Objectives:
1. To review the solutions to the Midterm Exam
2. To understand the Testing Phase of the Software Life Cycle
3. To identify problems present in software through the process of debugging
4. To understand the process of exception handling

Reading Assignment:
1. Nell/Chip/Mark, Chapter 9

Concepts:
1. Testing
2. Debugging
3. Exception Handling
Midterm Solution Review

1) Which of these data types are signed integrals?
   B. byte, short, int, and long
   These are the Integral data types used for signed integer representation

2) What is the size of an int?
   C. 32 bits

3) Consider the following line of code:
   char x[] = new char[10];
   After execution, what is value of x[0]?
   B. ' '

4) What are the two integral data types for representing floating point numbers?
   Float and Double

5) True or False: All Operands are evaluated left to right.
   True

6) Consider the following line of code:
   int x = 7;
   int y = 4;
   int z = 7 / 4;
   After execution what is value of z?
   C. 1

7) Please select the true statement for ! operator
   A. The ! operator inverts the value of a boolean expression

8) Which two of these arithmetic operations can result in an ArithmeticException?
   B. Division: /
   C. Modulo: %

9) Please select operators which perform bit shifts of the binary representation of the integral types
   A. <<
   B. >>
   C. >>>

10) A class variable is declared using following modifiers
    D. static

11) True or False: An unary operator operates on a single value?
    True

12) What type of loop is guaranteed to be executed once?
    Do loop

13) If word1 is “Bob” and word2 is “Barb”, what does the expression below return?
    0 < word1.compareTo(word2)?
    True

14) Write a statement that sets boolean variable ok to true if the value of the variable number is less than or equal to 10.
    ok = (number <= 10);
15) True or False? Comparing strings using == and != is legal, but you do not get the answer you expect.
   True
   Explanation: \( s1 == s2 \) returns true if \( s1 \) and \( s2 \) reference the same String object

16) Write a count-controlled loop that prints your name 1000 times.
   assuming your name is bob:
   ```java
   int index = 1;
   while (index <= 100)
   {
       System.out.println("bob");
       index = index + 1;
   }
   ```

17) Write a Java if statement that assigns the larger of the values num1 and num2 to the variable larger:
   ```java
   if (num1 > num2)
       larger = num1;
   else
       larger = num2;
   ```

18) Design a program to calculate \( a^n \). You can demonstrate your design in pseudocode or Java.
   Pseudocode Form:
   If \( n == 0 \), then \( a^n = 1 \)
   If \( n > 0 \), then \( a^n = a \cdot \ldots \cdot a \)
   \( (a \) is used as a factor \( n \) times)\)
   If \( n < 0 \), then \( a^n = 1/a^{-n} \)
   Algorithm Form:
   result = 1.0;
   count = 1;
   while (count <= Math.abs(n))
   {
       result = result * a;
       count++;
   }
   // result holds value of \( a^n \)
   // if \( n < 0 \), \( a^n = 1/result \)
   // if \( n >= 0 \), \( a^n = result \)

1. Testing Phase of Software Life Cycle
   - Why is testing done?
     - Testing finds bugs that the programmer did not expect
     - It is always best to design a good solution in the first place than to try to patch up a bad design
     - A whole course is needed to fully understand what is necessary to comprehensively test a program.

2. Debugging
   - Using the Debugger
   - Understanding Error messages

3. Exception Handling
   - A exception is an unusual situation that occurs while the program is executing
   - A good programmer will recognize exceptions early in the design phase and fix problems before they occur.
   - Steps:
     - Define error conditions
ii. Enclose code containing possible error (try)
iii. Alert system if the error occurs (throw)
iv. Handle error if it occurs (catch)
Objectives:
1. To practice exception handling

Student Activities:
1. Add Exception Handling with try / throw / catch to the code below to prevent an illegal division by zero

```java
public class Divide {
    public Divide(int a, int b) {
        try {
            answer = a / b;
        } catch (IOException except) {
            System.out.println("Illegal Division by Zero");
        }
    }

    public Float getAnswer() {
        return answer;
    }

    private float answer;
}
```

Lab Solution:

```java
public class Divide {
    public Divide(int a, int b) {
        try {
            answer = a / b;
        } catch (IOException except) {
            System.out.println("Illegal Division by Zero");
        }
    }

    public Float getAnswer() {
        return answer;
    }

    private float answer;
}
```