

**1 EX 10.2 Describe the output for the *ProductCodes* program if a finally clause were added to the try statement that printed the string "Got here!".**

```
//*****
// ProductCodes.java      Author: Lewis/Loftus
//
// Demonstrates the use of a try-catch block.
//*****

import java.util.Scanner;

public class Generic
{
    //-----
    // Counts the number of product codes that are entered with a
    // zone of R and and district greater than 2000.
    //-----
    public static void main (String[] args)
    {
        String code;
        char zone;
        int district, valid = 0, banned = 0;

        Scanner scan = new Scanner (System.in);

        System.out.print ("Enter product code (STOP to quit): ");
        code = scan.nextLine();

        while (!code.equals ("STOP"))
        {
            try
            {
                zone = code.charAt(9);
                district = Integer.parseInt(code.substring(3, 7));
                valid++;
                if (zone == 'R' && district > 2000)
                    banned++;
            }
            catch (StringIndexOutOfBoundsException exception)
            {
                System.out.println ("Improper code length: " + code);
            }
            catch (NumberFormatException exception)
            {
                System.out.println ("District is not numeric: " + code);
            }
            finally
            {
                System.out.println ("Got here!");
            }
        }
    }
}
```

```

        System.out.print ("Enter product code (STOP to quit): ");
        code = scan.nextLine();
    }

    System.out.println ("# of valid codes entered: " + valid);
    System.out.println ("# of banned codes entered: " + banned);
}
}

```

### Output:

```

<terminated> Generic [Java Application] C:\Program Files\Java\jdk1.6.0_35\bin\javaw.exe (Feb 16, 2015, 9:22:00 PM)
Enter product code (STOP to quit): RNA123456789RRAA
Got here!
Enter product code (STOP to quit): STOP
# of valid codes entered: 1
# of banned codes entered: 0

```

The finally clause did not serve any purpose here other than causing the CPU to be more busy for no reason.

## **2 EX 10.6 Look up the following exception classes in the online Java API documentation and describe their purpose:**

- ArithmeticException
- NullPointerException
- NumberFormatException
- PatternSyntaxException

Exception	Purpose		
ArithmeticException	Thrown when an exceptional arithmetic condition has occurred. For example, an integer "divide by zero" throws an instance of this class. ArithmeticException objects may be constructed by the virtual machine as if suppression were disabled and/or the stack trace was not writable.		
NullPointerException	Thrown when an application attempts to use null in a case where an object is required. These include: <ul style="list-style-type: none"> <li>Calling the instance method of a null object.</li> <li>Accessing or modifying the field of a null object.</li> </ul>		

	<ul style="list-style-type: none"> <li>• Taking the length of null as if it were an array.</li> <li>• Accessing or modifying the slots of null as if it were an array.</li> <li>• Throwing null as if it were a Throwable value.</li> </ul> <p>Applications should throw instances of this class to indicate other illegal uses of the null object. NullPointerException objects may be constructed by the virtual machine as if suppression were disabled and/or the stack trace was not writable.</p>		
<b>NumberFormatException</b>	Thrown to indicate that the application has attempted to convert a string to one of the numeric types, but that the string does not have the appropriate format.		
<b>PatternSyntaxException</b>	Unchecked exception thrown to indicate a syntax error in a regular-expression pattern.		

