## How to run:

1. Clone the repository:
   a. git clone " https://github.com/rkand4/project-euler.git"
   URL of the repository (https://github.com/rkand4/project-euler)
2. Go to the above location and then run ( DO NOT MISS THIS ):
   b. **mvn clean test**
3. Import this project into Intellij now.
3. The source code should be under "/org/rakesh/projecteuler/**"
   a. problem2
   b. problem6
   c. problem16
4. Tests are written in "**Spock**" groovy library allows to write data driven tests and Behavioral Driven Tests (BDD) Style. Can be located under:
   a. "src/test/groovy/org/rakesh/projecteuler/**"
5. "Log4j" logging framework has been used to log output.

## Problems Choosen:

1. Problem2 (EvenFibonacciNumbers.java): Even Fibonacci numbers
2. Problem6 (SumSquareDifference.java) : Sum square difference
3. Problem16 (PowerDigitSum.java) :Power digit sum

## Problem2 (Even Fibonacci numbers Sum):

1. Why I choose this problem:
   a. Fibonacci numbers are so interesting for me always and I used to ask people when I used to interview other people. I decided to take this up and solve it.

2. Process followed:
   a. Fibonacci number generation code is in place which works by storing the previous sequence terms and the next term is produced based on the previous two terms.
   2. The current term is checked for even using bit operations by AND'ing with 1.
   (number & 1) == 0 ( Even Number)
   (number & 1) != 0 (Odd Number)
   3. Running sum of the above even numbers and making sure the iteration doesn't exceed the upper bound of the sequence asked for.
   4. "int" datatype will not overflow and fit the sum till 4-Million upper bound. I analyzed this and in this context I choose int.

3. I haven't looked at any resources as it is straight forward.

4. How to Run Problem2 Main:
   a. Either import this cloned project in IntelliJ and run the Main method in "EvenFibonacciNumbers.java"
   b. Make sure you go to target folder of this maven project after "mvn clean package".

Run this EvenFibonacciNumbers.class file.

5. Sample Output of Problem2 (EvenFibonacciNumbers.java):
2017-10-22 16:59:22 INFO  EvenFibonacciNumbers:24 - Execution time of EvenFibonacciNumbers: 0.00683(ms)
2017-10-22 16:59:22 INFO  EvenFibonacciNumbers:28 - Sum of even fibonacci numbers until the bound limit is: 4613732

## Problem6 (SumSquareDifference):

1. Why I choose this problem:
   a. I found it interesting... nothing good reason per se.

2. Process followed:
   a. Loop through till the upper bound natural number inclusive.
   b. During looping I maintain a running sum of natural numbers and also have a running sum of squares of individual numbers.
   c. Calculate the square of the running sum maintained and compute the difference between this and the running sum of the squares of the numbers and return it.
3. I haven't looked at any resources as it is straight forward.

4. How to Run Problem6 Main:
   a. Either import this cloned project in IntelliJ and run the Main method in "SumSquareDifference.java"
   b. Make sure you go to target folder of this maven project after "mvn clean package".
   Run this SumSquareDifference.class file.

5. Sample Output of Problem6 (SumSquareDifference.java):
2017-10-22 17:08:24 INFO  SumSquareDifference:25 - Execution time of SumSquareDifference: 0.00363(ms)
2017-10-22 17:08:24 INFO  SumSquareDifference:29 - Difference of (Square of sum - Sum of Squares ) is : 25164150

## Problem16 (Power digit sum ):

1. Why I choose this problem:
   a. Initially I thought just a simple Left Shift of 1 the times of the power will give result.
   Then I realized the overflow scenario and found it challenging to look for BigInteger and took over this problem.

2. Process followed:
   a. BigInteger is used for Left Shifting BigInteger of value 1 by that many times of the input power.
   b. The result from the left shift is sent to another method which converts it to a String format.

c. Now we loop through the string length and add the numbers and keep a running count of the sum
and return the result. "int" result type is assumed for the returned sum keeping in mind the range of the power in this problem context (1000).

3. I haven't looked at any resources as it is straight forward.

4. How to Run Problem16 Main:
a. Either import this cloned project in IntelliJ and run the Main method in "PowerDigitSum.java"
b. Make sure you go to target folder of this maven project after "mvn clean package".
Run this PowerDigitSum.class file.

5. Sample Output of Problem16 (PowerDigitSum.java):
2017-10-22 17:17:40 INFO  PowerDigitSum:25 - Execution time of PowerDigitSum: 1.45300(ms)
2017-10-22 17:17:40 INFO  PowerDigitSum:28 - Total Digits (Sum of 2 raised to the Power Thousand) is: 1366