# Чек-лист Triangle

Поскольку данная задача классическая, разумно ожидать описания ее решения в общем виде. В качестве базы взята статья Analyzing the Triangle Problem[[1]](#footnote-1).

Итоговый набор тестов и их архитектура определялись заявленными возможностями и структурой программы. Тесты разделены на две группы, соответствующие классам программы. Далее создавались тестовые классы для отдельных методов. С одной стороны, это должно обеспечить достаточное покрытие тестами, с другой, помогает избежать overengenering и нерациональной траты времени.

С учебными целями тесты выполнены и в jUnit, и в TestNG.

## Main.java

Для Main – тесты, проверяющие основные сценарии последовательного вызова методов Triangle.

### Позитивные тесты

* 1. Создание объекта Triangle.
  2. Сценарий создания корректного равностороннего треугольника, прохождения проверки, выдачи пустого сообщения об ошибке, корректное определение типа.
  3. Сценарий создания корректного равнобедренного треугольника, прохождения проверки, выдачи пустого сообщения об ошибке, корректное определение типа.
  4. Сценарий создания корректного «обычного» треугольника, прохождения проверки, выдачи пустого сообщения об ошибке, корректное определение типа
  5. Сценарий создания корректного прямоугольного «обычного» треугольника, прохождения проверки, выдачи пустого сообщения об ошибке, корректное определение типа.

### Негативные тесты

* 1. Сценарий создания некорректного треугольника с нулевой стороной, ожидание ошибки проверки, ожидание сообщения об ошибке, ожидание исключения при определении типа.
  2. Сценарий создания некорректного треугольника с невозможной комбинацией длин сторон, ожидание ошибки проверки, ожидание сообщения об ошибке, ожидание исключения при определении типа.

## Triangle.java

Для Triangle – тесты, проверяющие работу его методов.

### Проверка метода checkMessages().

* 1. В методе предусмотрено 9 сообщений об ошибках. На примерах некорректных треугольников проверяем возврат false из метода и текст ошибки.
  2. Для каждого сообщения учитываем две ситуации: сторона меньше нуля и равна нулю.

### Проверка метода checkTriangle().

Метод должен возвращать true в следующих ситуациях:

* 1. Равносторонний треугольник, положительные числа длин сторон.
  2. Прямоугольный треугольник, положительные числа длин сторон.
  3. Равнобедренные треугольники, положительные числа длин сторон.
  4. Обычный треугольник, положительные числа, разный порядок длин сторон.

Метод должен возвращать false в следующих ситуациях:

* 1. Равнобедренный прямоугольный треугольник с выходом за пределы double. Ожидаемое поведение – выдача исключения.
  2. Ноль в значениях длин сторон.
  3. Отрицательные значения длин сторон.

### Проверка метода detectTriangle().

* 1. Определение равнобедренного треугольника.
  2. Определение равностороннего треугольника.
  3. Определение прямоугольного треугольника.
  4. Определение прямоугольного равнобедренного треугольника.
  5. Определение прямоугольного обычного треугольника.
  6. Определение обычного треугольника.
  7. Определение равностороннего равнобедренного треугольника.

### Проверка метода getSquare().

* 1. Расчет площади треугольника методом Герона (применен в приложении) – проверяем корректность реализации формулы в коде.
  2. Площадь прямоугольного треугольника методом катетов (дает другую точность).
  3. Проверка выдачи исключения для попытки посчитать площадь невозможного треугольника.
  4. Проверка выдачи исключения для попытки посчитать площадь треугольника с нулевой стороной.

1. <http://www.testingeducation.org/conference/wtst3_collard5.pdf> [↑](#footnote-ref-1)