CS 1110 Assignment 5

1. Meeting a Specification

Name: `Book.java`

Often times when you create a class, it will be part of a larger project. In those cases, you will often be given some documentation or specification, such as a UML diagram, from which you are to create your class. In this exercise, your task is to create a class named Book that meets the specification illustrated in the UML diagram. Use the attached Java file (TestBook.java) to test your class. You should not need to modify anything inside of TestBook.java. Note that the items in your class need to be named exactly as indicated so they will meet the specification and work with TestBook.java.

Book
title: String
author: String
isbn: long
Book()
Book(title: String, author: String, isbn: long)
getTitle(): String
setTitle(title: String)
getAuthor(): String
setAuthor(author: String)
getISBN(): long
setISBN(isbn: long)
toString(): String

Sample Run (from TestBook.java):

```
Default constructor and getters:
Title: The Art of War
Author: Sun Tzu
ISBN: 1111111111

Result of toString():
"The Art of War" by Sun Tzu (1111111111)

Using the setters and toString():
"The Book of Five Rings" by Miyamoto Musashi (2222222222)

Using other constructor:

Using the getters:
Title: The C Programming Language
Author: Kernighan and Ritchie
ISBN: 3333333333
```

2. Big Primes

Name: `BigPrimes.java`

A prime number is a number that is only divisible by 1 and itself. Write a program that gets a starting number from the user and an ending number and displays all prime numbers in that range. Note that your program needs to use BigInteger.

Sample Runs:

```
Enter start of range: 1
Enter end of range: 20

Primes:
2
3
5
7
11
13
17
19
```

```
Enter start of range: 6189700196426901374495621l0
Enter end of range: 618970019642690137449562120

Primes:
618970019642690137449562111
```

3. Mersenne Primes

Name: `MersennePrimes.java`

A Mersenne prime is a prime number that can be written in the following form:

$$2^p - 1$$

where p is a prime number. Note that not all numbers that can be written in that form are prime. Your task is to write a program that gets a number from the user and displays all Mersenne primes such that p is less than or equal to the number entered. Hint: Use BigInteger's isProbablePrime() method.

Sample Run:

```
Enter max p: 100

p           Mersenne Prime
2           3
3           7
5           31
7           127
13          8191
17          131071
19          524287
31          2147483647
61          2305843009213693951
89          618970019642690137449562111
```

4. Clean Strings

Name: `CleanStrings.java`

In an imaginary society, the following words are considered extremely offensive: "help", "dime". "ask", "beach", "ship", and "fish". Your task is to write a program that gets text from the user and replaces any occurrences of the offensive words with asterisks and display the censored results. You may assume all the letters will be lowercase. Hint: Use replaceAll().

Sample Run:

```
Enter text: what the help?

Cleaned text: what the ****?
```

```
Enter text: go fish yourself!

Cleaned text: go **** yourself!
```

5. Words and Characters Count

Name: `WCCount.java`

Write a program that gets a line of text from the user and displays how many words occurred in the text and how many non-whitespace characters appeared. Hint: Use split().

Sample Run:

```
Enter text: Hello goodbye

Word count: 2
Character count: 12
```

```
Enter text: The quick brown fox jumped over the lazy dogs.

Word count: 9
Character count: 38
```

6. Big Pi (Bonus)

Name: `BigPi.java`

The magic number π can be estimated with the following formula:

$$\pi = 4\left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \ldots + \frac{(-1)^{i+1}}{2i-1}\right)$$

Write a program that gets a number from the user for how many iterations (i.e., number of fractions) and how many decimal places and displays the resulting estimate of π. Use BigDecimal so that your program can support an arbitrary amount of precision. Hint: The number of decimal places stored in the BigDecimal can be specified in the divide() method of BigDecimal (i.e., *dividend*.divide(*divisor*, *decimalPlaces*, BigDecimal.ROUND_UP)). Note that for large amounts of iterations and decimal places, your program could take a long time to run.

Sample Runs:

```
Enter number of iterations: 100
Enter number of decimal places: 5

Estimate: 3.13164
```

```
Enter number of iterations: 100
Enter number of decimal places: 20

Estimate: 3.131592903558552764160
```

```
Enter number of iterations: 1000000
Enter number of decimal places: 100

Estimate: 3.14159165358979323871264338327919038419717035250010581556478834235715332034804914389171886837074201920
```

7. Big e (Bonus)

Name: `Bige.java`

e is known as Euler's number and is another magic number like π. The numerical value of e is approximately 2.71828. The following formula can produce an estimate of e:

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \cdots + \frac{1}{i!}$$

Write a program that gets a number from the user for how many iterations and how many decimal places and displays the resulting estimate of e. Note that for large amounts of iterations and decimal places, your program could take a long time to run.

Sample Runs:

```
Enter number of iterations: 100
Enter number of decimal places: 5

Estimate: 2.71921
```

```
Enter number of iterations: 100
Enter number of decimal places: 20

Estimate: 2.71828182845904523623
```

```
Enter number of iterations: 1000000
Enter number of decimal places: 100

Estimate:
2.71828182845904523536028747135266249775724709369995957496696762772407
66303535475945713821785252664235
```