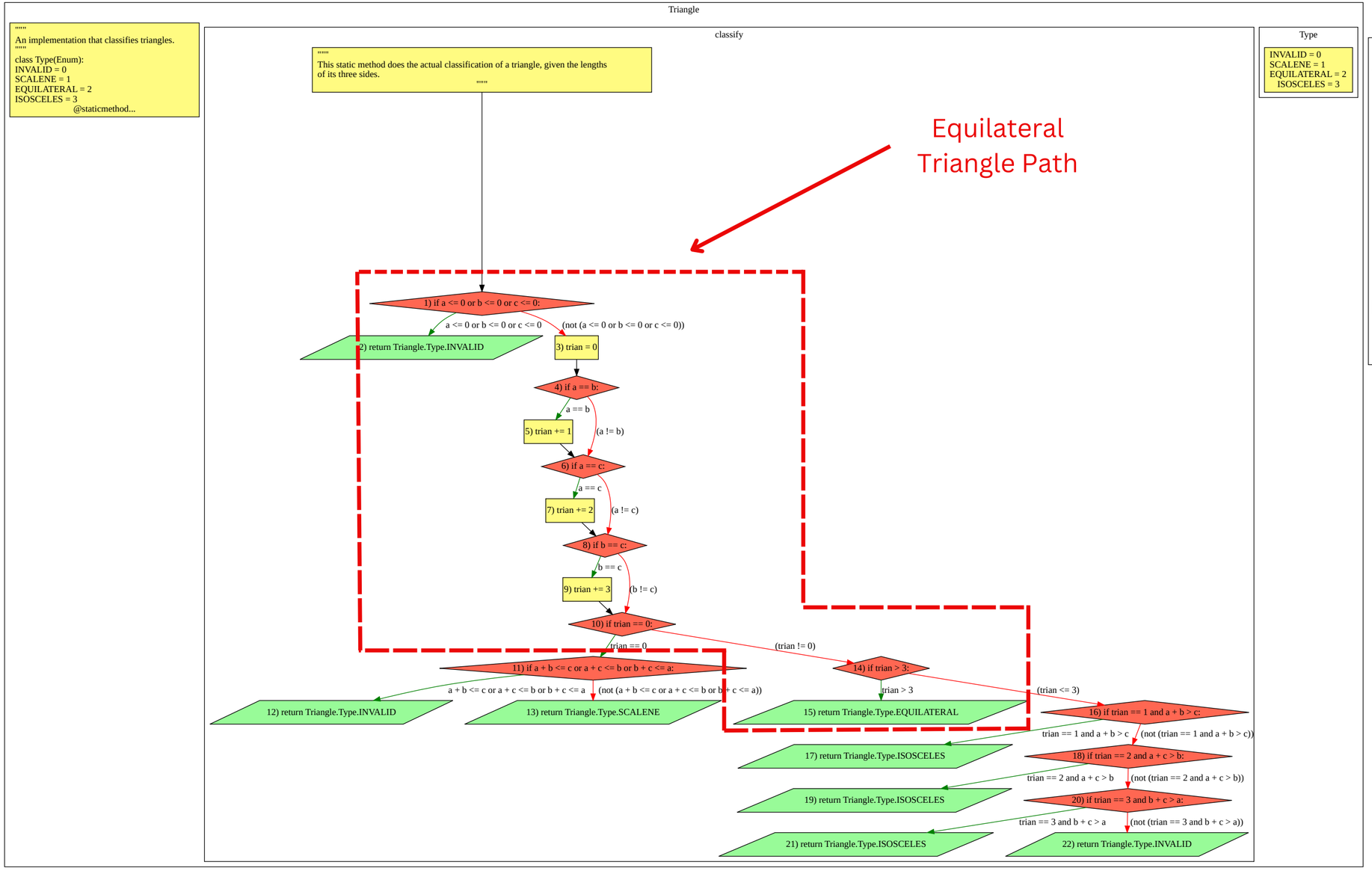
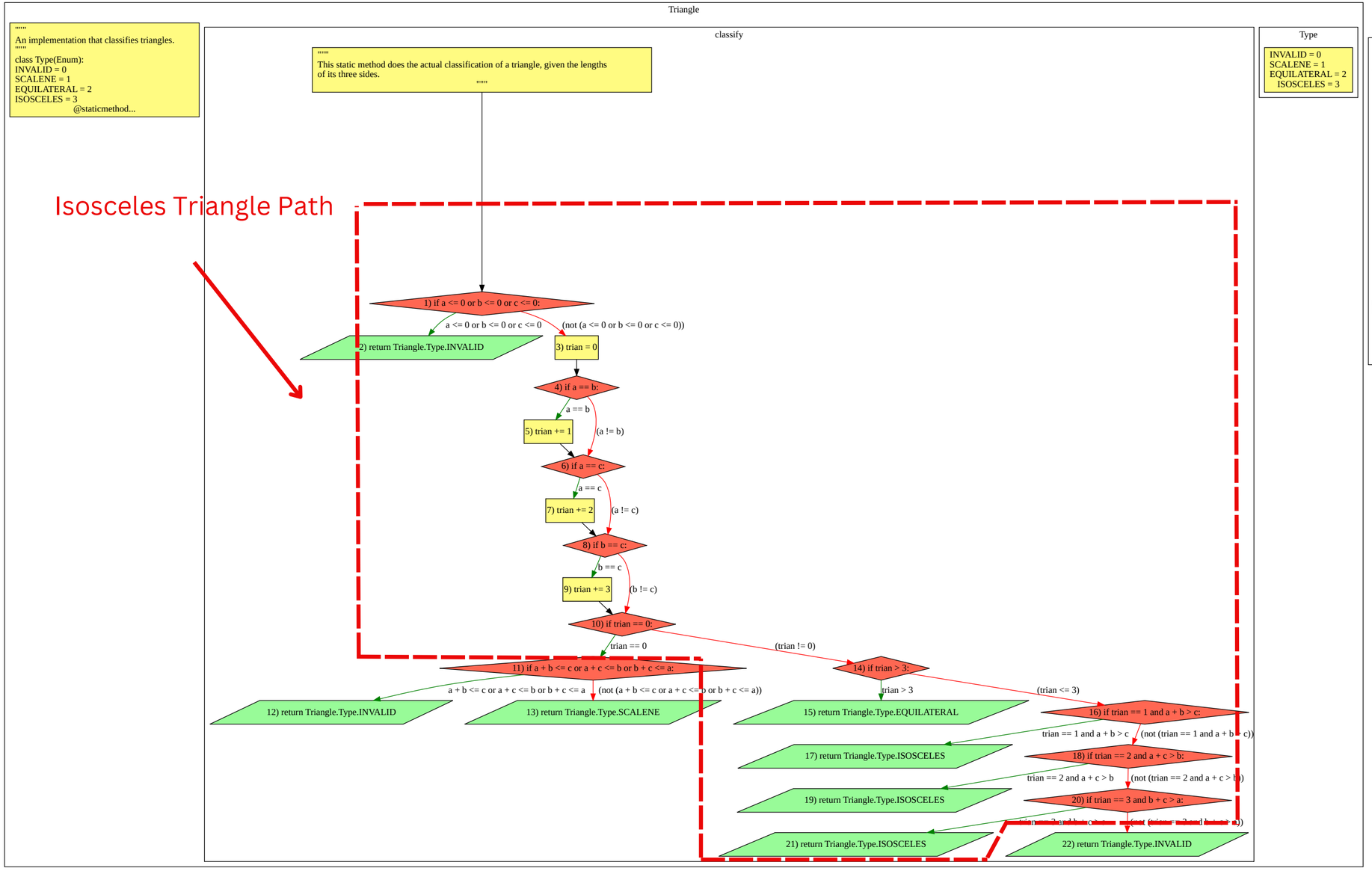
```
from enum import Enum
class Triangle:
"""
An implementation that classifies triangles.
