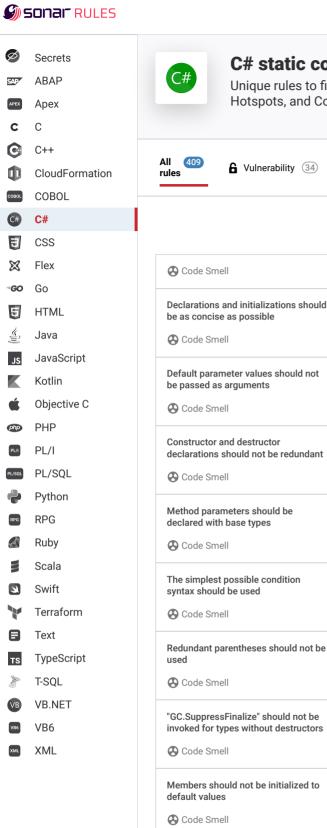
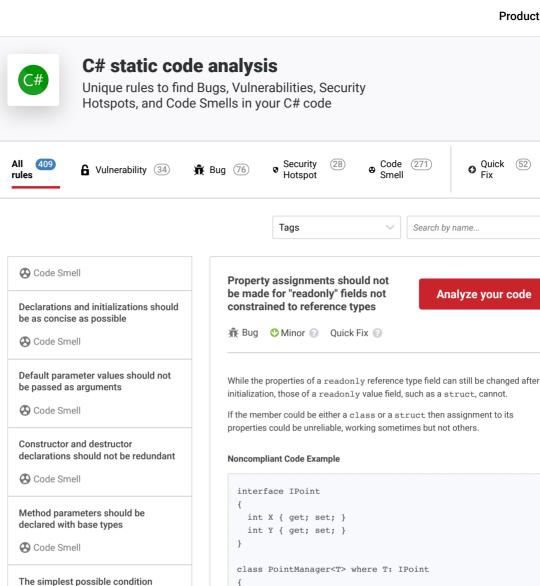
Q







readonly T point; // this could be a struct public PointManager(T point) this.point = point; public void MovePointVertically(int newX) point.X = newX; //Noncompliant; if point is a struct, th Console.WriteLine(point.X); }

## **Compliant Solution**

Sequential tests should not check the

Redundant modifiers should not be

Methods and properties that don't

access instance data should be static

same condition Code Smell

Code Smell

Code Smell

```
interface IPoint
{
  int X { get; set; }
  int Y { get; set; }
class PointManager<T> where T : IPoint
  readonly T point; // this could be a struct
  public PointManager(T point)
    this.point = point;
  public void MovePointVertically(int newX) // assignment ha
    Console.WriteLine(point.X):
```

"Exception" should not be caught when not required by called methods

Code Smell

"sealed" classes should not have "protected" members

Code Smell

Underscores should be used to make large numbers readable

Code Smell

"ToString()" calls should not be redundant

Code Smell

```
interface IPoint
 int X { get; set; }
 int Y { get; set; }
class PointManager<T> where T : class, IPoint
 readonly T point; // this can only be a class
 public PointManager(T point)
   this.point = point;
 public void MovePointVertically(int newX)
   point.X = newX; // this assignment is guaranteed to wor
   Console.WriteLine(point.X);
}
```

Available In:

sonarlint ⊕ | sonarcloud 👌 | sonarqube

© 2008-2022 SonarSource S.A., Switzerland. All content is copyright protected. SONAR, SONARSOURCE, SONARLINT, SONARQUBE and SONARCLOUD are trademarks of SonarSource S.A. All other trademarks and copyrights are the property of their respective owners. All rights are expressly reserved. Privacy Policy