**Exception Handling Lab**

Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Section: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Matching**

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

\_\_l\_\_\_ 2. finally block

\_\_\_\_\_ 3. Exception

\_\_\_i\_\_ 4. catch

\_\_\_\_\_ 5. throw

\_\_\_\_\_ 6. ArithmeticException

\_\_\_\_\_ 7. NumberFormatException

\_\_\_\_\_ 8. printStackTrace

\_\_\_\_\_ 9. stack unwinding

\_\_\_\_\_ 10. getStackTrace

\_\_\_\_\_ 11. RuntimeException

\_\_\_\_\_ 12. Error

a) Keyword that initiates an exception.

b) Displays the method-call stack at the time that an exception occurred.

c) Thrown when a program attempts to divide by zero in integer arithmetic.

d) Contains code that may generate exceptions.

e) Superclass from which all exceptions are derived.

f) Serious problem from which most programs cannot recover.

g) Exception that can occur at any point during the execution of the program and can usually be avoided by coding properly.

h) Method that returns an array of StackTraceElements.

i) Keyword that begins the declaration of an exception handler.

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.

k) Occurs when an attempt is made to convert a String to a numeric value and the String does not represent a number.

l) Typically, contains code that releases resources allocated in its corresponding try block.

**Fill in the Blank**

**Fill in the Blank**

Fill in the blanks for each of the following statements:

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

14. Each \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ specifies the type of exception it can handle.

15. Only \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 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 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

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

18. A catch clause for type \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ can handle exceptions of any type.

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

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

21. A(n) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ lists the exceptions that a method might throw.

22. Class Throwable has two subclasses— \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ and \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

23. There are two categories of exceptions in Java— \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_and \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

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

**Short Answer**

**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.

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

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

caught.

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?

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

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?

31. Why would a catch block rethrow an exception?

32. Explain the process of stack unwinding

**Programming Output**

**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?

**1** public class Test

**2** {

**3** public static String lessThan100( int number ) throws Exception

**4** {

**5** if ( number >= 100 )

**6** throw new Exception( "Number too large." );

**7**

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

**9** }

**10**

**11** public static void main( String args[] )

**12** {

**13** try

**14** {

**15** System.out.println( lessThan100( 1 ) );

**16** System.out.println( lessThan100( 22 ) );

**17** System.out.println( lessThan100( 100 ) );

**18** System.out.println( lessThan100( 11 ) );

**19** }

**20** catch( Exception exception )

**21** {

**22** System.out.println( exception.toString() );

**23** }

24 }// end main method

25 }// end class Test

*Your answer:*

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

**1** import javax.swing.JOptionPane;

**2**

3public class Test

**4** {

**5** public static String sum( int num1, int num2 )

**6** {

**7** return String.format( "%d + %d = %d", num1, num2, ( num1 + num2 ) );

**8** }

**9**

**10** public static void main( String args[] )

**11** {

**12** int number1;

**13** int number2;

**14**

**15** try

**16** {

**17** number1 =

**18** Integer.parseInt( JOptionPane.showInputDialog( "Enter an integer: " ) );

**19**

**20** number2 = Integer.parseInt(

**21** JOptionPane.showInputDialog( "Enter another integer: " ) );

**22**

**23** System.out.println( sum( number1, number2 ) );

**24** }

**25** catch ( NumberFormatException numberFormatException )

**26** {

**27** System.out.println( numberFormatException.toString() );

**28** }

**29** } // end main method

**30** } // end class Test

*Your answer:*