Cook Your Lasgna

https://exercism.org/tracks/java/exercises/lasagna/solutions/shaz-ahammed

Two fer

https://exercism.org/tracks/java/exercises/two-fer/solutions/shaz-ahammed

Resistor Colour

https://exercism.org/tracks/java/exercises/resistor-color/solutions/shaz-ahammed

Error handling

https://exercism.org/tracks/java/exercises/error-handling/solutions/shaz-ahammed

Linked List

https://exercism.org/tracks/java/exercises/simple-linked-list/solutions/shaz-ahammed

Nth Prime

https://exercism.org/tracks/java/exercises/nth-prime/solutions/shaz-ahammed

RNA transcription

https://exercism.org/tracks/java/exercises/rna-transcription/solutions/shaz-ahammed

Pythagorean triplet

https://exercism.org/tracks/java/exercises/pythagorean-triplet/solutions/shaz-ahammed

import java.util.\*;

public class PythagoreanTriplet {

private int a;

private int b;

private int c;

public PythagoreanTriplet(int a, int b, int c) {

this.a = a;

this.b = b;

this.c = c;

}

public String toString() {

return String.format("(%d, %d, %d)", a, b, c);

}

public boolean equals(Object o) {

if ( this == o ) {

return true;

} else if ( o instanceof PythagoreanTriplet ) {

PythagoreanTriplet pt = (PythagoreanTriplet) o;

return a == pt.a && b == pt.b && c == pt.c;

} else {

return false;

}

}

public static TripletsList makeTripletsList() {

return new TripletsList();

}

public static class TripletsList {

private int n;

private Integer maxFactor = null;

//

// This is the limit of the ratio of legs a, b: a = b

// a < b => a < n/(2 + √2)

//

private static final double PERIMETER\_RATIO\_LIMIT = 1 + 1 + Math.sqrt(2);

public TripletsList withFactorsLessThanOrEqualTo(int maxFactor) {

this.maxFactor = maxFactor;

return this;

}

public TripletsList thatSumTo(int n) {

this.n = n;

if ( maxFactor == null ) {

maxFactor = n;

}

return this;

}

//

// Assuming that a + b + c = n and a² + b² = c², then:

// b + c = n - a

// and

// a² + b² + 2bc + c² = c² + 2bc + c²

// a² + (b + c)² = 2bc + 2c²

// a² + (b + c)² = c(2b + 2c)

//

// a² + (b + c)²

// ------------- = c

// 2(b + c)

//

// Substituting n - a for b + c:

// a² + (n - a)²

// ------------- = c

// 2(n - a)

//

public List<PythagoreanTriplet> build() {

List<PythagoreanTriplet> triplets = new ArrayList<>();

for (int a = 1; a <= Math.floor(n / PERIMETER\_RATIO\_LIMIT); a++) {

int numerator = a \* a + (int) Math.pow(n-a, 2);

int denominator = 2 \* (n - a);

if ( numerator % denominator == 0 ) {

int c = numerator / denominator;

if ( c <= maxFactor ) {

int b = n - a - c;

triplets.add(new PythagoreanTriplet(a, b, c));

}

}

}

return triplets;

}

}

}