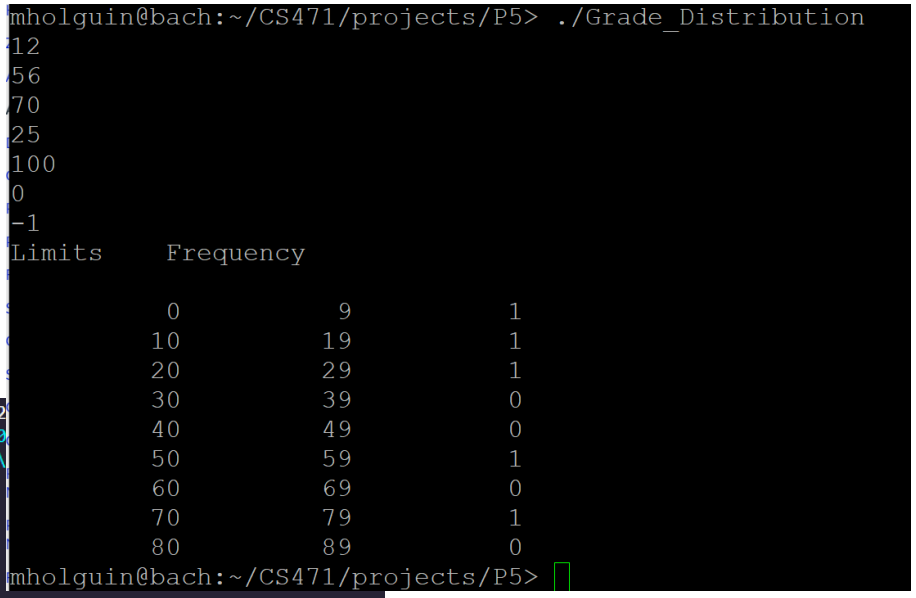
By creating a dummy array of length 3 and then trying to access it's index at 5 which is out of bounds would cause the exception handler to activate and jump us to the catch statement. Whitin this statement all the changes were done to Freq in order to make sure the segment was indeed being executed I inserted a simple print statement that would say "caught exception". As illustrated in the second image I show the utilization of the catch statement to print and run the changes to Freq.

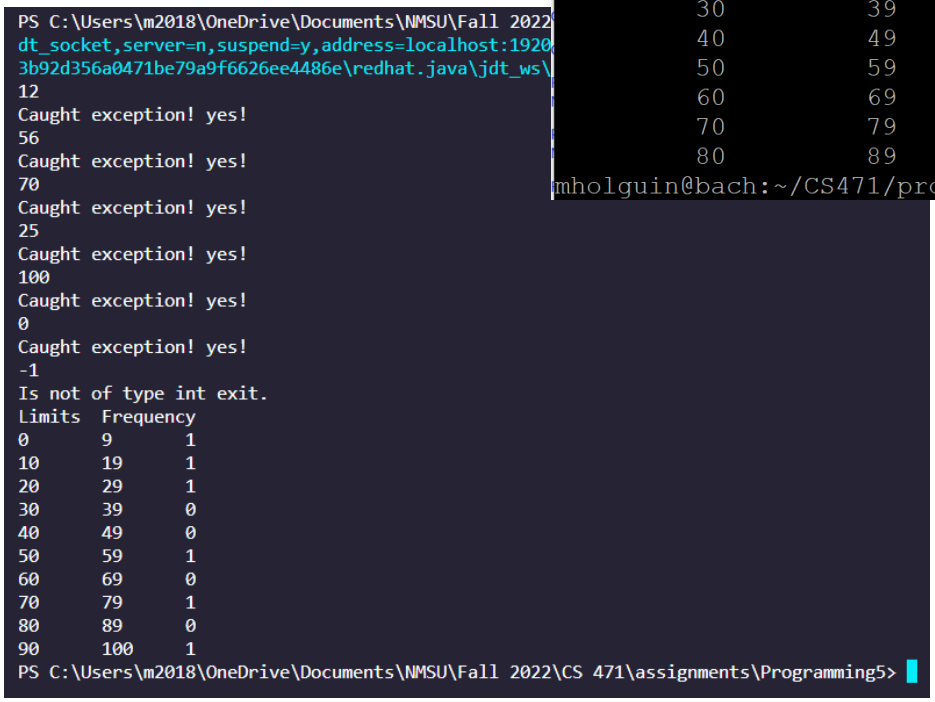
```
} catch (ArrayIndexOutOfBoundsException e) {
    //In this segment all changes to Freq are made//
    //For debugging purposes and to see if we really are in the catch scope.
    System.out.println(x: "Caught exception! yes!");

    ////////////////Changes to Freq array//////////////////////
    if (NewGrade == 100) {
        Freq[9] = Freq[9] + 1;
    }
    else if( NewGrade < 100){
        Freq[index] = Freq[index] + 1;
    }
    else {
        System.out.println("Error-- " + NewGrade + " --is out of range.");
        e.printStackTrace();
    }
    ////////////////////end of the changes//////////////////////////
```

Conclusion

Finally, to make sure the program was working equally to ADA I ran the programs in parallel and compared the printout results for both. The assignment has shown me that programming can still occur outside a regular scope and be run within an exception's catch statement the system does not necessarily have to stop when it hits an exception. Below are the images of the outcomes for the assignment concluding my findings for programming assignment # 5 project given to us by Dr. Cooper. The statement "Caught in exception! yes!" was used to prove the code in the catch segment was ran.

