```java
@Test
public void casoListaVazia() {
        ArrayList<Triangulo> lt =  new ArrayList<>();
        ListaDeTriangulos lista = new ListaDeTriangulos(lt);
        Assertions.assertFalse(lista.containsApenasTriangulo());
}
```

```java
@Test
public void casoContemApenasTriangulos() {
        ArrayList<Triangulo> lt =  new ArrayList<>();
        lt.add(new Triangulo(1, 1, 1));
        lt.add(new Triangulo(5, 5, 6));
        lt.add(new Triangulo(3, 4, 5));
        ListaDeTriangulos lista = new ListaDeTriangulos(lt);
        Assertions.assertTrue(lista.containsApenasTriangulo());
}
```

```java
@Test
public void casoContemNotTriangulo() {
        ArrayList<Triangulo> lt =  new ArrayList<>();
        lt.add(new Triangulo(1, 1, 1));
        lt.add(new Triangulo(5, 5, 6));
        lt.add(new Triangulo(3, 4, 5));
        lt.add(new Triangulo(3, 4, 8));
        ListaDeTriangulos lista = new ListaDeTriangulos(lt);
        Assertions.assertFalse(lista.containsApenasTriangulo());
}
```

Complexidade ciclomática = V(G) = NúmeroDeArcos – NúmeroDeNodos + 2

NúmeroDeArcos = 8
NúmeroDeNodos = 7

V(G) = 8 - 7 + 2 = 3

Os 3 caminhos independentes e seus respectivos casos de teste são:
(1,2,7): CasoDeTeste1 -> vetor vazio (R: false)
(1,2,3,4,6,2,...): CasoDeTeste2 -> não encontrou Tipo.NOT_TRIANGULO (R: true)
(1,2,3,5,6,2,...): CasoDeTeste3 -> encontrou Tipo.NOT_TRIANGULO (R: false)

```java
public boolean containsApenasTriangulo(){
    int counter = 0;
    boolean isOnlyTriangulo = !(this.listaDeTriangulos.isEmpty());
    while (counter < this.listaDeTriangulos.size()) {
        if (this.listaDeTriangulos.get(counter).getForma() == Tipo.NOT_TRIANGULO) {
            isOnlyTriangulo = false;
        }
        counter++;
    }
    return isOnlyTriangulo;
}
```