```java
package moduleExercises;

public class Exercises {

   public static void main(String[] args) {
       System.out.println("Sum of multiples of 3 and 5 less than 1000:
"+multiplesOf3and5());
       System.out.println("Sum of even Fibonacci Numbers less than 4 million: " +
evenFibonacciNumbers());
       System.out.println("Largest prime factor of 600851475143l: " +
largestPrimeFactor());
       System.out.println("Largest palindrome that is a product of two three-digit
numbers: "+largestPalindrome());
       System.out.println("Smallest positive number evenly divisible by all numbers
from 1 to 20: " + smallestMultiple());
       System.out.println("Difference between sum of the squares of first one hundred
natural numbers and the square of the sum: " + sumSquareDifference());
       System.out.println("10001st prime integer: " + prime_10001st());
       System.out.println("Largest product between 13 adjacent digit in the 1000 digit
number: " + largestProduct());
       System.out.println("Product of the pythagorean triplet: " +
specialPythagoreanTriplet());
       System.out.println("Sum of primes under 2 million: " + summationOfPrimes());
   }

   public static int multiplesOf3and5(){
       int sumThree = 0;
       int sumFive = 0;
       for(int i = 0; i<1000; i+=3) {
           sumThree+=i;
       }
       for(int i = 0; i<1000; i+=5) {
           if(i%3!=0) {
               sumFive+=i;
           }
       }
       return sumThree + sumFive;
   }

   public static int evenFibonacciNumbers(){
       int fibSum = 0;
       int val1=0;
```

```
    int val2=1;

    int tempSum=0;
    while(tempSum<4*Math.pow(10, 6)) {
        if(tempSum%2==0) {
            fibSum+=tempSum;
        }
        tempSum = val1 + val2;
        val1 = val2;
        val2 = tempSum;
    }
    return fibSum;
}

public static int largestPrimeFactor() {
    long factoredNum = 600851475143l;
    int lpf = 1;
    for(int i = 2; i<=factoredNum; i++) {
        if(factoredNum%i==0) {
            factoredNum/=i;
            lpf=(i>lpf)?i:lpf;
        }
    }
    return lpf;
}

public static int largestPalindrome() {
    for(int i = 9; i>0;i--) {
        for(int j=9; j>=0; j--) {
            for(int k=9; k>=0;k--) {
                int palindrome = generatePalindrome(i, j, k);
                for(int n=999; n>Math.sqrt(palindrome); n--) {
                    if(palindrome%n==0) {
                        return palindrome;
                    }
                }
            }
        }
    }
    return 0;
}
public static int generatePalindrome(int d1, int d2, int d3) {
```

```
        return (int)
(d1*Math.pow(10,5)+d2*Math.pow(10,4)+d3*Math.pow(10,3)+d3*Math.pow(10,2)+d2*Math.pow(1
0,1)+d1);
    }

    public static int smallestMultiple() {
        mainLoop: for(int i = 20*19*17*13*11*7*3; i<Integer.MAX_VALUE;i+=20) {
            for(int j=20;j>0;j--) {
                if(i%j!=0) {
                    continue mainLoop;
                }
            }
            return i;
        }
        return 0;
    }
    public static int sumSquareDifference() {
        int totSum = 0;
        int sumOfSquares  = 0;
        for(int i=1;i<=100;i++) {
            totSum+=i;
            sumOfSquares += Math.pow(i, 2);
        }
        return (int)Math.pow(totSum, 2)-sumOfSquares;
    }

    public static int prime_10001st() {
        int primeInd = 0;
        int currentPrime = 0;
        int num = 2;
        while(primeInd<10001) {
            boolean isPrime = true;
            for(int i = 2; i<=Math.sqrt(num); i++) {
                if(num%i==0) {
                    isPrime = false;
                    break;
                }
            }
            if(isPrime) {
                primeInd++;
                currentPrime=num;
            }
```

```
            num++;
        }
        System.out.println();
        return currentPrime;
    }

    public static long largestProduct() {
        String bigNum = "73167176531330624919225119674426574742355349194934"
                + "96983520312774506326239578318016984801869478851843"
                + "85861560789112949495459501737958331952853208805511"
                + "12540698747158523863050715693290963295227443043557"
                + "66896648950445244523161731856403098711121722383113"
                + "62229893423380308135336276614282806444486645238749"
                + "30358907296290491560440772390713810515859307960866"
                + "70172427121883998797908792274921901699720888093776"
                + "65727333001053367881220235421809751254540594752243"
                + "52584907711670556013604839586446706324415722155397"
                + "53697817977846174064955149290862569321978468622482"
                + "83972241375657056057490261407972968652414535100474"
                + "82166370484403199890008895243450658541227588666881"
                + "16427171479924442928230863465674813919123162824586"
                + "17866458359124566529476545682848912883142607690042"
                + "24219022671055626321111109370544217506941658960408"
                + "07198403850962455444362981230987879927244284909188"
                + "84580156166097919133875499200524063689912560717606"
                + "05886116467109405077541002256983155200055935729725"
                + "71636269561882670428252483600823257530420752963450";
        int adjLen = 13;
        long maxProduct = 0;
        for(int startInd = 0; startInd+adjLen<=bigNum.length(); startInd++) {
            long product = 1;
            for(int i = 0; i<adjLen; i++) {
                product*=Integer.parseInt(bigNum.substring(startInd +i,startInd +i+1));
            }
            if(product>maxProduct) {
                maxProduct = product;
            }
        }
        return maxProduct;
    }
```

```java
    public static int specialPythagoreanTriplet() {
        int a = 1;
        int b = 1;
        double c = getC(a,b);
        while(a+b+c!=1000) {
            b=a;
            while(a+b+getC(a,b)<=1000) {
                b++;
                if(a+b+getC(a,b)==1000) {
                    return a*b*(int)(getC(a,b));
                }
            }
            a++;
        }
        return 0;
    }
    public static double getC(int a, int b) {
        return Math.sqrt(Math.pow(a, 2) + Math.pow(b, 2));
    }

    public static long summationOfPrimes() {
        int num = 2;
        long sum = 0;
        while(num<2*Math.pow(10,6)) {
            boolean isPrime = true;
            for(int i = 2; i<=Math.sqrt(num); i++) {
                if(num%i==0) {
                    isPrime = false;
                    break;
                }
            }
            if(isPrime) {
                sum+=num;
            }

            num++;
        }
        System.out.println(sum);
        return sum;
    }
}
```