CS1410
Exception Handling Lab

Name: _Anthony Browness____________________ Section: __004_______

Matching

After reading Chapter 11 *Exception Handling: a Deeper Look* of *Java How to Program: 9/e*, answer the given questions. These questions are intended to test and reinforce your understanding of key Java concepts.

For each term in the column on the left, write the corresponding letter for the description that best matches it from the column on the right.

_D___ 1. `try` block	a) Keyword that initiates an exception.
_I___ 2. `finally` block	b) Displays the method-call stack at the time that an exception occurred.
_A____3. `Exception`	c) Thrown when a program attempts to divide by zero in integer arithmetic.
_L____ 4. `catch`	d) Contains code that may generate exceptions.
_A___ 5. `throw`	e) Superclass from which all exceptions are derived.
_C___ 6. `ArithmeticException`	f) Serious problem from which most programs cannot recover.
_K___ 7. `NumberFormatException`	g) Exception that can occur at any point during the execution of the program and can usually be avoided by coding properly.
_B___ 8. `printStackTrace`	h) Method that returns an array of `StackTraceElements`.
_J___ 9. stack unwinding	i) Keyword that begins the declaration of an exception handler.
_H___ 10. `getStackTrace`	j) The process by which an exception that is not caught is returned to a calling method in an attempt to locate an appropriate exception handler.
_G___ 11. `RuntimeException`	k) Occurs when an attempt is made to convert a `String` to a numeric value and the `String` does not represent a number.
_F___ 12. `Error`	l) Typically, contains code that releases resources allocated in its corresponding `try` block.

Fill in the Blank

Fill in the blanks for each of the following statements:

13. A(n) _Exception____________ is an indication that a problem occurred during the program’s execution.

14. Each _Catch_______________ specifies the type of exception it can handle.

15. Only _Thrown_____________ objects can be used with the exception-handling mechanism.

16. If no exception handler matches a particular thrown object, the search for a match continues with the exception handlers of an enclosing _Exception Class______________.

17. Once an exception is thrown, program control cannot return directly to the __Thread__________.

18. A `catch` clause for type _Exception_______________ can handle exceptions of any type.

19. A `catch` clause for type _Throwable________________ can catch any object that can be used with the exception-handling mechanism.

20. A(n) _Finally_________________ always executes as long as program control enters its corresponding `try` block.

21. A(n) __Stack Trace________________ lists the exceptions that a method might throw.

22. Class `Throwable` has two subclasses— __Exception________________ and _Error____________________.

23. There are two categories of exceptions in Java— _IOException______________________and __RunTimeException_____________________.

24. A(n) _Try______________________must be true when a method is invoked and a(n) must be true after a method successfully
returns.

Short Answer

In the space provided, answer each of the given questions. Your answers should be concise; aim for two or three sentences.

25. Explain when exception handling should be used.

Exception Handling should be used when processing synchronous errors, or errors that occur when a statement executes. Basically anything that runs within the confines of the Programs control.

26. What is the difference between the termination model of exception handling, used in Java, and the resumption model of exception handling?

Termination stops the flow of the program and does not execute any further statements in the Try block. Resumption just logs the error and continues one.

27. Describe the general flow of control through a `try…catch…finally` when an exception occurs and is caught.

First the Try block is entered, this is where the error could be thrown in the program, once an error is thrown it an exception is looked for in the catch block. The finally block allows for a slight resumption in program control after the try block.

28. Describe the general flow of control through a `try…catch…finally` when an exception occurs and is not caught. What happens to the exception object that was thrown?

If an Exception occurs in the try block and there is no catch block to correct the thrown exception then the finally block will still execute. This creates and uncaught exception which will terminate the program if it is single threaded.

29. Explain the restrictions on the `throws` clause of a subclass method that overrides a superclass method.

The throws clause must be stated in the method parameter list and before the method body. It lists the possible exceptions that may be encountered during the methods execution.

30. Explain the "catch or declare" requirement of Java exception handling. How does this affect exception types that are direct or indirect subclasses of `RuntimeException`?

The code that is to be caught must be within a methods try block, if not then the Exception must be declared in the Methods throws clause.

31. Why would a `catch` block rethrow an exception?

When the catch block receives the exception and can only process part of it or cannot process it, it will rethrow the exception to another catch block that may fit the exception better.

32. Explain the process of stack unwinding

The method terminates and an attempt is made to catch the exception in the outer try block of that method.

Programming Output

For each of the given program segments, read the code and write the output in the space provided below each program. [*Note:* Do not execute these programs on a computer.]