"""


class Type(Enum):
INVALID = 0
SCALENE = 1
EQUILATERAL = 2
ISOSCELES = 3

@staticmethod
def classify(a, b, c):
"""
This static method does the actual classification of a triangle, given the lengths
of its three sides.
"""
if a <= 0 or b <= 0 or c <= 0:
return Triangle.Type.INVALID
trian = 0
if a == b:
trian += 1
if a == c:
trian += 2
if b == c:
trian += 3
if trian == 0:
if a + b <= c or a + c <= b or b + c <= a:
return Triangle.Type.INVALID
else:
return Triangle.Type.SCALENE
if trian > 3:
return Triangle.Type.EQUILATERAL
if trian == 1 and a + b > c:
return Triangle.Type.ISOSCELES
elif trian == 2 and a + c > b:
return Triangle.Type.ISOSCELES
elif trian == 3 and b + c > a:
return Triangle.Type.ISOSCELES
                return Triangle.Type.INVALID
```

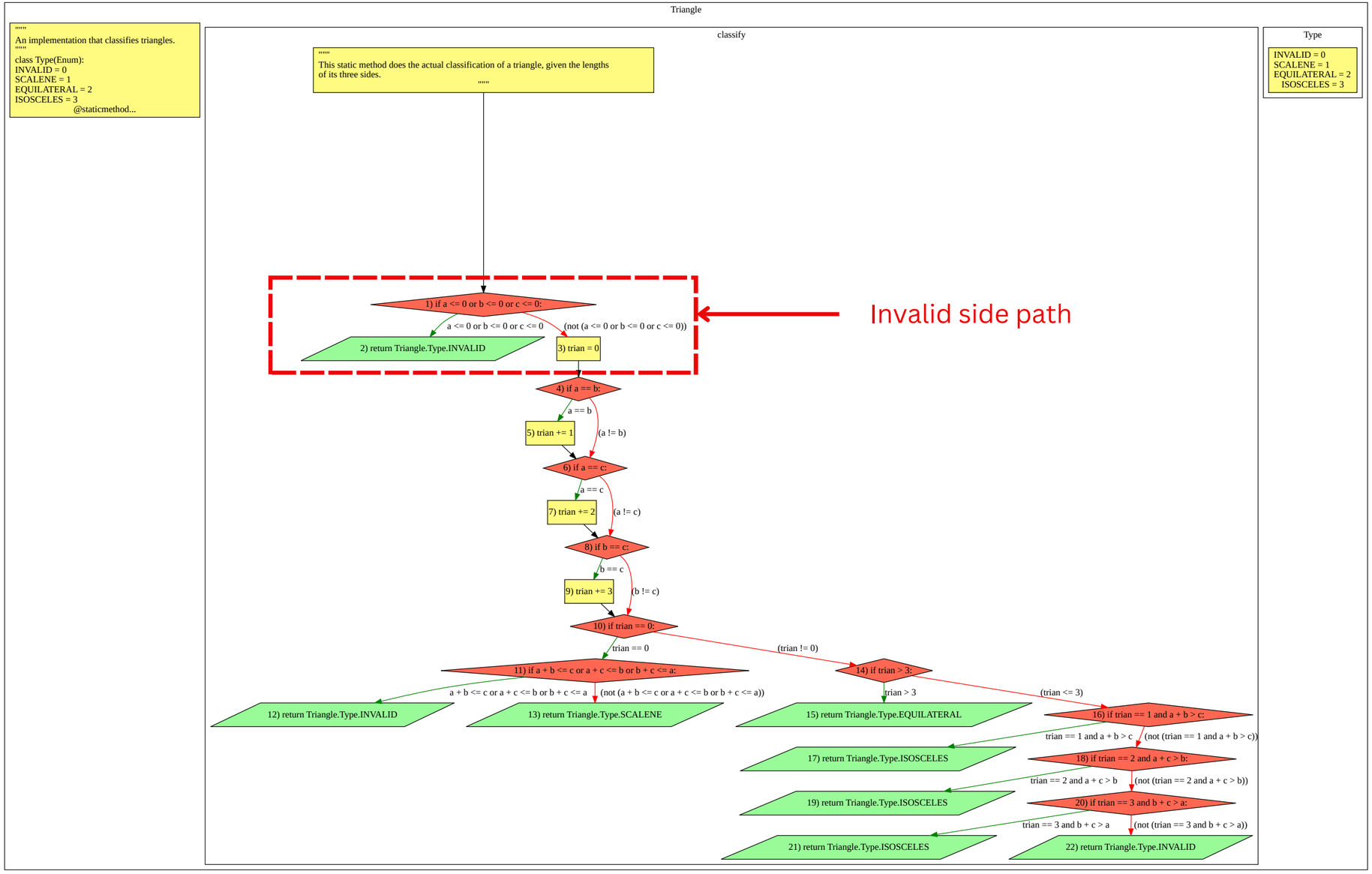


```
from enum import Enum
class Triangle:
"""
An implementation that classifies triangles.
