Exercise Four: Exception, Thread

1. (20 Points) NumberFormatException

Write the `bin2Dec(String binaryString)` method to convert a binary string into a decimal number. Implement the `bin2Dec` method to throw a `NumberFormatException` if the string is not a binary string. Write a test program.

2 (20 Points) ArrayIndexOutOfBoundsException

Write a program meets the following requirements：

(1) Create an array with `100` randomly chosen integers.

(2) Prompts the user to enter the index of the array, then displays the corresponding element value. If the specified index is out of bounds, display the message **Out of Bounds**.

(3) Write a test program.

3 (60 Points) Thread Exercise

You are asked to do some exercises with Java multithreading. You can use SumThread example as a starting point, although it might be more useful to start from scratch to make sure you understand all of the steps.

(1) Max value

Write a program called `MaxValue.java` that finds the maximum value in an array of ints using 4 threads. Your main should be similar as the one in the above `SumThread example`, though you should construct your array of random numbers instead of increasing numbers. You may assume in your threaded code that the array has at least 4 elements.

(2) Reverse hello

Write a program called `ReverseHello.java` that creates a thread (let's call it Thread 1). Thread 1 creates another thread (Thread 2); Thread 2 creates Thread 3; and so on, up to Thread 50. Each thread should print "Hello from Thread <num>!", but you should structure your program such that the threads print their greetings in reverse order.

(3) Shared counter

Write a program called `SharedCounter.java` in which 10 threads each increment a shared int counter 10 times. When all the threads have finished, print the final value of the counter. If the initial value is zero, do you always get 100? Arrange for your code to sometimes print the wrong answer. (Hint: try using some well-placed calls to `Thread.yield()` or `Thread.sleep()`.)

4 (20 points) A multi-thread program

(1) Task 1: Mutual exclusion

Consider the `UnsafeCounter.java` that implements a shared counter. Write a program `UnsafeCounterDemo` where two threads increment one million times each a unique shared instance of `UnsafeCounter`. Write the final value of the counter on the standard output stream. Save your program in a file named `UnsafeCounterDemo.java`.

The expected result of the program is: two million. However, it is unlikely that you will obtain this result in practice. Why is it that the observed behavior is different than the expected one?

(2) Task 2: Thread-safe counter

Modify your code from the first question and write a thread-safe counter. Please save your code in the files `Counter.java` and `CounterDemo.java`. Verify that the value of the counter at the end is the expected value of two million.