Conclusion Images





Code Base

Sample code from ADA

```
with Ada.Text_IO , Ada.Integer_Text_IO ;
use Ada.Text_IO, Ada.Integer_Text_IO;
procedure Grade_Distribution is
  Freq: array (1..10) of Integer := (others => 0);
  New_Grade : Natural;
  Index,
  Limit_1,
  Limit_2 : Integer;
  begin
  Grade_Loop:
    loop
    begin
    Get(New_Grade);
    exception
     when Constraint_Error =>
       exit Grade_Loop;
    end;
     Index := New_Grade/10 + 1;
       begin
        Freq(Index) := Freq(Index) +1 ;
        exception
           when Constraint_Error =>
            if New_Grade = 100 then
              Freq(10) := Freq(10) + 1;
            else
             Put("Error -- new grade: ");
             Put(New_Grade);
Put(" is out of range");
New_Line;
     end if;
end;
   end loop Grade_Loop;
   Put("Limits    Frequency");
   New_Line; New_Line;
   for Index in 0..8 loop
     Limit_1 := 10 * Index;
     Limit_2 := Limit_1 + 9;
     if Index = 9 then
       Limit_2 := 100;
     end if;
     Put(Limit_1);
     Put(Limit_2);
```

```
      Put(Freq(Index +1));
      New_Line;
    end loop;
end Grade_Distribution;
```

JAVA

```
/*Author: Manuel Holguin
 * Title: Grades.java
 * Date: 10-04-2022
 * Description: This program is a interpretation from the grade_distribution.adb Ada code
 *          which was translated into Java.  Per project specifications only the second
 *          exception in the code was modified to run the updating of Freq array within
 *          the catch block instead of the try block. Putting in a negative number in as
 *          input breaks the while loop as the same in the ADA code.
 */
import java.util.*;
import java.util.Scanner;

public class Grades {

   static void Grade_Distribution() {
      Scanner getGrade = new Scanner(System.in);
      // Initialized required variables
      int Freq[] = new int[10];
      int NewGrade = 0;
      int index = 0, limit_1 = 0, limit_2 = 0;
      boolean innerwhile = true;
      // Create grade loop
      while (innerwhile) {
         // First exception will catch if the input from the user is not an int and if the number is
         // a natural number no negative numbers are allowed otherwise if negative exit loop.
            try {
               //Gets the grade input from the user since we are working with Natural numbers
               // 0,1,2,3 ... n any negative number throws the runtime exception which exists the loop
```

```
      //Otherwise the user can continue to input numbers until they exit by exception.
      NewGrade = getGrade.nextInt();
      if (NewGrade < 0) {
         throw new RuntimeException("Not seen: Must be a positive number!");
      }
   }
   // If the input is not an int run exception and exit loop.
   catch (RuntimeException e) {
      System.out.println("Is not of type int exit.");
      innerwhile = false;
      break;
   }
index = NewGrade / 10  ;
//*88888888888888888888888888888888888888888888888888888888 */
//Segment for debugging purposes
//System.out.println("This is the Index" + index + "Length of array is " + Freq.length);
/*88888888888888888888888888888888888888888888888888888888 */

/*This block runs the second exception in the program. By assignment specifications
 *this exception is the only one modified to always execute throw and activate the catch.
 *The exception used in the program is ArrayIndexOutOfbounds
 * which runs the catch portion that does all the changes to the array Freq within catch*/
try {
   //Segement causes exception and puts us within the catch section.
   int[] exceptionArray = {1,2,3};
   System.out.println(exceptionArray[5]);

} catch (ArrayIndexOutOfBoundsException e) {
   //In this segment all changes to Freq are made//
```

```
      //For debugging purposes and to see if we really are in the catch scope.
      System.out.println("Caught exception! yes!");

      /////////////////Changes to Freq array//////////////////////
      if (NewGrade == 100) {
         Freq[9] = Freq[9] + 1;
      }
      else if( NewGrade < 100){
         Freq[index] = Freq[index] + 1;
      }
      else {
         System.out.println("Error-- " + NewGrade + " --is out of range.");
         e.printStackTrace();
      }
      ////////////////////end of the changes//////////////////////////
   }
} // Closure of grade loop

// While loop that shall not be changed per assignment requirements only prints results
   System.out.println("Limits  Frequency");
   for (int k = 0; k <= 9; k++) {
      limit_1 = 10 * k;
      limit_2 = limit_1 + 9;
      if (k == 9)
         limit_2 = 100;

      System.out.print(limit_1 + "\t");
      System.out.print(limit_2 + "\t");
      System.out.print(Freq[k]);
```

```
            System.out.println();
        }

    }

    public static void main(String[] args) {

        Grade_Distribution();
    }
}
```