**3      PP 10.1 Design and implement a program that reads a series of 10 integers from the user and prints their average. Read each input value as a string, and then attempt to convert it to an integer using the `Integer.parseInt` method. If this process throws a `NumberFormatException` (meaning that the input is not a valid number), print an appropriate error message and prompt for the number again. Continue reading values until 10 valid integers have been entered.**

```
//*****
//Author: Hieu Pham
//ID: 0953-827
//Section: 28317
//Date: 02/16/2015
//
//PP 10.1 Design and implement a program that reads a series of 10
//integers from the user and prints their average. Read each input
//value as a string, and then attempt to convert it to an integer
//using the Integer.parseInt method. If this process throws a
```

```

//NumberFormatException (meaning that the input is not a valid number),
//print an appropriate error message and prompt for the number again.
//Continue reading values until 10 valid integers have been entered.
//
//*****
import java.util.Scanner;
public class ReadTen
{
    public static void main (String [] args)
    {
        //int a,b;
        int[] numbers = new int [10]; //In case of need for an int array
        int count = 0; int a = 0; //Variables used in program
        String aString = ""; //String buffer
        int sum = 0; //Used for summing at the end

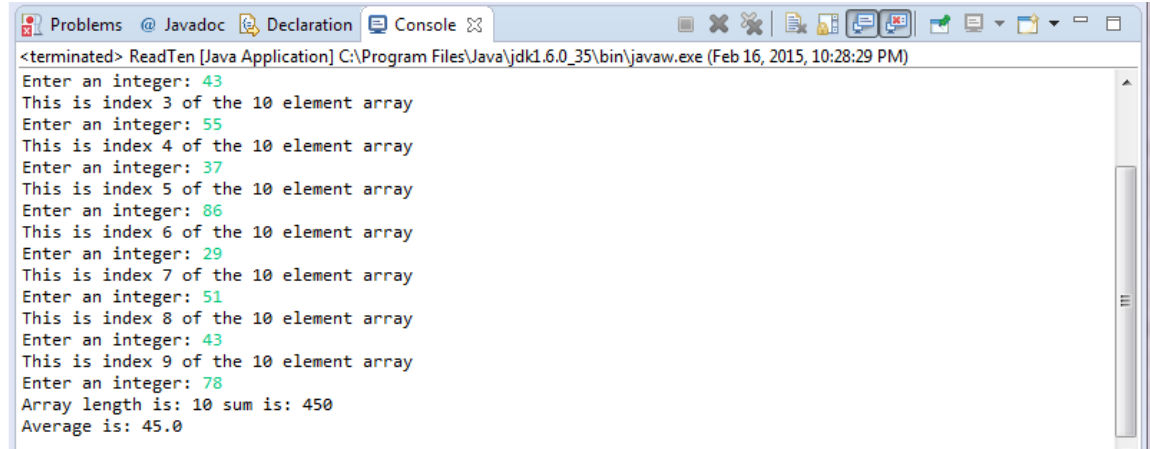
        Scanner ReadNumbs = new Scanner(System.in); //Read from keyboard
        do
        {
            System.out.println("This is index " + count + " of the 10 element
array");
            try
            {
                System.out.print("Enter an integer: ");
                aString = ReadNumbs.nextLine();
                a = Integer.parseInt(aString);
                count += 1; //If everything is good, move on, increment
count
            }
            catch(NumberFormatException e)
            {
                System.out.println("An exception has been thrown. " +
aString +
                " is not a valid number");
                count -= 1; //Otherwise, decrement count
            }
            finally
            {
                sum = sum + a; //No matter what, update the sum
            }

        }while(count < 10); //Get out when 10 numbers reached

        System.out.println("Array length is: " + numbers.length + " sum is:
" + sum);
        System.out.println("Average is: " + ((float) (sum /
numbers.length)));
    }
}

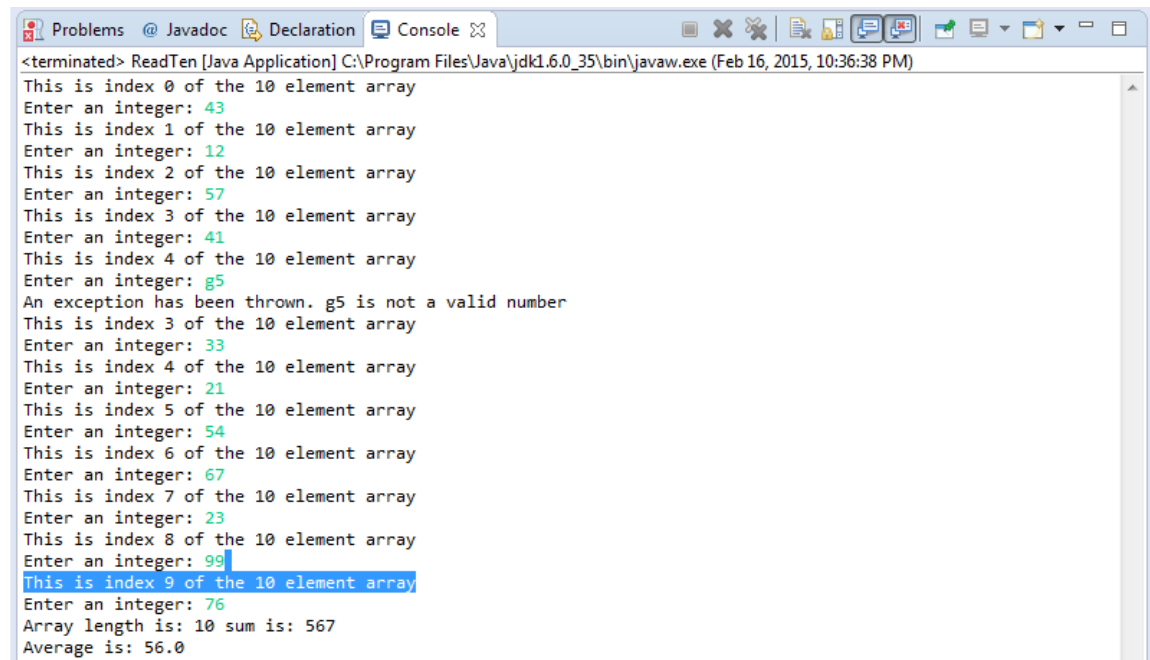
```

