FIRST EIGHT DIGIT PRIME IN CONSECUTIVE DIGITS OF EULER CONSTANT

Computing e

The number e is the base of natural logarithms and one of the most important numbers. The first few digits of e are 2.718281828459045235360..., but the constant is irrational, meaning that its digits never fall into a repeating pattern. The constant has been computed to an absurd number of digits over the years, but although you can find lists of the first few million digits of e on the web, for this assignment you must compute it directly. There are many algorithms for generating the digits of e, but for this assignment you should use the following simple formulation for computing e:

$$e = \sum_{i=0}^{\infty} \left(\frac{1}{i!} \right)$$

or, in other words

$$e = \frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \frac{1}{5!} + \cdots$$

The more terms you calculate, the closer you will get to the true value of \boldsymbol{e} . You will only arrive at the exact value of \boldsymbol{e} if you carry on adding up the sequence forever.

In this coding exercise you will develop a class, EulerPuzzle, that has two methods.

The first method *BigDecimal calculateE()* will calculate e constant to 10.000-digit precision. The second method *int findPrime(BigDecimal e)* that finds the first 8-digit prime found in its fractional digits and returns the starting index otherwise return -1. The index of the first digit of fraction is 0. You can develop other helper methods and fields, if you need.

Your *BigDecimal calculateE()* method should iterate for first *2000* terms.

1. The incomplete EulerPuzzle class is as below.

```
import java.math.BigDecimal;
import java.math.BigInteger;
import java.math.MathContext;

public class EulerPuzzle {
    public final int NUM_OF_DIGITS = 10000;
    public final int NUMBER_OF_TERMS = 2000;

    public static void main(String[] args) {
        EulerPuzzle eulerPuzzle = new EulerPuzzle();
        System.out.println(eulerPuzzle.calculateE());
        System.out.println(eulerPuzzle.findPrime(eulerPuzzle.calculateE()));
    }

    public BigDecimal calculateE() {
        return e;
    }

    public int findPrime(BigDecimal e) {
```

```
return -1;
}
```

2. The output of the above code should be as below

```
6145914429364514286171585873397467918975712119561873857836447584484
2355558105002561149239151889309946342841393608038309166281881150371
5284967059741625628236092168075150177725387402564253470879089137291
7228286115159156837252416307722544063378759310598267609442032619242
8531701878177296023541306067213604600038966109364709514141718577701
4180606443636815464440053316087783143174440811949422975599314011888
6833148328027065538330046932901157441475631399972217038046170928945
7909627166226074071874997535921275608441473782330327033016823719364
8002173285734935947564334129943024850235732214597843282641421684878
7216733670106150942434569844018733128101079451272237378861260581656
6805371439612788873252737389039289050686532413806279602593038772769
7783792868409325365880733988457218746021005311483351323850047827169
3762180049047955979592905916554705057775143081751126989851884087185
6402603530558373783242292418562564425502267215598027401261797192804\\
7139600689163828665277009752767069777036439260224372841840883251848
7704726384403795301669054659374616193238403638931313643271376888410
2681121989127522305625675625470172508634976536728860596675274086862
7407912856576996313789753034660616669804218267724560530660773899624
2183408598820718646826232150802882863597468396543588566855037731312
9658797581050121491620765676995065971534476347032085321560367482860
8378656803073062657633469774295634643716709397193060876963495328846
8336130388294310408002968738691170666661468000151211434422560238744
7432525076938707777519329994213727721125884360871583483562696166198
0572526612206797540621062080649882918454395301529982092503005498257
0433905535701686531205264956148572492573862069174036952135337325316
6634546658859728665945113644137033139367211856955395210845840724432
3835586063106806964924851232632699514603596037297253198368423363904
6321367101161928217111502828016044880588023820319814930963695967358
3274202498824568494127386056649135252670604623445054922758115170931
4921879592718001940968866986837037302200475314338181092708030017205
9355305207007060722339994639905713115870996357773590271962850611465
1483752620956534671329002599439766311454590268589897911583709341937
0441155121920117164880566945938131183843765605550693145459135564072
8368935332264410700718679891276367186588005862784550675208735184499
2887101879010014786606240518189354048965161319709557524405191330628
9401087471632625922973856045223904389501212826286314635188732749696
8544914324981986263981242791736974874989749776310235375240259534653
6437750990004966067509887678050097321115265662526300268581800541955
7275814723373848796847168063785295599819372455000077877165241863693
7320131311620023309719206618941672543435946238385816918451321703668
3577206421893386250948873330335712730558124732731113699364198871732
4511764850188431408353088800141080656959409687280107268968171377885
6674444698095976957570700555182929076417229205295309613364618982296
5047427378192105574761803228251021227590504328859953715615437478639
8331418282609627383147016676045460744517591412207369774733373338611
6851410132154888797843920786071090316397819198492776299751628536356
0986273707033109823663344149168803455896481548925476029265123665317
4202782838713137828459643232485571341324835500891558663971615620072
3368299063876173901613372362015970600023394574712448645838435550656
4377902627877471159036148238088636225766789663264845108423438397903
0267243354670305906374709375240137030300232697467282591604520396368
2401651055060304464836805292668740274019313036105920116073597892779
3154612982535169829334709055865747782324276768461989375102243023518
8216792385048846733994013347454074517826148209794707292467662938532
0206041083039131464966693851516042629653421800566728897909130598206\\
0180502712166245704357737172394848775047733953895796085484296500268
0290382240374960548966436880857404025261785270934171655171327754987
2560270836627106149348180674541875915862997261896139547206862757466
2016965228008648167736092963507635948391746864833755222810844313591
5589268562353417591363311706054442650809841747239931810933128482603
2555509748160417183379455241759443783632178837688919496436449678272
3169163572824558681565703735466586538583830557379126832638801920112
```