"""


class Type(Enum):
INVALID = 0
SCALENE = 1
EQUILATERAL = 2
ISOSCELES = 3

@staticmethod
def classify(a, b, c):
"""
This static method does the actual classification of a triangle, given the lengths
of its three sides.
"""
if a <= 0 or b <= 0 or c <= 0:
return Triangle.Type.INVALID
trian = 0
if a == b:
trian += 1
if a == c:
trian += 2
if b == c:
trian += 3
if trian == 0:
if a + b <= c or a + c <= b or b + c <= a:
return Triangle.Type.INVALID
else:
return Triangle.Type.SCALENE
if trian > 3:
return Triangle.Type.EQUILATERAL
if trian == 1 and a + b > c:
return Triangle.Type.ISOSCELES
elif trian == 2 and a + c > b:
return Triangle.Type.ISOSCELES
elif trian == 3 and b + c > a:
return Triangle.Type.ISOSCELES
                                return Triangle.Type.INVALID
```



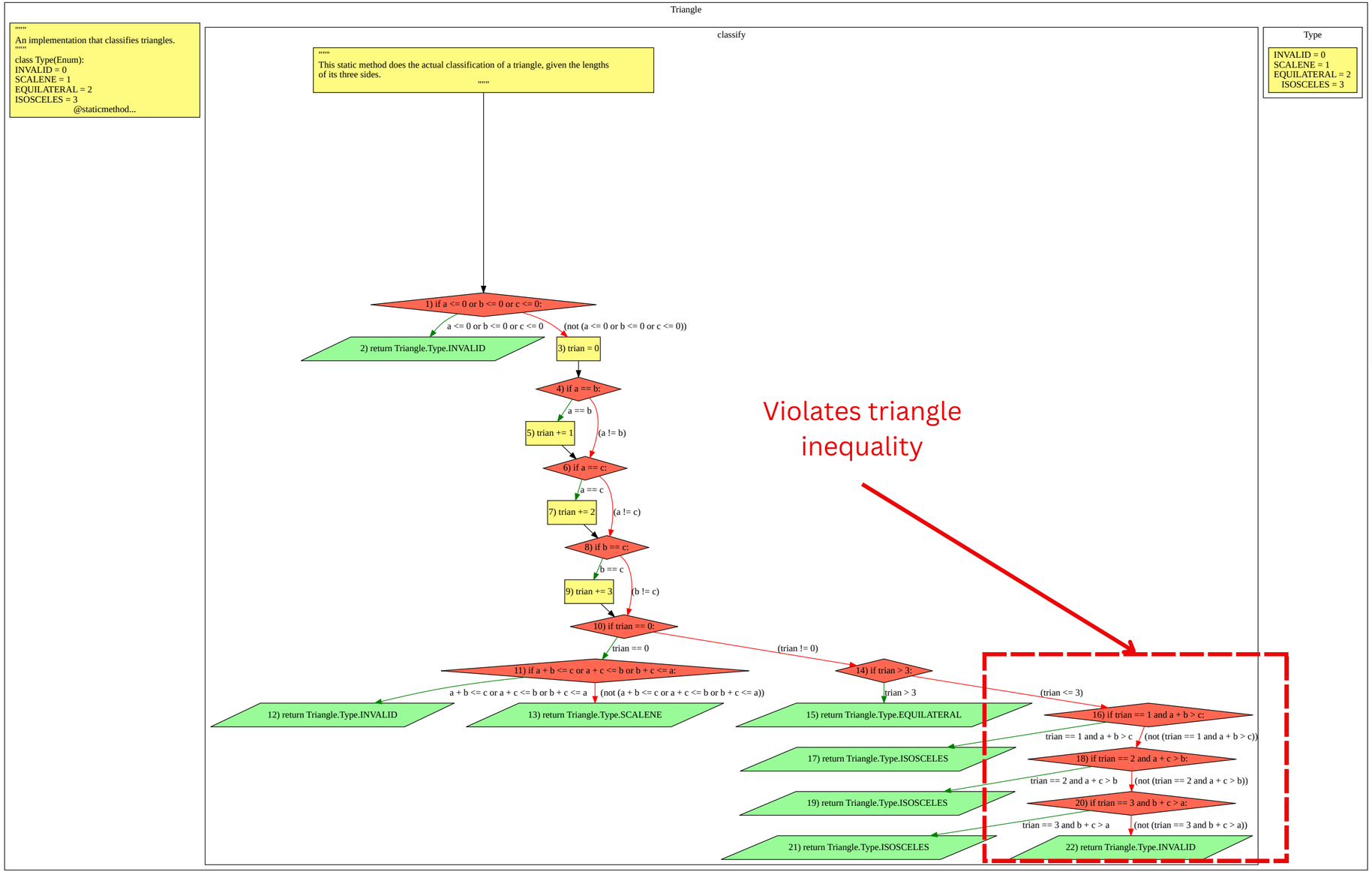
isTriangle.py

```
from enum import Enum
class Triangle:
"""
An implementation that classifies triangles.
"""


class Type(Enum):
INVALID = 0
SCALENE = 1
EQUILATERAL = 2
ISOSCELES = 3

@staticmethod
def classify(a, b, c):
"""
This static method does the actual classification of a triangle, given the lengths
of its three sides.
"""
if a <= 0 or b <= 0 or c <= 0:
return Triangle.Type.INVALID
trian = 0
if a == b:
trian += 1
if a == c:
trian += 2
if b == c:
trian += 3
if trian == 0:
