Что требуется проверить:

1. Конструктор. Его можно проверить только косвенно через вызовы других методов, так как доступа к переменным, которые он инициализирует у нес нет.( Нет Getters и эти переменные имеют уровень доступа private).
2. public String getMessage() - мы можем протестировать только на то, что выдаётся не null ссылка и что возвращающаяся строка пустая. А также можно протестировать её после вызова метода checkRectangle() с негативными данными (связанный тест).
3. public boolean chechRectangle() – следует протестировать с данными, согласно check list.
4. public int detectedRectangle() – следует протестировать , согласно check list.
5. public double getSquare() – следует протестировать с данными из check list.

Для составления чек листа понадобиться некоторые формальные рассуждения, которые помогут нам определить граничные условия и классы эквивалентности переменных. Эту информацию можно и нужно искать в требованиях и иной документации проекта, однако в нашем случае у нас такой возможности нет, поэтому будем пользоваться своим опытом и здравым смыслом (в тестировании это допустимый подход). Итак, очевидно, что треугольник не может существовать при хотя бы одной стороне равной отрицательному числу или нулю. А также есть условия существования треугольника, согласно которому сумма двух любых сторон должна быть больше третьей стороны. Подытожим:

* Стороны треугольника строго больше нуля;
* Сумма двух любых сторон строго больше третьей.

Check List:

Простые позитивные тесты (малые величины сторон):

* Прямоугольный треугольник с разными катетами (например, Пифагора);
* Равнобедренный прямоугольный треугольник;
* Равнобедренный треугольник (с острым и тупым углом);
* Равносторонний треугольник;
* Разносторонний треугольник.

Простые негативные тесты (на малых величинах):

* Несуществующие треугольники с положительными значениями сторон;
* Несуществующие треугольники с одной из сторон равной нулю;
* Несуществующие треугольники с одной из сторон отрицательной.

Сложные позитивные тесты (на предельных возможных величинах, Double=MaxValue = 1.7E+308):

* Равносторонний треугольник.
* Разносторонний треугольник.

Сложные негативные тесты (на предельных величинах, Double = MaxValue = 1.7E+308):

* Несуществующий треугольник.

Примечание: Проверку на ввод чисел других типов и на поступление в конструктор параметров не соответствующих объявленным я проводить не буду по причине того, что компилятор не пропустит такие вещи сам.

Данные для тестирования я буду брать из открытых источников и сведу их в таблицу вида (округление до 0,01):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Сторона “a” | Сторона “b” | Сторона “c” | Характеристики треугольника (expected) | Площадь (expected) |
| 1 | 3 | 4 | 5 | Прямоугольный (detectedTriangle() = 8) | 6 |
| 2 | 65 | 72 | 97 | Прямоугольный(detectedTriangle() = 8) | 2340 |
| 3 | 68 | 285 | 293 | Прямоугольный(detectedTriangle() = 8) | 9690 |
| 4 | 3 | 3 | 3\*√2 | Корень из двух √2= 1,4142135623730950488016887242097  Равнобедренный прямоугольный (detectedTriangle() = 10) | ≈ 4.5 |
| 5 | 3 | 3 | 5 | Равнобедренный (тупой угол) (detectedTriangle() = 2) | 4.14578098794425 |
| 6 | 10 | 10 | 15 | Равнобедренный (тупой угол)  (detectedTriangle() = 2) | ≈ 49.607837082461074 |
| 7 | 97 | 97 | 123 | Равнобедренный (тупой угол) (detectedTriangle() = 2) | ≈ 4613.217444203124 |
| 8 | 3 | 3 | 2 | Равнобедренный (острый угол) (detectedTriangle() = 2) | ≈2.8284271247461903 |
| 9 | 10 | 10 | 9 | Равнобедренный (острый угол) (detectedTriangle() = 2) | ≈ 40.18628497385644 |
| 10 | 99 | 99 | 94 | Равнобедренный (острый угол) (detectedTriangle() = 2) | ≈ 4095.2079312288893 |
| 11 | 5 | 5 | 5 | Равносторонний (detectedTriangle() = 3) | ≈ 10.825317547305483 |
| 12 | 17 | 17 | 17 | Равносторонний (detectedTriangle() = 3) | ≈ 125.14067084685138 |
| 13 | 111 | 111 | 111 | Равносторонний (detectedTriangle() = 3) | ≈ 5335.149500014034 |
| 14 | 1 | 1 | 1 | Минимальный равносторонний(detectedTriangle() = 3) | ≈ 0.4330127018922193 |
| 15 | 10 | 6 | 5 | Разносторонний(detectedTriangle() = 4) | ≈ 11.399013115177997 |
| 16 | 26 | 31 | 12 | Разносторонний(detectedTriangle() = 4) | ≈ 151.96525096218542 |
| 17 | 113 | 158 | 256 | Разносторонний(detectedTriangle() = 4) | ≈ 5601.6429230628 |
| 18 | 10 | 3 | 2 | Несуществующий(detectedTriangle() = 0) | NaN |
| 19 | 5 | 12 | 4 | Несуществующий(detectedTriangle() = 0) | нет |
| 20 | 20 | 20 | 51 | Несуществующий(detectedTriangle() = 0) | нет |
| 21 | 1 | 1 | 0 | Несуществующий(detectedTriangle() = 0) | нет |
| 22 | 1 | 0 | 1 | Несуществующий(detectedTriangle() = 0) | нет |
| 23 | 0 | 1 | 1 | Несуществующий(detectedTriangle() = 0) | нет |
| 24 | -5 | 4 | 3 | Несуществующий(detectedTriangle() = 0) | нет |
| 25 | 4 | 3 | -5 | Несуществующий(detectedTriangle() = 0) | нет |
| 26 | 3 | -5 | 4 | Несуществующий(detectedTriangle() = 0) | нет |
| 27 | 5 | 5 | 12 | Несуществующий(detectedTriangle() = 0) | нет |
| 28 | MaxValue | MaxValue | 1 | Равнобедренный (detectedTriangle() = 2) | Вне диапазона |
| 29 | MaxValue | MaxValue | MaxValue | Равносторонний(detectedTriangle() = 3) | Вне диапазона |
| 30 | MaxValue | 500 | 400 | Несуществующий(detectedTriangle() = 0) | Вне диапазона |
| 31 | MaxValue | MaxValue-1 | MaxValue-2 | Разносторонний(detectedTriangle() = 4) | Вне диапазона |
|  | MaxValue | MaxValue | MaxValue | Равносторонний(detectedTriangle() = 3) | Вне диапазона |
|  | Double.NaN | 5 | 5 | Равнобедренный (detectedTriangle() = 2) | Double.NaN |
|  | Double.  PositiveInfinity | 5 | 5 | Равнобедренный (detectedTriangle() = 2) | Double.NaN |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Примечание: все ожидаемые результаты были сформированы мной на основе просмотра кода и понимания того, что в нём написано. ОБРАЩАЮ внимание на то, что в реальных проектах должна быть документация по проекту и эта информация должна там присутствовать.