33. What is output by the following application?

```
public class Test
{
   public static String lessThan100( int number ) throws Exception
   {
        if ( number >= 100 )
             throw new Exception( "Number too large." );

        return String.format( "The number %d is less than 100", number );
   }

   public static void main( String args[] )
   {
        try
        {
             System.out.println( lessThan100( 1 ) );
             System.out.println( lessThan100( 22 ) );
             System.out.println( lessThan100( 100 ) );
             System.out.println( lessThan100( 11 ) );
        }
        catch( Exception exception )
        {
             System.out.println( exception.toString() );
        }
   }// end main method
}// end class Test
```

Your answer:

```
The number 1 is less than 100
The number 22 is less than 100
Number too large
The number 11 is less than 100
```

34. What is output by the following program if the user enters the values `3` and `4.7`?

```
import javax.swing.JOptionPane;

public class Test
{
     public static String sum( int num1, int num2 )
     {
          return String.format( "%d + %d = %d", num1, num2, ( num1 + num2 ) );
     }

     public static void main( String args[] )
     {
          int number1;
          int number2;

          try
          {
               number1 =
                    Integer.parseInt( JOptionPane.showInputDialog( "Enter an integer: " ) );

               number2 = Integer.parseInt(
                    JOptionPane.showInputDialog( "Enter another integer: " ) );

               System.out.println( sum( number1, number2 ) );
          }
          catch ( NumberFormatException numberFormatException )
          {
               System.out.println( numberFormatException.toString() );
          }
     } // end main method
} // end class Test
```

Your answer:
A box will pop up when the user enters 4.7 that will say "Enter another integer: "

Correct the Code

Determine if there is an error in each of the following program segments. If there is an error, specify whether it is a logic error or a compilation error, circle the error in the program and write the corrected code in the space provided after each problem. If the code does not contain an error, write “no error.” [*Note:* There may be more than one error in each program segment.]

35. The following code segment should `catch` only `NumberFormatException`s and display an error message dialog if such an exception occurs:

```
catch ( Exception exception )
     JOptionPane.showMessageDialog( this,
          "A number format exception has occurred",
          "Invalid Number Format", JOptionPane.ERROR_MESSAGE );
```

Your answer:

```
Catch( NumberFormatException numberFormatException)
{
     JOptionPane.showMessageDialog( this,
               "A number format exception has occurred",
               "Invalid Number Format", JOptionPane.ERROR_MESSAGE );
}
```

36. In the following code segment, assume that `method1` can throw both `NumberFormatException`s and `ArithmeticException`s. The following code segment should provide appropriate exception handlers for each exception type and should display an appropriate error message dialog in each case:

```
try
     method1();

catch ( NumberFormatException n, ArithmeticException a )
{
     JOptionPane.showMessageDialog( this,
          String.format( "The following exception occurred¥n%s¥n%s¥n",
          n.toString(), a.toSting() ),
          "Exception occurred", JOptionPane.ERROR_MESSAGE );
}
```

Your answer:

```
   try
        method1();

   catch ( NumberFormatException numberFormatException)
   {
        JOptionPane.showMessageDialog( this,
             String.format( "The following exception occurred: ",
             numberFormatException.toString() ),
             "Incorrect Number Format. Please Try Again.", JOptionPane.ERROR_MESSAGE );
   }
   Catch(ArithmeticException arithmeticExecption)
   {
        JOptionPane.showMessageDialog( this,
             String.format( "The following exception occurred: ",
             arithmeticException.toString() ),
             "Arithmetic Error. Please Try Again.", JOptionPane.ERROR_MESSAGE );
   }
```

Lab Exercise — Access Array

The problem is divided into five parts:

1. Lab Objectives
2. Description of the Problem
3. Program Template
4. Problem-Solving Tips
5. Sample Output

The program template represents a complete working Java program with one or more key lines of code replaced with comments. Read the problem description and examine the sample output, then study the template code. Using the problem-solving tips as a guide, replace the `/* */` comments with Java code. Compile and execute the program. Compare your output with the sample output provided. Then answer the follow-up question. The source code for the template is available at `www.pearsonhighered.com/deitel`.

Lab Objectives

This lab was designed to reinforce programming concepts from Chapter 1 of *Java How to Program: 9/e*. In this lab, you will practice:

- Using exception handling to determine valid inputs.
- Using exception handling to write more robust and more fault-tolerant programs. The follow-up question and activity also will give you practice:
- Creating your own exception type and throwing exceptions of that type.

