In this problem you will first implement the reduce method in the Fraction class. The reduce method simplifies the numerator and denominator in the given fraction.

Special note: This Fraction class is also used in the Marbles problem.

The fraction 2/7 is represent by following code: new Fraction(2, 7);

And the fraction 13/31 is represent by following code: new Fraction(13, 31);

The following code shows the results of the reduce method.

|  |  |
| --- | --- |
| The following code | Returns |
| Fraction temp = new Fraction(2\*5\*7\*7, 2\*3\*7); |  |
| temp.reduce(); |  |
| System.out.print(**temp.getNumerator()**); | 35 = 5\*7 |
| System.out.print(**temp.getDenominator()**); | 3 |

The following code shows the results of the reduce method.

|  |  |
| --- | --- |
| The following code | Returns |
| Fraction temp = new Fraction(0, 2\*5\*7\*7\*2\*3\*7); |  |
| temp.reduce(); |  |
| System.out.print(**temp.getNumerator()**); | 0 |
| System.out.print(**temp.getDenominator()**); | 1 |

The rest of this problem is focused on the Farey Sequence. The Farey sequence of order n (denoted Fn) is the sequence of all completely reduced fractions between 0 and 1 which when in lowest terms has denominators less than or equal to n, arranged in order of increasing size.

Each Farey sequence starts with the value 0, denoted by the fraction , and ends with the value 1, denoted by the fraction . Below are examples of F1 through F5.

**Examples:** 

You will implement two static methods in the FareySequence class. The first method is the generateOrder(int n) method (n <= 25). This method returns a List<Fraction> containing all completely reduced fractions of the form:



The following code shows the results of the generateOrder method.

|  |  |
| --- | --- |
| The following code | Returns |
| List<Fraction> sol = FareySequence.generateOrder(5); |  |
| System.out.print(**sol.get(0))**; | new Fraction(0, 1) == |
| System.out.print(**sol.get(1))**; | new Fraction(1, 5) == |
| System.out.print(**sol.get(2)**); | new Fraction(1, 4) == |
| System.out.print(**sol.get(3)**); | new Fraction(1, 3) == |
| System.out.print(sol.get(4)); | new Fraction(2, 5) == |
| System.out.print(sol.get(5)); | new Fraction(1, 2) == |
| System.out.print(sol.get(6)); | new Fraction(3, 5) == |
| System.out.print(sol.get(7)); | new Fraction(2, 3) == |
| System.out.print(sol.get(8)); | new Fraction(3, 4) == |
| System.out.print(sol.get(9)); | new Fraction(4, 5) == |
| System.out.print(sol.get(10)); | new Fraction(1, 1) == |

A second example follows on the next page.

The following code shows the results of the generateOrder method.

|  |  |
| --- | --- |
| The following code | Returns |
| List<Fraction> sol = FareySequence.generateOrder(7); |  |
| sol.get(0)); | new Fraction(0, 1) == |
| sol.get(1)); | new Fraction(1, 7) == |
| sol.get(2)); | new Fraction(1, 6) == |
| sol.get(3)); | new Fraction(1, 5) == |
| sol.get(4)); | new Fraction(1, 4) == |
| sol.get(5)); | new Fraction(2, 7) == |
| sol.get(6)); | new Fraction(1, 3) == |
| sol.get(7)); | new Fraction(2, 5) == |
| sol.get(8)); | new Fraction(3, 7) == |
| sol.get(9)); | new Fraction(1, 2) == |
| sol.get(10)); | new Fraction(4, 7) == |
| sol.get(11)); | new Fraction(3, 5) == |
| sol.get(12)); | new Fraction(2, 3) == |
| sol.get(13)); | new Fraction(5, 7) == |
| sol.get(14)); | new Fraction(3, 4) == |
| sol.get(15)); | new Fraction(4, 5) == |
| sol.get(16)); | new Fraction(5, 6) == |
| sol.get(17)); | new Fraction(6, 7) == |
| sol.get(18)); | new Fraction(1, 1) == |

Farey sequences have several interesting properties. One is that they give an interesting way of getting rational approximations to irrational numbers.

In order to find a rational approximation to an irrational number using Farey fractions you need to pick the interval between Farey fractions that contains the target number and narrow the interval at each step. If the target number is between  and  then at the next step you have to decide which of contains the target number.

The following table shows the results of carrying out this process for finding the approximations to within 0.001 , and for pi (Math.PI) with the given initial left bound and right bound.

**(The bound in bold indicates the updated bound.)**

|  |  |  |
| --- | --- | --- |
| Irrational number: Math.sqrt(2)/2  With leftBound: new Fraction(0, 1)  And rightBound: new Fraction(1, 1) |  | Irrational number: Math.PI  With leftBound: new Fraction(3, 1)  And rightBound: new Fraction(16 ,5) |
| **leftBound: new Fraction(1, 2)**  rightBound: new Fraction(1, 1) |  | leftBound: new Fraction(3, 1)  **rightBound: new Fraction(19, 6)** |
| **leftBound: new Fraction(2, 3)**  rightBound: new Fraction(1, 1) |  | leftBound: new Fraction(3, 1)  **rightBound: new Fraction(22, 7)** |
| leftBound: new Fraction(2, 3)  **rightBound:** **new Fraction(3, 4)** |  | **leftBound: new Fraction(25, 8)**  rightBound: new Fraction(22, 7) |
| leftBound: new Fraction(2, 3)  **rightBound: new Fraction(5, 7)** |  | **leftBound: new Fraction(47, 15)**  rightBound: new Fraction(22, 7) |
| **leftBound: new Fraction(7, 10)**  rightBound: new Fraction(5, 7) |  | **leftBound: new Fraction(69, 22)**  rightBound: new Fraction(22, 7) |
| **leftBound: new Fraction(12, 17)**  rightBound: new Fraction(5, 7) |  | **leftBound: new Fraction(91, 29)**  rightBound: new Fraction(22, 7) |
| leftBound: new Fraction(12, 17)  **rightBound: new Fraction(17, 24)** |  | **leftBound: new Fraction(113, 36)**  rightBound: new Fraction(22, 7) |
| **leftBound: new Fraction(29, 41)**  rightBound: new Fraction(17, 24) |  | **leftBound: new Fraction(135, 43)**  rightBound: new Fraction(22, 7) |
| **Final answer**: new Fraction(29, 41) |  | **leftBound: new Fraction(157, 50)**  rightBound: new Fraction(22, 7) |
|  |  | **leftBound: new Fraction(179, 57)**  rightBound: new Fraction(22, 7) |
|  |  | leftBound: **new Fraction(201, 64**)  rightBound: new Fraction(22, 7) |
|  |  | **Final answer**: new Fraction(201, 64) |

Continue on next page for final method to implement ☺

The final method to implement is the getApproximation(double num, Fraction leftBound, Fraction rightBound) method. This method returns the rational approximation (as a Fraction) of num using the given leftbound and rightbound.

The following code shows the results of the getApproximation method.

|  |  |
| --- | --- |
| The following code | Returns |
| FareySequence.getApproximation(Math.sqrt(2)/2.,  new Fraction(0, 1),  new Fraction(1, 1))) | new Fraction(29, 41) |
| FareySequence.getApproximation( Math.PI,  new Fraction(3, 1),  new Fraction(16, 5)) | new Fraction(201, 64) |