if a + b <= c or a + c <= b or b + c <= a:
return Triangle.Type.INVALID
else:
return Triangle.Type.SCALENE
if trian > 3:
return Triangle.Type.EQUILATERAL
if trian == 1 and a + b > c:
return Triangle.Type.ISOSCELES
elif trian == 2 and a + c > b:
return Triangle.Type.ISOSCELES
elif trian == 3 and b + c > a:
return Triangle.Type.ISOSCELES
                                return Triangle.Type.INVALID
```



isTriangle.py

```
from enum import Enum
class Triangle:
"""
An implementation that classifies triangles.
"""


class Type(Enum):
INVALID = 0
SCALENE = 1
EQUILATERAL = 2
ISOSCELES = 3

@staticmethod
def classify(a, b, c):
"""
This static method does the actual classification of a triangle, given the lengths
of its three sides.
"""
if a <= 0 or b <= 0 or c <= 0:
return Triangle.Type.INVALID
trian = 0
if a == b:
trian += 1
if a == c:
trian += 2
if b == c:
trian += 3
if trian == 0:
if a + b <= c or a + c <= b or b + c <= a:
return Triangle.Type.INVALID
else:
return Triangle.Type.SCALENE
if trian > 3:
return Triangle.Type.EQUILATERAL
if trian == 1 and a + b > c:
return Triangle.Type.ISOSCELES
elif trian == 2 and a + c > b:
return Triangle.Type.ISOSCELES
elif trian == 3 and b + c > a:
return Triangle.Type.ISOSCELES
                                return Triangle.Type.INVALID
```



isTriangle.py