In this problem you will implement one method in the Fraction class and you will implement three methods in the Marble class.

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

The method to be implemented in the Fraction class is the reduce method. After a Fraction has been reduced, both the numerator and denominator have no common factors except 1. If a Fraction has a numerator equal to 0, the denominator should be set to 1.

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(); |  |
| temp.getNumerator() | 35 = 5\*7 |
| temp.getDenominator() | 3 |

The following code shows the results of the getRoll method with a six sided die.

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

When a Marble object is created it is passed a List<Integer>. This List represents a bag of randomly numbered marbles. In the Marble class you are to implement three methods.

The first method to implement is the getMinSum(int numMarbles). getMinSum returns the smallest possible sum of numMarbles randomly picked marbles without replacement.

The following code shows the results of the getMinSum method.

|  |  |
| --- | --- |
| The following code | Returns |
| List<Integer> bag = new ArrayList<Integer>();  bag.add( new Integer(7));  bag.add( new Integer(3));  bag.add( new Integer(0));  bag.add( new Integer(3));  bag.add( new Integer(11));  bag.add( new Integer(6));  bag.add( new Integer(9));  bag.add( new Integer(9));  bag.add( new Integer(5));  bag.add( new Integer(1));  Marbles m = new Marbles(bag); |  |
| m.getMinSum(3) | 4 = 0+1+3 |

The second method to implement is the getMaxSum(int numMarbles). getMaxSum returns the largest possible sum of numMarbles randomly picked marbles without replacement.

The following code shows the results of the getMaxSum method.

|  |  |
| --- | --- |
| The following code | Returns |
| List<Integer> bag = new ArrayList<Integer>();  bag.add( new Integer(7));  bag.add( new Integer(3));  bag.add( new Integer(0));  bag.add( new Integer(3));  bag.add( new Integer(11));  bag.add( new Integer(6));  bag.add( new Integer(9));  bag.add( new Integer(9));  bag.add( new Integer(5));  bag.add( new Integer(1));  Marbles m = new Marbles(bag); |  |
| m.getMaxSum(3) | 29 = 9+9+11 |

The third method to implement is the getProbability(int numMarbles, int target). getProbability returns the probability (as a Fraction) that the sum of numMarbles randomly selected marbles, chosen without replacement equals the parameter target. The Fraction returned must be in reduced form.

The following code shows the results of the getProbability method.

|  |  |
| --- | --- |
| The following code | Returns |
| List<Integer> bag = new ArrayList<Integer>();  bag.add( new Integer(7));  bag.add( new Integer(3));  bag.add( new Integer(0));  bag.add( new Integer(3));  bag.add( new Integer(11));  bag.add( new Integer(6));  bag.add( new Integer(9));  bag.add( new Integer(9));  bag.add( new Integer(5));  bag.add( new Integer(1));  Marbles m = new Marbles(bag);  Fraction f = m.getProbability (3, 20); |  |
| f.getNumerator() ); | 1 |
| f.getDenominator() | 20 |