### Output (Good):



```
<terminated> ReadTen [Java Application] C:\Program Files\Java\jdk1.6.0_35\bin\javaw.exe (Feb 16, 2015, 10:28:29 PM)
Enter an integer: 43
This is index 3 of the 10 element array
Enter an integer: 55
This is index 4 of the 10 element array
Enter an integer: 37
This is index 5 of the 10 element array
Enter an integer: 86
This is index 6 of the 10 element array
Enter an integer: 29
This is index 7 of the 10 element array
Enter an integer: 51
This is index 8 of the 10 element array
Enter an integer: 43
This is index 9 of the 10 element array
Enter an integer: 78
Array length is: 10 sum is: 450
Average is: 45.0
```

### Output (Bad):



```
<terminated> ReadTen [Java Application] C:\Program Files\Java\jdk1.6.0_35\bin\javaw.exe (Feb 16, 2015, 10:36:38 PM)
This is index 0 of the 10 element array
Enter an integer: 43
This is index 1 of the 10 element array
Enter an integer: 12
This is index 2 of the 10 element array
Enter an integer: 57
This is index 3 of the 10 element array
Enter an integer: 41
This is index 4 of the 10 element array
Enter an integer: g5
An exception has been thrown. g5 is not a valid number
This is index 3 of the 10 element array
Enter an integer: 33
This is index 4 of the 10 element array
Enter an integer: 21
This is index 5 of the 10 element array
Enter an integer: 54
This is index 6 of the 10 element array
Enter an integer: 67
This is index 7 of the 10 element array
Enter an integer: 23
This is index 8 of the 10 element array
Enter an integer: 99
This is index 9 of the 10 element array
Enter an integer: 76
Array length is: 10 sum is: 567
Average is: 56.0
```

**4 PP 10.6** Write a program that reads strings from the user and writes them to an output file called *userStrings.dat*. Terminate processing when the user enters the string "DONE". Do not write the sentinel string to the output file.

```

//*****
//  TestData.java      Author: Lewis/Loftus
//
//  Demonstrates I/O exceptions and the use of a character file
//  output stream.
//*****

//*****
//Author: Hieu Pham
//ID: 0953-827
//Section: 28317
//Date: 02/16/2015
//
//PP 10.6 Write a program that reads strings from the user and writes them
//to an output file called userStrings.dat. Terminate processing when the
//user enters the string "DONE". Do not write the sentinel string to the
//output file.
//Most of the work was based on Lewis example.
//*****
import java.util.Scanner;
import java.io.*;

public class UserString
{
    //-----
    //  Creates a file of test data that consists of ten lines each
    //  containing ten integer values in the range 10 to 99.
    //-----
    public static void main (String[] args) throws IOException
    {
        String file = "userStrings.dat";
        String code;
        Scanner scan = new Scanner (System.in);

        FileWriter fw = new FileWriter(file);//Inherited from the author.
        BufferedWriter bw = new BufferedWriter(fw);//Inherited from the author.
        PrintWriter outFile = new PrintWriter(bw);//Inherited from the author.
        //Basically bw contains fw, and outFile contains bw, hence buffering.

        System.out.print ("Enter product code (DONE to quit): ");
        code = scan.nextLine();

        while (!code.equals ("DONE"))
        {
            if(!code.equals ("DONE"))//Don't write the sentinel
            {
                outFile.print(code); //Store the user string
                outFile.println();//Adds a new line
            }
            System.out.print ("Enter product code (DONE to quit): ");
            code = scan.nextLine(); //Continue
        }
        outFile.close();//Close the output file
        System.out.println ("Output file has been created: " + file);
    }
}

```

}

### Output:

```
<terminated> UserString [Java Application] C:\Program Files\Java\jdk1.6.0_35\bin\javaw.exe (Feb 16, 2015, 11:05:38 PM)
Enter product code (STOP to quit): THIS
Enter product code (STOP to quit): IS
Enter product code (STOP to quit): A-Good
Enter product code (STOP to quit): Test
Enter product code (STOP to quit): DONE
Output file has been created: userStrings.dat
```

### 4.3 EX 11.1 What is the order of the following growth functions?

- a.  $10n^2 + 100n + 1000$
- b.  $10n^3 - 7$
- c.  $2n + 100n^3$
- d.  $n^2 \log n$

Growth Function		Order	
$10n^2 + 100n + 1000$		$O(n^2)$	
$10n^3 - 7$		$O(n^3)$	
$2n + 100n^3$		$O(n)$	
$n^2 \log n$		$O(n^2 \log n)$	

### 5 EX 11.2 Arrange the growth functions of the previous exercise in ascending order of efficiency for $n=10$ and again for $n = 1,000,000$ .

Growth Function	Order	$n = 10$	$n = 1000000$
$10n^2 + 100n + 1000$	$O(n^2)$	$n^2 \log n$	$n^2 \log n$
$10n^3 - 7$	$O(n^3)$	$10n^2 + 100n + 1000$	$10n^2 + 100n + 1000$
$2n + 100n^3$	$O(n^3)$	$10n^3 - 7$	$10n^3 - 7$
$n^2 \log n$	$O(n^2 \log n)$	$10n^3 - 7$	$10n^3 - 7$