Description of the Problem

Write a program that allows a user to input integer values into a 10-element array and search the array. The program should allow the user to retrieve values from the array by index or by specifying a value to locate. The program should handle any exceptions that might arise when inputting values or accessing array elements. The program should throw a `NumberNotFoundException` if a particular value cannot be found in the array during a search. If an attempt is made to access an element outside the array bounds, catch the `ArrayIndexOutOfBoundsException` and display an appropriate error message. Also, the program should throw an `Array-IndexOutOfBoundsException` if an attempt is made to access an element for which the user has not yet input a value.

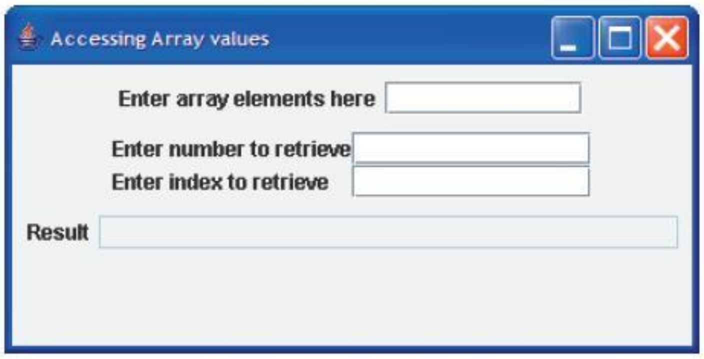
Program Template (Download from the Canvas Files page)

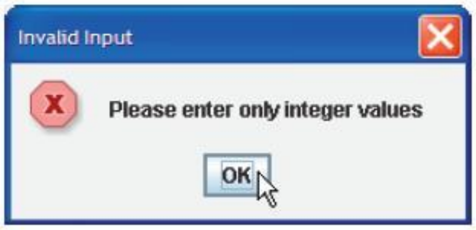
`ArrayAccess.java, ArrayAccessTest.java, NumberNotFoundException.java)`

Problem-Solving Tips

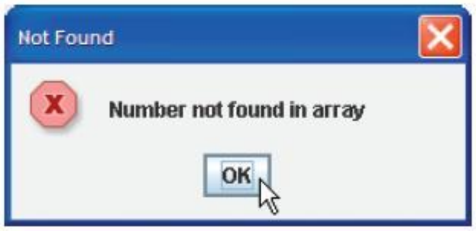
1. When you search the array for a value, you should define a `boolean` value at the beginning of the `try` block and initialize it to `false`. If the value is found in the array, set the `boolean` value to `true`. This will help you determine whether you need to `throw` an exception due to a search key that is not found.
2. Refer to the sample output to decide what messages to display in the error dialogs.
3. Each of the three event handlers will have its own `try` statement.

Sample Output

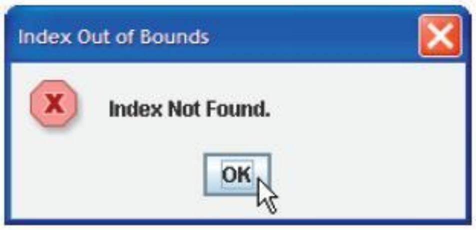




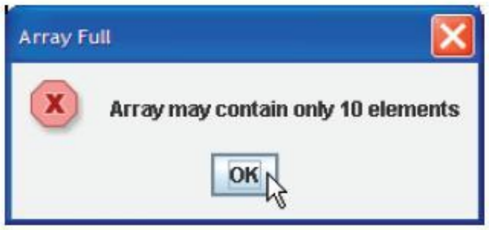
Please enter only integer values



Number not found in array



Index Not Found.



Array may contain only 10 elements

Follow-Up Question and Activity

Create another exception class called DuplicateValueException that will be thrown if the user inputs a value that already resides in the array. Modify your lab exercise solution to use this new exception class to indicate when a duplicate value is input, in which case an an appropriate error message should be displayed. The program should continue normal execution after handling the exception.

```
public class DuplicateValueException extends Exception
{
        // no-argument constructor specifies default error message
        public DuplicateValueException()
        {
                super( "Duplicate Values are not supported" );
  }

  // constructor to allow customized error message
  public DuplicateValueException( String message )
  {
    super( message );
  }
} // end class DuplicateValueException

try
{
        String text = inputField.getText();
        num = Integer.parseInt(text);
        for(int i = 0; i < array.length; i++)
        {
                if(array[i] == num)
                {
                        throw new DuplicateValueException();
                }
        }
        array[index] = num;
        index++;
}

catch(DuplicateValueException duplicateValueException)
{
        JOptionPane.showMessageDialog(null, "Please do not enter Duplicate values", "Invalid
Input", JOptionPane.ERROR_MESSAGE);
}
```