Andrew Baca
October 1, 2018
CS471
Exception Handling

Purpose: We will test exception handling in java to catch run time errors. Exception handling is useful to detect errors at run time and take special steps to deal with these errors rather than to hit a hard fail and stop the process. In other words, exception handling is good for when you want the program to continue execution when certain errors or hit, rather than stopping execution.

Code: (Mainly sourced from ch14 of the textbook PG 612 - 613)

```
/*************************************************************************************************************************************
* Andrew Baca
* September 28, 2018
* CS 471
* Grade_Distribution.java
*
* Java Exception Handling
*
* Purpose: We are to re-write the Given ada program for Grade Distributions. Our task , in your JAVA version, is to change the second
* half of the first  loop so that all assignments to the counting Array "Freq()" are updated in the Exception portion of the code.
* There should be no valid updates to "Freq()" anywhere else in the loop.
*
* IMPORTANT!!!!!!******************RETRIEVED FROM: Chapter 14 of Concepts Of Programming Languages page 612 - 613**********************
*
*************************************************************************************************************************************/

import java.util.Scanner;                    //import for communication with user

class NegativeInputException extends Exception{        //negative input exception to print and hit the catch once triggered

        public NegativeInputException() {
                System.out.println("Negative Input. End of Data Input Reached");
        }

}

class Grade_Distribution{



        public static void main(String args[]){

                Scanner scnr = new Scanner(System.in);

                int freq[] = new int[10];
                int newGrade = 0;
                int index, limit1, limit2;
```

```
		try{
			while(true){
				System.out.println("Please enter a grade: ");
				newGrade = scnr.nextInt();
				if(newGrade < 0)                              //will hit this expression, and jump to the catch if throw is triggered
					throw new NegativeInputException();
				index = newGrade / 10;
				try{
					freq[index]++;
				}
				catch(ArrayIndexOutOfBoundsException e) {      //will hit this catch and go to the top of the while loop if num exceeds 100
					if(newGrade == 100)
						freq[9]++;
					else
						System.out.println("Error - New Grade: " + newGrade + " is out of range");
				}
			}
		}
		catch(NegativeInputException e) {                                  //will reach this from the negative input, and print frequencies
			System.out.println("\n\tLimits \t Frequency\n");
			for(index = 0; index < 10; index++){
				limit1 = 10 * index;
				limit2 = limit1 + 9;
				if(index == 9)
					limit2 = 100;
				System.out.println("\t" + limit1 + " \t" + limit2 + "       " + freq[index]);
			}
		}

	}
}
```

Output (next page)

```
CS471/JavaExcept> javac Grade_Distribution.java
CS471/JavaExcept> java Grade_Distribution
Please enter a grade:
100
Please enter a grade:
95
Please enter a grade:
90
Please enter a grade:
110
Error - New Grade: 110 is out of range
Please enter a grade:
1000
Error - New Grade: 1000 is out of range
Please enter a grade:
50
Please enter a grade:
30
Please enter a grade:
10
Please enter a grade:
11
Please enter a grade:
1
Please enter a grade:
1
Please enter a grade:
2
Please enter a grade:
55
Please enter a grade:
57
Please enter a grade:
-1
Negative Input. End of Data Input Reached

        Limits   Frequency

        0       9       3
        10      19       2
        20      29       0
        30      39       1
        40      49       0
        50      59       3
        60      69       0
        70      79       0
        80      89       0
        90      100       3
```